

# Wing IDE Personal Reference Manual

## Wing IDE Personal

Wingware  
[www.wingware.com](http://www.wingware.com)

Version 4.1.14  
September 6, 2013

## Contents

### Introduction

- 1.1. Product Levels
- 1.2. Licenses
- 1.3. Supported Platforms
- 1.4. Supported Python versions
- 1.5. Technical Support
- 1.6. Prerequisites for Installation
- 1.7. Installing
- 1.8. Running the IDE
- 1.9. Installing your License
- 1.10. User Settings Directory
- 1.11. Upgrading
  - 1.11.1. Migrating from older versions of Wing
  - 1.11.2. Fixing a Failed Upgrade
- 1.12. Installation Details and Options
  - 1.12.1. Linux Installation Notes
  - 1.12.2. Installing Extra Documentation
- 1.13. Removing Wing IDE
- 1.14. Command Line Usage

### Customization

- 2.1. Keyboard Personalities
  - 2.1.1. Key Equivalents
  - 2.1.2. Key Maps
  - 2.1.3. Key Names
- 2.2. User Interface Options
  - 2.2.1. Windowing Policies
  - 2.2.2. User Interface Layout
  - 2.2.3. Altering Text Display
  - 2.2.4. Setting Overall Display Theme
- 2.3. Preferences
- 2.4. File Filters

### Project Manager

- 3.1. Creating a Project
- 3.2. Removing Files and Directories
- 3.3. Saving the Project
- 3.4. Sorting the View
- 3.5. Navigating to Files
  - 3.5.1. Keyboard Navigation
- 3.6. Project-wide Properties
  - Environment
  - Debug
  - Options
  - Extensions
  - 3.6.1. Variable Expansion
- 3.7. Per-file Properties
  - File Attributes
  - Editor
  - Environment
  - Debug

## Source Code Editor

- 4.1. Syntax Colorization
- 4.2. Right-click Editor Menu
- 4.3. Navigating Source
- 4.4. File status and read-only files
- 4.5. Transient vs. Sticky Editors
- 4.6. Auto-completion
- 4.7. Source Assistant
- 4.8. Indentation
  - 4.8.1. How Indent Style is Determined
  - 4.8.2. Indentation Preferences
  - 4.8.3. Indentation Policy
  - 4.8.4. Auto-Indent
  - 4.8.5. The Tab Key
  - 4.8.6. Checking Indentation
  - 4.8.7. Changing Block Indentation
  - 4.8.8. Indentation Manager

- 4.9. Structural Folding
- 4.10. Brace Matching
- 4.11. Support for files in .zip or .egg files
- 4.12. Keyboard Macros
- 4.13. Notes on Copy/Paste
- 4.14. Auto-reloading Changed Files
- 4.15. Auto-save

## Search/Replace

- 5.1. Toolbar Quick Search
- 5.2. Keyboard-driven Mini-Search/Replace
- 5.3. Search Tool
- 5.4. Search in Files Tool
  - 5.4.1. Replace in Multiple Files
- 5.5. Wildcard Search Syntax

## Interactive Python Shell

- 6.1. Python Shell Auto-completion
- 6.2. Python Shell Options

## Debugger

- 7.1. Quick Start
- 7.2. Specifying Main Entry Point
- 7.3. Debug Properties
- 7.4. Setting Breakpoints
- 7.5. Starting Debug
- 7.6. Debugger Status
- 7.7. Flow Control
- 7.8. Viewing the Stack
- 7.9. Viewing Debug Data
  - 7.9.1. Stack Data View
    - 7.9.1.1. Popup Menu Options
    - 7.9.1.2. Filtering Value Display
  - 7.9.2. Problems Handling Values
- 7.10. Debug Process I/O

- 7.10.1. External I/O Consoles
  - 7.10.2. Disabling Debug Process I/O Multiplexing
- 7.11. Debugging Multi-threaded Code
- 7.12. Managing Exceptions
  - Exception Reporting Mode
  - Reporting Logged Exceptions
  - Exception Type Filters

## Advanced Debugging Topics

- 8.1. Debugging Externally Launched Code
  - 8.1.1. Importing the Debugger
  - 8.1.2. Debug Server Configuration
  - 8.1.3. Debugger API
  - 8.1.4. Debugging Embedded Python Code
- 8.2. Remote Debugging
  - 8.2.1. File Location Maps
    - 8.2.1.1. File Location Map Examples
  - 8.2.2. Remote Debugging Example
  - 8.2.3. Installing the Debugger Core
- 8.3. OS X Debugging Notes
  - System-Provided Python
  - MacPorts Python
  - Debugging 32-bit Python on a 64-bit System
- 8.4. Debugger Limitations

## Source Code Analysis

- 9.1. How Analysis Works
- 9.2. Static Analysis Limitations
- 9.3. Helping Wing Analyze Code
- 9.4. Analysis Disk Cache

## Trouble-shooting Guide

- 10.1. Trouble-shooting Failure to Start
- 10.2. Issues on Microsoft Windows
- 10.3. Issues on Linux
- 10.4. Trouble-shooting Failure to Debug

- 10.4.1. Failure to Start Debug
- 10.4.2. Failure to Stop on Breakpoints or Show Source Code
- 10.4.3. Failure to Stop on Exceptions
- 10.4.4. Extra Debugger Exceptions
- 10.5. Obtaining Diagnostic Output
- 10.6. Speeding up Wing
- 10.7. Failure to Detect HTTP Proxy and Connect to wingware.com
- 10.8. Trouble-shooting Failure to Open Filenames Containing Spaces
- 10.9. Trouble-shooting Failure to Print

## Preferences Reference

- User Interface
- Files
- Editor
- Debugger
- Source Analysis
- Network
- Internal Preferences
- Core Preferences
- User Interface Preferences
- Editor Preferences
- Project Manager Preferences
- Debugger Preferences
- Source Analysis Preferences

## Command Reference

- 12.1. Top-level Commands
  - Application Control Commands
  - Dock Window Commands
  - Document Viewer Commands
  - Global Documentation Commands
  - Window Commands
  - Wing Tips Commands
- 12.2. Project Manager Commands
  - Project Manager Commands

## Project View Commands

### 12.3. Editor Commands

Editor Browse Mode Commands

Editor Insert Mode Commands

Editor Non Modal Commands

Editor Panel Commands

Editor Replace Mode Commands

Editor Split Commands

Editor Visual Mode Commands

Active Editor Commands

General Editor Commands

Shell Or Editor Commands

### 12.4. Search Manager Commands

Toolbar Search Commands

Search Manager Commands

Search Manager Instance Commands

### 12.5. Debugger Commands

Debugger Commands

Debugger Watch Commands

Call Stack View Commands

Exceptions Commands

## Key Binding Reference

### 13.1. Normal Personality

### 13.2. Emacs Personality

### 13.3. VI/VIM Personality

### 13.4. Visual Studio Personality

### 13.5. OS X Personality

### 13.6. Eclipse (Experimental) Personality

### 13.7. Brief Personality

## License Information

### 14.1. Wing IDE Software License

### 14.2. Open Source License Information

---

Wingware, the feather logo, Wing IDE, Wing IDE 101, Wing IDE Personal, Wing IDE Professional, and “The Intelligent Development Environment” are trademarks or registered trademarks of Wingware in the United States and other countries.

Disclaimers: The information contained in this document is subject to change without notice. Wingware shall not be liable for technical or editorial errors or omissions contained in this document; nor for incidental or consequential damages resulting from furnishing, performance, or use of this material.

Hardware and software products mentioned herein are used for identification purposes only and may be trademarks of their respective owners.

Copyright (c) 1999-2013 by Wingware. All rights reserved.:

Wingware  
P.O. Box 400527  
Cambridge, MA 02140-0006  
United States of America



# Introduction

Thanks for choosing Wingware's Wing IDE Personal! This manual will help you get started and serves as a reference for the entire feature set of this product.

The manual is organized by major functional area of Wing IDE, including source code editor, project manager, and debugger. Several appendices document the entire command set, provide pointers to resources and tips for Wing and Python users, and list the full software license.

The rest of this chapter describes how to install and start using Wing IDE Personal. If you hate reading manuals, you should be able to get started by reading this chapter only, or try the **quick start guide** or **tutorial**.

## Key Concepts

Throughout this manual, key concepts, important notes, and non-obvious features are highlighted in the same way as this paragraph. If you are skimming only, look for these marks.

## 1.1. Product Levels

This manual is for the Wing IDE Personal product level of the Wing IDE product line, which currently includes Wing IDE Professional, Wing IDE Personal, and Wing IDE 101.

Wing IDE Professional is the full-featured Wing IDE product, and may be licensed for commercial or non-commercial uses. Wing IDE Personal is for non-commercial use only and contains a subset of the features found in Wing IDE Professional. Both products are commercial products for sale from our website; Wing IDE Personal is not a free download.

Wing IDE 101 is a heavily scaled back IDE that was designed for teaching entry level computer science courses. It is free to download and use for educational and personal use.

Wing IDE Professional, Wing IDE Personal, and Wing IDE 101 are independent products and may be installed at the same time on your system without interfering with each other.

For a list of features in each product level, please refer to <http://wingware.com/wingide/features>.

## 1.2. Licenses

Wing IDE requires a separate license for each developer working with the product. For the full license text, see the **Software License**.

### License Activation

To run for more than 10 minutes, Wing IDE requires activation of a time-limited trial or permanent purchased license. Time-limited trials last for 10 days and can be renewed two times, for a total of 30 days.

An activation ties the license to the machine through a series of checks of the hardware connected to the system. This information is never transmitted over the internet. Instead an SHA hash of some of the values is passed back and forth so that the machine will be identifiable without us knowing anything specific about it.

The machine identity metrics used for activation are designed to be forgiving so that replacing parts of your machine's hardware or upgrading the machine will usually not require another activation. By the same token, activating multiple times on the same machine (for example if the activation file is lost) usually does not increase your activation count.

Licenses come with ten activations per year by default and additional activations can be obtained from the [self-serve license manager](#) or by emailing [sales at wingware.com](mailto:sales@wingware.com). As a fall-back in cases of emergency where we cannot be contacted and you don't have an activation, Wing IDE will run for 10 minutes at a time without any license at all, or a trial license can be used until any license problem is resolved.

See **Installing Your License** for more information on obtaining and activating licenses.

## 1.3. Supported Platforms

This version of Wing IDE is available for Microsoft Windows, Linux, Mac OS X (with X11 Server), and some other operating systems where customers compile the product from source code.

## Microsoft Windows

Wing IDE supports Windows 2000, XP, 2003 Server, Vista, Windows 7, and Windows 8 for Intel processors. Windows 95, 98, and ME are not supported and will not work. Windows NT4 is not supported but *may* work with IE5+ installed.

## Linux/Intel

Wing IDE runs on Linux versions with glibc2.2 or later (e.g. Ubuntu 6+, RedHat 7.1+, Mandrake 8.0+, SUSE 7.1+, and Debian 3.0+).

On Suse, you may need to install the gmp and python packages, or install Python from source, since Python is not installed by default here.

## Mac OS X

Wing IDE runs on Mac OS X 10.3.9+. Wing IDE for OS X also requires an X11 Server and Window Manager. For details see **OS X Quick Start Guide**.

## Other Platforms

Wing IDE can be compiled from source by customers wishing to use it on other operating systems (such as Linux PPC, Free BSD, or Solaris). This requires a [non-disclosure agreement](#).

# 1.4. Supported Python versions

Wing supports CPython 2.0 through 3.3, Stackless Python 2.4 through 3.2, and cygwin Python 2.2 through 2.5. Wing can also be used with IronPython and Jython, but the debugger will not work with these implementations of Python.

Wing's debugger is pre-built for each of these versions of Python with and without `--with-pydebug`. Both 32-bit and 64-bit compilations are supported. CPython `--with-framework` builds are also supported on OS X. If necessary, it is possible for customers to compile Wing's debugger against other custom versions of Python.

Before installing Wing, you may need to [download Python](#) and install it if you do not already have it on your machine.

*On Windows*, Python must be installed using one of the installers from the python.org (or by building from source if desired).

*On Linux*, most distributions come with Python. Installing Python is usually only necessary on SUSE or a custom-built Linux installation.

*On SUSE Linux*, you can install the gmp and python packages that come with your distribution, or install from the materials available through the links given above.

*On Mac OS X*, Wing IDE only supports Python 2.2 and later.

## 1.5. Technical Support

If you have problems installing or using Wing IDE, please submit a bug report or feedback using the **Submit Bug Report** or **Submit Feedback** items in Wing IDE's **Help** menu.

Wingware Technical Support can also be contacted by email at [support at wingware.com](mailto:support@wingware.com), or online at <http://wingware.com/support>.

Bug reports can also be sent by email to [bugs at wingware.com](mailto:bugs@wingware.com). Please include your OS and product version number and details of the problem with each report.

If you are submitting a bug report via email, see **Obtaining Diagnostic Output** for more information on how to capture a log of Wing IDE and debug process internals. Whenever possible, these should be included with email-based bug reports.

## 1.6. Prerequisites for Installation

To run Wing IDE, you will need to obtain and install the following, if not already on your system:

### Prerequisites for all platforms

- A [downloaded](#) copy of Wing IDE
- A **supported version of Python**
- A working TCP/IP network configuration (for the debugger; no outside access to the internet is required)

### Additional Prerequisites for Mac OS X

To run Wing IDE on OS X, you will need an X11 window server. We strongly recommend [XQuartz](#) since the X11 provided with OS X often contains more bugs that affect key bindings, the clipboard, and integration with Spaces.

See the **OS X How-To** for details on installing and using Wing on OS X.

## 1.7. Installing

Before installing Wing IDE, be sure that you have installed the **necessary prerequisites**. If you are upgrading from a previous version, see **Upgrading** first.

**Note:** The installation location for Wing IDE is referred to as `WINGHOME`. On OS X this is the `Contents/MacOS` directory within Wing's `.app` folder.

### Windows

Install Wing IDE by running the downloaded executable. Wing's files are installed by default in `C:\Program Files\Wing IDE Personal 4.1`, but this location may be modified during installation. Wing will also create a **User Settings Directory** in the location appropriate for your version of Windows. This is used to store preferences and other settings.

The Windows installer supports a `/silent` command line option that uses the default options, including removing any prior install of version 4.1 of Wing IDE. If a prior install is removed, a dialog with a progress bar will appear. You can also use a `/dir=<dir name>` option to specify an alternate installation directory.

### Linux (glibc 2.2+)

Use the RPM, Debian package, or tar file installer as appropriate for your system type. Installation from packages is at `/usr/lib/wingide-personal4.1` or at the selected location when installing from the tar file. Wing will also create a **User Settings Directory** in `~/.wingpersonal4`, which is used to store preferences and other settings.

For more information, see the **Linux installation details**.

### Mac OS X 10.3.9+

Wing IDE on Mac OS X requires that you first install an X11 Server. For details on installing and running on OS X, see the **OS X Quickstart**.

## 1.8. Running the IDE

For a quick introduction to Wing's features, refer to the **Wing IDE Quickstart Guide**. For a more gentle in-depth start, see the **Wing IDE Tutorial**.

**On Windows**, start Wing IDE from the Program group of the Start menu. You can also start Wing from the command line with `wing-personal.exe` (located inside the Wing IDE installation directory).

**On Linux/Unix**, just execute `wing-personal4.1` (or `wing-personal` located inside the Wing IDE installation directory).

**On Mac OS X**, start Wing IDE by double clicking on the app folder or from the command line using `Contents/MacOS/wing` inside the Wing IDE app folder. For the latter, you will need to start your X11 Server manually first and may need to set your `DISPLAY` environment variable.

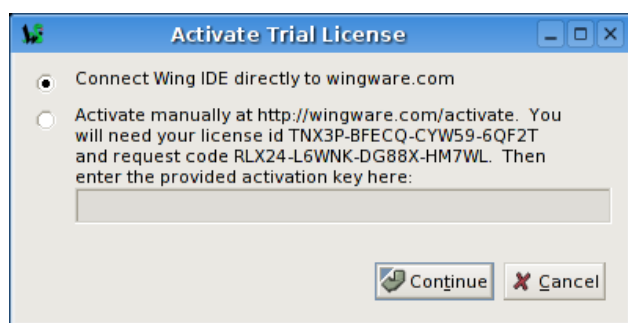
## 1.9. Installing your License

Wing IDE requires a time-limited trial or permanent license and the license needs to be activated on each machine (see the **Licenses** section for general information). When Wing IDE is first started, you can obtain a trial licence, purchase a permanent license, install & activate a permanent license, or use Wing for up to 10 minutes without any license:



### Trial Licenses

Trial licenses allow evaluation of Wing IDE for 10 days, with an option to extend the evaluation twice for up to 30 days total (or more on request). The most convenient way to obtain a trial license is to ask Wing IDE to connect directly to `wingware.com` (via http, TCP/IP port 80). After the trial license is obtained, Wing will not attempt to connect to `wingware.com` (or any other site) unless you submit feedback or a bug report through the **Help** menu.



If you're unable or unwilling to connect Wing IDE directly to [wingware.com](http://wingware.com), you can go to <http://wingware.com/activate> and enter the license id and activation request number obtained from Wing. After entering this information, you will be given an activation key which you can enter into Wing's dialog box to complete the activation. This is exactly the same exchange of information that occurs when Wing IDE connects directly to [wingware.com](http://wingware.com) to obtain a trial license.

If activation fails, Wing will provide a way to configure an http proxy. Wing tries to detect and use proxies by default but in some cases they will need to be manually configured. Please ask your network administrator if you do not know what proxy settings to use. See also [how to determine proxy settings](#).

If you run into problems or need additional evaluation time, please email us at [sales at wingware.com](mailto:sales@wingware.com).

## Permanent Licenses

Permanent licenses and upgrades may be purchased in the online store at <http://wingware.com/store>. Permanent licenses include free upgrades through the 4.\* version series. Wing IDE Professional licenses also allow access to the product source code via <http://wingware.com/downloads> (requires signed [non-disclosure agreement](#)).

## Activating on Shared Drives

When Wing is installed on a shared drive (for example a USB keydrive, or on a file server), the **User Settings Directory** where the license activation is stored may be accessed from several different computers.

In this case, Wing must be activated once on each computer. The resulting extra activations will be stored as `license.act1`, `license.act2`, and so forth, and Wing will automatically select the appropriate activation depending on where it is running.

## Obtaining Additional Activations

If you run out of activations, you can use the [self-serve license manager](#) or email us at

[sales at wingware.com](http://sales.at.wingware.com) to obtain additional activations on any legitimately purchased license.

## 1.10. User Settings Directory

The first time you run Wing, it will create your **User Settings Directory** automatically. This directory is used to store your license, preferences, auto-save files, recent lists, and other files used internally by Wing. If the directory cannot be created, Wing will exit.

The settings directory is created in a location appropriate to your operating system. The location is listed as your **Settings Directory** in the **About Box** accessible from the **Help** menu.

These are the locations used by Wing:

**Linux/Unix** -- ~/.wingpersonal4 (a sub-directory of your home directory)

**Windows** -- In **Wing Personal 4** within the per-user application data directory. The location varies by version of Windows. For Windows 2000 and XP running on c: with an English localization the location is:

```
c:\Documents and Settings\${username}\Application Data\Wing Personal 4
```

For Vista running on c: with an English localization the location is:

```
c:\Users\${username}\AppData\Roaming\Wing Personal 4
```

Wing also creates a **Cache Directory** that contains the source analysis cache. This is often but not always in the same location as the above. On Windows, this directory is usually in the per-user directory under **Local Settings** on 2000 and XP and under **Local** on Vista. This directory is also listed in the **About Box**.

## 1.11. Upgrading

If you are upgrading within the same minor version number of Wing (for example from 3.0 to 3.0.x) this will replace your previous installation. Once you have upgraded, your previous preferences and settings should remain and you should immediately be able to start using Wing.



If you are upgrading across major releases (for example from 2.1 to 3.0), this will install the new version along side your old version of Wing.

To install an upgrade, follow the steps described in **Installing**

### 1.11.1. Migrating from older versions of Wing

Moving to Wing IDE 4.x from earlier versions should be easy. The first time you start Wing IDE 4.x, it will automatically convert your preferences from Wing IDE 2.x or 3.x and place them into your **User Settings Directory**.

Wing IDE 4.x can be installed and used side by side with older versions of Wing and operates completely independently.

- **Licensing**

Licenses for Wing IDE 1.x through 3.x must be upgraded before they can be activated for Wing IDE 4.x.

Wing IDE 2.x and 3.x licenses can be upgraded in our [online store](#). Wing IDE 1.x licenses must be upgraded by contacting [sales@wingware.com](mailto:sales@wingware.com)

- **Converting Projects**

Wing IDE 2.x and 3.x project files will be converted as they are opened and marked untitled so they can be saved under a new name. You should not overwrite your old project files as long as you still plan to use your earlier version of Wing IDE with them. Wing IDE 4.x project files cannot be read by earlier versions of Wing.

When version 2.x projects are converted, Wing will automatically replace your old project's file list with one or more auto-updating directory entries. You may wish to prune the project's contents after conversion, by adding and removing items, or editing their settings by right clicking on them and selecting **Directory Properties**.

### 1.11.2. Fixing a Failed Upgrade

In rare cases upgrading may fail to overwrite old files, resulting in random or bizarre behaviors and crashing. The fix for this problem is to completely uninstall and manually remove remaining files before installing the upgrade again.

## Windows

To uninstall on Windows, run the Add/Remove Programs control panel to uninstall Wing IDE. Then go into the directory where Wing was located and manually remove any remaining folders and files.

### **Linux RPM**

If you installed Wing IDE for Linux from RPM, issue the command `rpm -e wingide4.1`. Then go into `/usr/lib/wingide4.1` and remove any remaining files and directories.

### **Linux Debian**

If you installed Wing IDE for Linux from Debian package, issue the command `dpkg -r wingide4.1`. Then go into `/usr/lib/wingide4.1` and remove any remaining files and directories.

### **Linux Tar**

If you installed Wing IDE for Linux from the tar distribution, find your Wing installation directory and run the `wing-uninstall` script located there. Once done, manually remove any remaining files and directories.

### **Mac OS X**

On Mac OS X, just drag the entire Wing IDE application folder to the trash.

If this procedure does not solve the problem, try moving aside the **User Settings Directory** and then starting Wing. If this works, try restoring files from the old user settings directory one by one to find the problem. Key files to try are `license.act*`, `preferences` and `recent*`. Then submit a bug report to [support@wingware.com](mailto:support@wingware.com) with the offending file.

## **1.12. Installation Details and Options**

This section provides some additional detail for installing Wing and describes installation options for advanced users.

### **1.12.1. Linux Installation Notes**

On Linux, Wing can be installed from RPM, Debian package, or from tar archive. Use the latter if you do not have root access on your machine or wish to install Wing somewhere other than `/usr/lib/wingide-personal4.1`. Be sure to use the 64-bit packages if you are on a 64-bit system.

### Installing from RPM:

Wing can be installed from an RPM package on RPM-based systems, such as RedHat and Mandriva. To install, run `rpm -i wingide-personal4.1-4.1.14-1.i386.rpm` as root or use your favorite RPM administration tool to install the RPM. Most files for Wing are placed under the `/usr/lib/wingide-personal4.1` directory and the `wing-personal4.1` command is placed in the `/usr/bin` directory.

### Installing from Debian package:

Wing can be installed from a Debian package on Debian, Ubuntu, and other Debian-based systems.

You will need to install `enscript` before installing Wing, if it's not already on your system.

To install, run `dpkg -i wingide-personal4.1_4.1.14-1_i386.deb`

as root or use your favorite package administration tool to install. Most files for Wing are placed under the `/usr/lib/wingide-personal4.1` directory and the `wing-personal4.1` command is placed in the `/usr/bin` directory.

### Installing from Tar Archive:

Wing may also be installed from a tar archive. This can be used on systems that do not use RPM or Debian packages, or if you wish to install Wing into a directory other than `/usr/lib/wingide-personal4.1`. Unpacking this archive with `tar -zxvf wingide-personal-4.1.14-1-i386-linux.tar.gz` will create a `wingide-personal-4.1.14-1-i386-linux` directory that contains the `wing-install.py` script and a `binary-package.tar` file.

Running the `wing-install.py` script will prompt for the location to install Wing, and the location in which to place the executable `wing-personal4.1`. These locations default to `/usr/local/lib/wingide-personal` and `/usr/local/bin`, respectively. The install program must have read/write access to both of these directories, and all users running Wing must have read access to both.

### Using System-wide GTK:

By default, Wing IDE runs with its own copy of GTK2 and does not pick up on the system-configured theme. This is done to avoid problems and bugs sometimes brought out by binary incompatibilities in GTK versions.

On Linux versions that include GTK version 2.6 or later, you can tell Wing IDE to use the system-defined GTK2 by setting the **System GTK** preference or running with the `--system-gtk` command line argument.

Using the system-wide GTK2 in this way generally works quite well but may result in crashing or display bugs due to binary incompatibilities in GTK and related libraries. If you set the preference and Wing fails to start, specify the `--private-gtk` command line option to override the preference.

### Non-ascii File Paths on Older Linux Systems:

Some older Linux versions require setting the environment variable `G_BROKEN_FILENAMES` before Wing IDE's file open/save dialog will work properly with file paths that contain non-ascii characters. The environment variable is already set on some systems where it is needed but this is not always the case.

### Debugging 32-bit Python on 64-bit Systems

On a 64-bit system where you need to debug 32-bit Python, you will need to install the 32-bit version of Wing. This version can also debug 64-bit Python.

Installing the 32-bit version of Wing may require installing some compatibility packages as follows:

On 64-bit Ubuntu and Debian systems, you need to first install the 32 bit compatibility libraries. This is the `ia32-libs` package on Ubuntu. On Debian and Ubuntu 9+, the `ia32-libs-gtk` package is needed as well. Then install the 32-bit Wing with the command `dpkg -i --force-architecture --force-depends wingide-personal4.1_4.1.14-1_i386.deb`. The package contains what you need to run your debug process with 64-bit Python but Wing itself runs as a 32-bit application.

On CentOS 64-bit systems, installing the `libXtst.i386` and `gtk2*386` packages with `yum` provides the necessary 32 bit support.

On Arch linux, the necessary packages are instead named `lib32-glibc` and `lib32-gtk2`.

## 1.12.2. Installing Extra Documentation

On Windows, Wing looks for local copies of Python documentation in the `Doc` directory of the Python installation(s), either in CHM or HTML format.

If you are using Linux or OS X, the Python manual is not included in most Python installations, so you may wish to download and install local copies.

To do this, place the top-level of the [HTML formatted Python manual](#) (where `index.html` is found) into `python-manual/#.#` within your Wing IDE installation. Replace `#.#` with the major and minor version of the corresponding Python interpreter (for example, for the Python 2.7.x manual, use `python-manual/2.7`).

Once this is done, Wing will use the local disk copy rather than going to the web when the Python Manual item is selected from the Help menu.

## 1.13. Removing Wing IDE

### Windows

On Windows, use the Add/Remove Programs control panel, select **Wing IDE Personal 4** and remove it.

### Linux/Unix

To remove an RPM installation on Linux, type `rpm -e wingide-personal4.1`.

To remove an Debian package installation on Linux, type `dpkg -r wingide-personal4.1`.

To remove a tar archive installation on Linux/Unix, invoke the `wing-uninstall` script in the Wing IDE installation directory. This will automatically remove all files that appear not to have been changed since installation, It will ask whether it should remove any files that appear to be changed.

### Mac OS X

To remove Wing from Mac OS X, just drag its application folder to the trash.

## 1.14. Command Line Usage

Whenever you run `wing-personal4.1` from the command line, you may specify a list of files to open. These can be arbitrary text files and a project file. For example, the following will open project file `myproject.wpr` and also the three source files `mysource.py`, `README`, and `Makefile`:

```
wing-personal4.1 mysource.py README Makefile myproject.wpr
```

(on Windows, the executable is called `wing-personal.exe`)

Wing determines file type by extension, so position of the project file name (if any) on the command line is not important. A line number may be specified for the first file on the command line by appending `:<line-number>` to the file name (for example, `README:100` will position the cursor at the start of the `README` file).

The following valid options may be specified anywhere on the command line:

**--prefs-file** -- Add the file name following this argument to the list of preferences files that are opened by the IDE. These files are opened after the system-wide and default user preferences files, so values in them override those given in other preferences files.

**--new** -- By default Wing will reuse an existing running instance of Wing IDE to open files specified on the command line. This option turns off this behavior and forces creation of a new instance of Wing IDE. Note that a new instance is always created if no files are given on the command line.

**--reuse** -- Force Wing to reuse an existing running instance of Wing IDE even if there are no file names given on the command line. This just brings Wing to the front.

**--system-gtk** -- (*Posix only*) This option causes Wing to try to use the system-wide install of GTK2 rather than its own version of GTK, regardless of any preference setting. Running in this mode will cause Wing to pick up on system-wide theme defaults, but may result in crashing or display problems due to incompatibilities in GTK and related libraries.

**--private-gtk** -- (*Posix only*) This option causes Wing to use its private copy of GTK2 and related libraries, regardless of any preference settings. Use of private GTK may result in Wing not matching the system-wide theme, but also will avoid incompatibilities with the system-wide GTK library.

**--verbose** -- (*Posix only*) This option causes Wing to print verbose error reporting output to `stderr`. On Windows, run `console_wing.exe` instead for the same result.

**--display** -- (*Posix only*) Sets the X Windows display for Wing to run with. The display specification should follow this argument, in standard format, e.g. `myhost:0.0`.

**--use-sqlite-dotfile-locking** -- (*Posix only*) Use sqlite dotfile locking when opening databases which are located under the `~/.cache` directory or in the `$XDG_CACHE_DIR`.

**--use-winghome** -- (*For developers only*) This option sets `WINGHOME` to be used during this run. It is used internally and by developers contributing to Wing IDE. The directory to use follows this argument.

**--use-src** -- (*For developers only*) This option is used to force Wing to run from Python source files even if compiled files are present in the `bin` directory, as is the case after a distribution has been built.

**--orig-python-path** -- (*For developers only*) This option is used internally to indicate the original Python path in use by the user before Wing was launched. The path follows this argument.

**--squelch-output** -- (*For developers only*) This option prevents any output of any kind to `stdout` and `stderr`. Used on Windows to avoid console creation.





# Customization

There are many ways to customize Wing IDE in order to adapt it to your needs or preferences. This chapter describes the options that are available to you.

**These are some of the areas of customization that are available:**

- The editor can run with different personalities such as VI/Vim, Emacs, Visual Studio, Eclipse, and Brief emulation
- The action of the tab key can be configured
- The auto-completer's completion key(s) can be altered
- The layout, look, color, and content of the IDE windows can be configured
- Editor syntax colors can be configured
- Keyboard shortcuts can be added, removed, or altered for any Wing command
- File sets can be defined to control some of the IDE features
- Many other options are available through preferences

## 2.1. Keyboard Personalities

The default keyboard personality for Wing implements most common keyboard equivalents found in a simple graphical text editor. This uses primarily the graphical user interface for interacting with the editor and limits use of complex keyboard-driven command interaction.

### Emulation of Other Editors

The first thing any Vim, Emacs, Visual Studio, Eclipse, or Brief user will want to do is to set the keyboard personality to emulate their editor of choice. This is done with the Keyboard Personality item in the Edit menu or with the **Keyboard > Personality** preference.

Under the Vim and Emacs personalities, key strokes can be used to control most of the editor's functionality, using a textual interaction 'mini-buffer' at the bottom of the IDE window where the current line number and other informational messages are normally displayed.

Related preferences that alter keyboard behaviors include **Tab Key Action** and **Completion Keys** for the auto-completer.

It is also possible to add, alter, or remove individual keyboard command mappings within each of these personalities. See the following sub-sections for details.

#### 2.1.1. Key Equivalents

The command a key will invoke may be modified by specifying a custom key binding. A custom key binding will override any binding for a particular key found in the keymap. Custom key bindings are set via the **Custom Key Bindings** preference.

To add a binding, click the insert button, then press the key to be bound in the **Key** field, and enter the name of the command to invoke in the **Command** field. Commands are documented in the **Command Reference**.

Key bindings may consist of multiple key strokes in a row, such as **Ctrl-X Ctrl-U** or **Esc X Y Z**.

If multiple comma-separated commands are specified, the key binding will execute the first available command in the listed. For example, specifying **debug-restart, debug-continue** as the command will first try to restart an existing debug session, and if no debug session exists it will start a new one.

To disable a key binding, leave the command field blank.

Some commands take arguments, which can be specified in the binding, for example by using **show-panel(panel\_type="debug-probe")** or **enclose(start="(", end=")")** in the **Command** field. Any unspecified arguments that do not have a default defined by the command will be collected from the user, either in a dialog or in the data entry area at the bottom of the IDE window.

Key bindings defined by default or overridden by this preference will be shown in any

menu items that implement the same command. In cases where a command is given more than one key equivalent, only the last equivalent found will be displayed (although both bindings will work from the keyboard).

### 2.1.2. Key Maps

Wing ships with several key equivalency maps found at the top level of the Wing IDE installation, including `keymap.normal`, `keymap.emacs`, `keymap.vi`, and others. These are used as default key maps for the corresponding editor personalities, as set with the **Keyboard Personality** preference.

For developing entirely new key bindings, or in other cases where the **Custom Key Bindings** preference is not sufficient, it is possible to create a custom key equivalency map and use it as your default map through the **Key Map File** preference.

In a key map file, each key equivalent is built from names listed in the **Key Names** section. These names can be combined as follows:

- 1) A single unmodified key is specified by its name alone, for example `'Down'` for the down arrow key.
- 2) Modified keys are specified by hyphenating the key names, for example `'Shift-Down'` for the down arrow key pushed while shift is held down. Multiple modifiers may also be specified, as in `'Ctrl-Shift-Down'`.
- 3) Special modifiers are defined for Vim mode: **Visual**, **Browse**, **Insert**, and **Replace**. These correspond with the different editor modes, and will only work if the Keyboard Personality preference has been set to **VI/Vim**.
- 4) Multi-key combinations can be specified by listing multiple key names separated by a space. For example, to define a key equivalent that consists of first pushing **ctrl-x** and then pushing the **a** key by itself, use `'ctrl-x a'` as the key sequence.

The command portion of the key equivalency definition may be any of the commands listed in section **Command Reference**. See the examples below for usage options.

#### Examples

Here is an example of adding a key binding for a command. If the command already has a default key binding, both bindings will work:

```
'Ctrl-X P': 'debug-attach'
```

This example removes a key equivalent entirely:

```
'Ctrl-C Ctrl-C': None
```

These can be combined to changes the key binding for a command without retaining its default key binding:

```
'Ctrl-C Ctrl-C': None
'Ctrl-G': 'debug-continue'
```

Wing always retains only the last key binding for a given key combination. This example binds Ctrl-X to 'quit' and no other command:

```
'Ctrl-X': 'debug-stop'
'Ctrl-X': 'quit'
```

If multiple commands are specified separated by commas, Wing executes the first command that is available. For example, the following will either restart the debug process whether or not one is currently running:

```
'Ctrl-X': 'debug-restart, debug-continue'
```

Command arguments can be specified as part of the binding. Any unspecified arguments that do not have a default will be collected from the user in a dialog or in the data entry area at the bottom of the IDE window:

```
'Ctrl-X P': 'show-panel(panel_type="debug-probe")'
```

If Keyboard Personality is set to VI/Vim, modifiers corresponding to the editor modes restrict availability of the binding to only that mode:

```
'Visual-Ctrl-X': 'cut'
```

### 2.1.3. Key Names

**Key modifiers** supported by Wing IDE for key bindings are:

- **Ctrl** -- Either Control key.

- **Shift** -- Either Shift key. This modifier is ignored with some key names, as indicated below.
- **Alt** -- Either Alt key. Not recommended for general use since these bindings tend to conflict with menu accelerators and operating system or window manager operations.
- **Command** -- Macintosh Command/Apple key. This may be mapped to other keys on other systems, but is intended for use on the Macintosh.

On Linux and OS X it is possible to remap the function of the Control, Alt, command, and windows keys. In those cases, the Ctrl and Alt modifiers will refer to the keys specified in that mapping.

**Basic Keys** such as the digit keys and core western alphabet keys are specified as follows:

0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

**Most punctuation** can be specified but any Shift modifier will be ignored since these keys can vary in location on different international keyboards. Allowed punctuation includes:

‘ ~ ! @ # \$ % ^ & \* ( ) - \_ + = [ ] { } \ | ; : ’ “ / ? . > , <

**Special Keys** can also be used:

Escape, Space, BackSpace, Tab, Linefeed, Clear, Return, Pause, Scroll\_Lock, Sys\_Req, Delete, Home, Left, Up, Right, Down, Prior, Page\_Up, Next, Page\_Down, End, Begin, Select, Print, Execute, Insert, Undo, Redo, Menu, Find, Cancel, Help, Break, Mode\_switch, script\_switch, Num\_Lock,

F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, L1, F12, L2, F13, L3, F14, L4, F15, L5, F16, L6, F17, L7, F18, L8, F19, L9, F20, L10, F21, R1, F22, R2, F23, R3, F24, R4, F25, R5, F26, R6, F27, R7, F28, R8, F29, R9, F30, R10, F31, R11, F32, R12, F33, R13, F34, R14, F35, R15,

**Mouse Buttons** are also named for key bindings:

Pointer\_Left, Pointer\_Right, Pointer\_Up, Pointer\_Down, Pointer\_UpLeft, Pointer\_UpRight, Pointer\_DownLeft, Pointer\_DownRight, Pointer\_Button\_Dflt, Pointer\_Button1, Pointer\_Button2, Pointer\_Button3, Pointer\_Button4, Pointer\_Button5, Pointer\_DblClick\_Dflt, Pointer\_DblClick1, Pointer\_DblClick2, Pointer\_DblClick3, Pointer\_DblClick4, Pointer\_DblClick5, Pointer\_Drag\_Dflt, Pointer\_Drag1, Pointer\_Drag2, Pointer\_Drag3, Pointer\_Drag4, Pointer\_EnableKeys, Pointer\_Accelerate, Pointer\_DfltBtnNext, Pointer\_DfltBtnPrev,

**Keypad Keys** are specified like this:

KP\_Left, KP\_Right, KP\_Up, KP\_Down, KP\_Home, KP\_Page\_Up, KP\_Page\_Down, KP\_End, KP\_Insert, KP\_Delete, KP\_0, KP\_1, KP\_2, KP\_3, KP\_4, KP\_5, KP\_6, KP\_7, KP\_8, KP\_9,

**Additional Key Names** that also work but ignore the Shift modifier since they tend to appear in different locations on international keyboards:

KP\_Space, KP\_Tab, KP\_Enter, KP\_F1, KP\_F2, KP\_F3, KP\_F4, KP\_Prior, KP\_Next, KP\_Begin, KP\_Insert, KP\_Delete, KP\_Equal, KP\_Multiply, KP\_Add, KP\_Separator, KP\_Subtract, KP\_Decimal, KP\_Divide,

exclam, quotedbl, numbersign, dollar, percent, ampersand, apostrophe, quoteright, parenleft, parenright, asterisk, plus, comma, minus, period, slash, colon, semicolon, less, equal, greater, question, at, bracketleft, backslash, bracketright, asciicircum, underscore, grave, quoteleft, braceleft, bar, braceright,

EuroSign, EcuSign, ColonSign, CruzeiroSign, FFrancSign, LiraSign, MillSign, NairaSign, PesetaSign, RupeeSign, WonSign, NewSheqelSign, DongSign,

**Special Character Names** are available for use with key bindings on international or special purpose keyboards:

asciitilde, nobreakspace, exclamdown, cent, sterling, currency, yen, brokenbar, section, diaeresis, copyright, ordfeminine, guillemotleft, notsign, hyphen, registered, macron, degree, plusminus, twosuperior, threesuperior, acute, mu, paragraph, periodcentered, cedilla, onesuperior, masculine, guillemotright, onequarter, onehalf, threequarters, questiondown,

leftradical, topleftradical, horizconnector, topintegral, botintegral, vertconnector, topleftsqbracket, botleftsqbracket, toprightsqbracket, botrightsqbracket, topleftparens, botleftparens, toprightparens, botrightparens, leftmiddlecurlybrace, rightmiddlecurlybrace, topleftsummation, botleftsummation, topvertsummationconnector, botvertsummationconnector, toprightsummation, botrightsummation, rightmiddlesummation, lessthanequal, notequal, greaterthanequal, integral, therefore, variation, infinity, nabla, approximate, similarequal, ifonlyif, implies, identical, radical, includedin, includes, intersection, union, logicaland, logicalor, partialderivative, function, leftarrow, uparrow, rightrightarrow, downarrow, blank, soliddiamond, checkerboard, ht, ff, cr, lf, nl, vt, lowrightcorner, uprightcorner, upleftcorner, lowleftcorner, crossinglines, horizlinescan1, horizlinescan3, horizlinescan5, horizlinescan7, horizlinescan9, leftt, rightt, bott, topt, vertbar, emspace, enspace, em3space, em4space, digitsspace, punctspace, thinspace, hairspace, emdash, endash, signifblank, ellipsis, doubbaselinedot, onethird, twothirds, onefifth, twofifths, threefifths, fourfifths, onesixth, fivesixths, careof, figdash, leftanglebracket, decimalpoint, rightanglebracket, marker, oneeighth, threeeighths, fiveeighths, seveeneighths, trademark, signaturemark, trademarkincircle, leftopentriangle, rightopen-

triangle, emopencircle, emopenrectangle, leftsinglequotemark, rightsinglequotemark, leftdoublequotemark, rightrightdoublequotemark, prescription, minutes, seconds, latincross, hexagram, filledrectbullet, filledlefttribullet, filledrighttribullet, emfilledcircle, emfilledrect, emopencircbullet, emopensquarebullet, openrectbullet, opentribulletup, opentribulletdown, openstar, emfilledcircbullet, emfilledsqbullet, filledtribulletup, filledtribulletdown, leftpointer, rightpointer, club, diamond, heart, maltesecross, dagger, doubledagger, checkmark, ballotcross, musicalsharp, musicalflat, malesymbol, femalesymbol, telephone, telephonerecorder, phonographcopyright, caret, singlelowquotemark, doublelowquotemark, cursor, leftcaret, rightcaret, downcaret, upcaret, overbar, downtack, upshoe, downstile, underbar, jot, quad, uptack, circle, upstile, downshoe, rightshoe, leftshoe, lefttack, righttack,

Multi\_key, Codeinput, SingleCandidate, MultipleCandidate, PreviousCandidate, Kanji, Muhenkan, Henkan\_Mode, Henkan, Romaji, Hiragana, Katakana, Hiragana\_Katakana, Zenkaku, Hankaku, Zenkaku\_Hankaku, Touroku, Massyo, Kana\_Lock, Kana\_Shift, Eisu\_Shift, Eisu\_toggle, Kanji\_Bangou, Zen\_Koho, Mae\_Koho,

ISO\_Lock, ISO\_Level2\_Latch, ISO\_Level3\_Shift, ISO\_Level3\_Latch, ISO\_Level3\_Lock, ISO\_Group\_Shift, ISO\_Group\_Latch, ISO\_Group\_Lock, ISO\_Next\_Group, ISO\_Next\_Group\_Lock, ISO\_Prev\_Group, ISO\_Prev\_Group\_Lock, ISO\_First\_Group, ISO\_First\_Group\_Lock, ISO\_Last\_Group, ISO\_Last\_Group\_Lock, ISO\_Left\_Tab, ISO\_Move\_Line\_Up, ISO\_Move\_Line\_Down, ISO\_Partial\_Line\_Up, ISO\_Partial\_Line\_Down, ISO\_Partial\_Space\_Left, ISO\_Partial\_Space\_Right, ISO\_Set\_Margin\_Left, ISO\_Set\_Margin\_Right, ISO\_Release\_Margin\_Left, ISO\_Release\_Margin\_Right, ISO\_Release\_Both\_Margins, ISO\_Fast\_Cursor\_Left, ISO\_Fast\_Cursor\_Right, ISO\_Fast\_Cursor\_Up, ISO\_Fast\_Cursor\_Down, ISO\_Continuous\_Underline, ISO\_Discontinuous\_Underline, ISO\_Emphasize, ISO\_Center\_Object, ISO\_Enter

dead\_grave, dead\_acute, dead\_circumflex, dead\_tilde, dead\_macron, dead\_breve, dead\_abovedot, dead\_diaeresis, dead\_abovering, dead\_doubleacute, dead\_caron, dead\_cedilla, dead\_ogonek, dead\_iota, dead\_voiced\_sound, dead\_semivoiced\_sound, dead\_belowdot,

First\_Virtual\_Screen, Prev\_Virtual\_Screen, Next\_Virtual\_Screen, Last\_Virtual\_Screen, Terminate\_Server, AccessX\_Enable, AccessX\_Feedback\_Enable, RepeatKeys\_Enable, SlowKeys\_Enable, BounceKeys\_Enable, StickyKeys\_Enable, MouseKeys\_Enable, MouseKeys\_Accel\_Enable, Overlay1\_Enable, Overlay2\_Enable, AudibleBell\_Enable, Pointer\_Left, Pointer\_Right, Pointer\_Up,

\_3270\_Duplicate, \_3270\_FieldMark, \_3270\_Right2, 3270\_Left2, \_3270\_BackTab, \_3270\_EraseEOF, \_3270\_EraseInput, 3270\_Reset, \_3270\_Quit, \_3270\_PA1, \_3270\_PA2, \_3270\_PA3, \_3270\_Test, 3270\_Attn, \_3270\_CursorBlink, \_3270\_AltCursor, \_3270\_KeyClick, 3270\_Jump, \_3270\_Ident, \_3270\_Rule, \_3270\_Copy, \_3270\_Play, \_3270\_Setup, 3270\_Record, \_3270\_ChangeScreen, \_3270\_DeleteWord, \_3270\_ExSelect, 3270\_CursorSelect, \_3270\_PrintScreen, \_3270\_Enter,

Agrave, Aacute, Acircumflex, Atilde, Adiaeresis, Aring, AE, Ccedilla, Egrave, Eacute, Ecircumflex, Ediaeresis, Igrave, Iacute, Icircumflex, Idiaeresis, ETH, Eth, Ntilde, Ograve, Oacute, Ocircumflex, Otilde, Odiaeresis, multiply, Ooblique, Ugrave, Uacute, Ucircumflex, Udiaeresis, Yacute, THORN, Thorn, ssharp, agrave, aacute, acircumflex, atilde, adiaeresis, aring, ae, ccedilla, egrave, eacute, ecircumflex, ediaeresis, igrave, iacute, icircumflex, idiaeresis, eth, ntilde, ograve, oacute, ocircumflex, otilde, odiaeresis, division, oslash, ugrave, uacute, ucircumflex, udiaeresis, yacute, thorn, ydiaeresis, Aogonek, breve, Lstroke, Lcaron, Sacute, Scaron, Scedilla, Tcaron, Zacute, Zcaron, Zabove-dot, aogonek, ogonek, lstroke, lcaron, sacute, caron, scaron, scedilla, tcaron, zacute, doubleacute, zcaron, zabove-dot, Racute, Abreve, Lacute, Cacute, Ccaron, Eogonek, Ecaron, Dcaron, Dstroke, Nacute, Ncaron, Odoubleacute, Rcaron, Uring, Udoubleacute, Tcedilla, racute, abreve, lacute, cacute, ccaron, eogonek, ecaron, dcaron, dstroke, nacute, ncaron, odoubleacute, udoubleacute, rcaron, uring, tcedilla, abovedot, Hstroke, Hcircumflex, labovedot, Gbreve, Jcircumflex, hstroke, hcircumflex, idotless, gbreve, jcircumflex, Cabovedot, Ccircumflex, Gabovedot, Gcircumflex, Ubreve, Scircumflex, cabovedot, ccircumflex, gabovedot, gcircumflex, ubreve, scircumflex, kra, kappa, Rcedilla, Itilde, Lcedilla, Emacron, Gcedilla, Tslash, rcedilla, itilde, lcedilla, emacron, gcedilla, tslash, ENG, eng, Amacron, Iogonek, Eabovedot, Imacron, Ncedilla, Omacron, Kcedilla, Uogonek, Utilde, Umacron, amacron, iogonek, eabovedot, imacron, ncedilla, omacron, kcedilla, uogonek, utilde, umacron, OE, oe, Ydiaeresis, overline,

kana\_fullstop, kana\_openingbracket, kana\_closingbracket, kana\_comma, kana\_conjunctive, kana\_middledot, kana\_WO, kana\_a, kana\_i, kana\_u, kana\_e, kana\_o, kana\_ya, kana\_yu, kana\_yo, kana\_tsu, kana\_tu, prolongedsound, kana\_A, kana\_I, kana\_U, kana\_E, kana\_O, kana\_KA, kana\_KI, kana\_KU, kana\_KE, kana\_KO, kana\_SA, kana\_SHI, kana\_SU, kana\_SE, kana\_SO, kana\_TA, kana\_CHI, kana\_TI, kana\_TSU, kana\_TU, kana\_TE, kana\_TO, kana\_NA, kana\_NI, kana\_NU, kana\_NE, kana\_NO, kana\_HA, kana\_HI, kana\_FU, kana\_HU, kana\_HE, kana\_HO, kana\_MA, kana\_MI, kana\_MU, kana\_ME, kana\_MO, kana\_YA, kana\_YU, kana\_YO, kana\_RA, kana\_RI, kana\_RU, kana\_RE, kana\_RO, kana\_WA, kana\_N, voicedsound, semivoicedsound, kana\_switch,

Arabic\_comma, Arabic\_semicolon, Arabic\_question\_mark, Arabic\_hamza, Arabic\_maddaonalef, Arabic\_hamzaonalef, Arabic\_hamzaonwaw, Arabic\_hamzaunderalef, Arabic\_hamzaonyeh, Arabic\_alef, Arabic\_beh, Arabic\_tehmarbuta, Arabic\_teh, Arabic\_theh, Arabic\_jeem, Arabic\_hah, Arabic\_khah, Arabic\_dal, Arabic\_thal, Arabic\_ra, Arabic\_zain, Arabic\_seen, Arabic\_sheen, Arabic\_sad, Arabic\_dad, Arabic\_tah, Arabic\_zah, Arabic\_ain, Arabic\_ghain, Arabic\_tatweel, Arabic\_feh, Arabic\_qaf, Arabic\_kaf, Arabic\_lam, Arabic\_meem, Arabic\_noon, Arabic\_ha, Arabic\_heh, Arabic\_waw, Arabic\_alefmaksura, Arabic\_yeh, Arabic\_fathatan, Arabic\_dammatan, Arabic\_kasratan, Arabic\_fatha, Arabic\_damma, Arabic\_kasra, Arabic\_shadda, Arabic\_sukun, Arabic\_switch,

Serbian\_dje, Macedonia\_gje, Cyrillic\_io, Ukrainian\_ie, Ukranian\_je, Macedonia\_dse, Ukrainian\_i, Ukranian\_i, Ukrainian\_yi, Ukranian\_yi, Cyrillic\_je, Serbian\_je,



Cyrillic\_lje, Serbian\_lje, Cyrillic\_nje, Serbian\_nje, Serbian\_tshe, Macedonia\_kje, Ukrainian\_ghe\_with\_upturn, Byelorussian\_shortu, Cyrillic\_dzhe, Serbian\_dze, numerosign, Serbian\_DJE, Macedonia\_GJE, Cyrillic\_IO, Ukrainian\_IE, Ukrainian\_JE, Macedonia\_DSE, Ukrainian\_I, Ukrainian\_I, Ukrainian\_YI, Ukrainian\_YI, Cyrillic\_JE, Serbian\_JE, Cyrillic\_LJE, Serbian\_LJE, Cyrillic\_NJE, Serbian\_NJE, Serbian\_TSHE, Macedonia\_KJE, Ukrainian\_GHE\_WITH\_UPTURN, Byelorussian\_SHORTU, Cyrillic\_DZHE, Serbian\_DZE, Cyrillic\_yu, Cyrillic\_a, Cyrillic\_be, Cyrillic\_tse, Cyrillic\_de, Cyrillic\_ie, Cyrillic\_ef, Cyrillic\_ghe, Cyrillic\_ha, Cyrillic\_i, Cyrillic\_shorti, Cyrillic\_ka, Cyrillic\_el, Cyrillic\_em, Cyrillic\_en, Cyrillic\_o, Cyrillic\_pe, Cyrillic\_ya, Cyrillic\_er, Cyrillic\_es, Cyrillic\_te, Cyrillic\_u, Cyrillic\_zhe, Cyrillic\_ve, Cyrillic\_softsign, Cyrillic\_yeru, Cyrillic\_ze, Cyrillic\_sha, Cyrillic\_e, Cyrillic\_shcha, Cyrillic\_che, Cyrillic\_hardsign, Cyrillic\_YU, Cyrillic\_A, Cyrillic\_BE, Cyrillic\_TSE, Cyrillic\_DE, Cyrillic\_IE, Cyrillic\_EF, Cyrillic\_GHE, Cyrillic\_HA, Cyrillic\_I, Cyrillic\_SHORTI, Cyrillic\_KA, Cyrillic\_EL, Cyrillic\_EM, Cyrillic\_EN, Cyrillic\_O, Cyrillic\_PE, Cyrillic\_YA, Cyrillic\_ER, Cyrillic\_ES, Cyrillic\_TE, Cyrillic\_U, Cyrillic\_ZHE, Cyrillic\_VE, Cyrillic\_SOFTSIGN, Cyrillic\_YERU, Cyrillic\_ZE, Cyrillic\_SHA, Cyrillic\_E, Cyrillic\_SHCHA, Cyrillic\_CHE, Cyrillic\_HARDSIGN,

Greek\_ALPHAaccent, Greek\_EPSILONaccent, Greek\_ETAaccent, Greek\_IOTAaccent, Greek\_IOTAadiaeresis, Greek\_OMICRONaccent, Greek\_UPSILONaccent, Greek\_UPSILONdieresis, Greek\_OMEGAaccent, Greek\_accentdieresis, Greek\_horizbar, Greek\_alphaaccent, Greek\_epsilonaccent, Greek\_etaaccent, Greek\_iotaaccent, Greek\_iotadieresis, Greek\_iotaaccentdieresis, Greek\_omicronaccent, Greek\_upsilonaccent, Greek\_upsilondieresis, Greek\_upsilonaccentdieresis, Greek\_omegaaccent, Greek\_ALPHA, Greek\_BETA, Greek\_GAMMA, Greek\_DELTA, Greek\_EPSILON, Greek\_ZETA, Greek\_ETA, Greek\_THETA, Greek\_IOTA, Greek\_KAPPA, Greek\_LAMDA, Greek\_LAMBDA, Greek\_MU, Greek\_NU, Greek\_XI, Greek\_OMICRON, Greek\_PI, Greek\_RHO, Greek\_SIGMA, Greek\_TAU, Greek\_UPSILON, Greek\_PHI, Greek\_CHI, Greek\_PSI, Greek\_OMEGA, Greek\_alpha, Greek\_beta, Greek\_gamma, Greek\_delta, Greek\_epsilon, Greek\_zeta, Greek\_eta, Greek\_theta, Greek\_iota, Greek\_kappa, Greek\_lamda, Greek\_lambda, Greek\_mu, Greek\_nu, Greek\_xi, Greek\_omicron, Greek\_pi, Greek\_rho, Greek\_sigma, Greek\_finalsmallsigma, Greek\_tau, Greek\_upsilon, Greek\_phi, Greek\_chi, Greek\_psi, Greek\_omega, Greek\_switch,

hebrew\_doublelowline, hebrew\_aleph, hebrew\_bet, hebrew\_beth, hebrew\_gimel, hebrew\_gimmel, hebrew\_dalet, hebrew\_daleth, hebrew\_he, hebrew\_waw, hebrew\_zain, hebrew\_zayin, hebrew\_chet, hebrew\_het, hebrew\_tet, hebrew\_teth, hebrew\_yod, hebrew\_finalkaph, hebrew\_kaph, hebrew\_lamed, hebrew\_finalmem, hebrew\_mem, hebrew\_finalnun, hebrew\_nun, hebrew\_samech, hebrew\_samekh, hebrew\_ayin, hebrew\_finalpe, hebrew\_pe, hebrew\_finalzade, hebrew\_finalzadi, hebrew\_zade, hebrew\_zadi, hebrew\_qoph, hebrew\_kuf, hebrew\_resh, hebrew\_shin, hebrew\_taw, hebrew\_taf, Hebrew\_switch,

Thai\_kokai, Thai\_khokhai, Thai\_khokhuat, Thai\_khokhwai, Thai\_khokhon,  
 Thai\_khorakhang, Thai\_ngongu, Thai\_chochan, Thai\_choching, Thai\_chochang,  
 Thai\_soso, Thai\_chochoe, Thai\_yoying, Thai\_dochada, Thai\_topatak, Thai\_thothan,  
 Thai\_thonangmontho, Thai\_thophuthao, Thai\_nonen, Thai\_dodek, Thai\_totao,  
 Thai\_thothung, Thai\_thothahan, Thai\_thothong, Thai\_nonu, Thai\_bobaimai,  
 Thai\_popla, Thai\_phophung, Thai\_fofa, Thai\_phophan, Thai\_fofan, Thai\_phosamphao,  
 Thai\_moma, Thai\_yoyak, Thai\_rorua, Thai\_ru, Thai\_loling, Thai\_lu,  
 Thai\_wowaen, Thai\_sosala, Thai\_sorusi, Thai\_sosua, Thai\_hohip, Thai\_lochula,  
 Thai\_oang, Thai\_honokhuk, Thai\_paiyannoi, Thai\_saraa, Thai\_maihanakat,  
 Thai\_saraaa, Thai\_saraam, Thai\_sarai, Thai\_saraii, Thai\_saraue, Thai\_sarauee,  
 Thai\_sarau, Thai\_sarauu, Thai\_phinthu, Thai\_maihanakat\_maitho, Thai\_baht,  
 Thai\_sarae, Thai\_saraae, Thai\_sarao, Thai\_saraaimaimuan, Thai\_saraaimaimalai,  
 Thai\_lakkhangyao, Thai\_maiyamok, Thai\_maitaikhu, Thai\_maiek, Thai\_maitho,  
 Thai\_maitri, Thai\_maichattawa, Thai\_thanthakhat, Thai\_nikhahit, Thai\_leksun,  
 Thai\_lekngung, Thai\_leksong, Thai\_leksam, Thai\_leksi, Thai\_lekha, Thai\_lekhok,  
 Thai\_lekchet, Thai\_lekpaet, Thai\_lekkao,

Hangul, Hangul\_Start, Hangul\_End, Hangul\_Hanja, Hangul\_Jamo, Hangul\_Romaja,  
 Hangul\_Codeinput, Hangul\_Jeonja, Hangul\_Banja, Hangul\_PreHanja,  
 Hangul\_PostHanja, Hangul\_SingleCandidate, Hangul\_MultipleCandidate,  
 Hangul\_PreviousCandidate, Hangul\_Special, Hangul\_Switch, Hangul\_Kiyeog,  
 Hangul\_SsangKiyeog, Hangul\_KiyeogSios, Hangul\_Nieun, Hangul\_NieunJieuj,  
 Hangul\_NieunHieuh, Hangul\_Dikeud, Hangul\_SsangDikeud, Hangul\_Rieul,  
 Hangul\_RieulKiyeog, Hangul\_RieulMieum, Hangul\_RieulPieub, Hangul\_RieulSios,  
 Hangul\_RieulTieut, Hangul\_RieulPhieuf, Hangul\_RieulHieuh, Hangul\_Mieum,  
 Hangul\_Pieub, Hangul\_SsangPieub, Hangul\_PieubSios, Hangul\_Sios, Hangul\_SsangSios,  
 Hangul\_Ieung, Hangul\_Jieuj, Hangul\_SsangJieuj, Hangul\_Cieuc, Hangul\_Khieuh,  
 Hangul\_Tieut, Hangul\_Phieuf, Hangul\_Hieuh, Hangul\_A, Hangul\_AE,  
 Hangul\_YA, Hangul\_YAE, Hangul\_EO, Hangul\_E, Hangul\_YEO, Hangul\_YE,  
 Hangul\_O, Hangul\_WA, Hangul\_WAE, Hangul\_OE, Hangul\_YO, Hangul\_U,  
 Hangul\_WEO, Hangul\_WE, Hangul\_WI, Hangul\_YU, Hangul\_EU, Hangul\_YI,  
 Hangul\_I, Hangul\_J\_Kiyeog, Hangul\_J\_SsangKiyeog, Hangul\_J\_KiyeogSios,  
 Hangul\_J\_Nieun, Hangul\_J\_NieunJieuj, Hangul\_J\_NieunHieuh, Hangul\_J\_Dikeud,  
 Hangul\_J\_Rieul, Hangul\_J\_RieulKiyeog, Hangul\_J\_RieulMieum, Hangul\_J\_RieulPieub,  
 Hangul\_J\_RieulSios, Hangul\_J\_RieulTieut, Hangul\_J\_RieulPhieuf, Hangul\_J\_RieulHieuh,  
 Hangul\_J\_Mieum, Hangul\_J\_Pieub, Hangul\_J\_PieubSios, Hangul\_J\_Sios,  
 Hangul\_J\_SsangSios, Hangul\_J\_Ieung, Hangul\_J\_Jieuj, Hangul\_J\_Cieuc,  
 Hangul\_J\_Khieuh, Hangul\_J\_Tieut, Hangul\_J\_Phieuf, Hangul\_J\_Hieuh,  
 Hangul\_RieulYeorinHieuh, Hangul\_SunkyeongeumMieum, Hangul\_SunkyeongeumPieub,  
 Hangul\_PanSios, Hangul\_KkogjiDalrinIeung, Hangul\_SunkyeongeumPhieuf,  
 Hangul\_YeorinHieuh, Hangul\_AraeA, Hangul\_AraeAE, Hangul\_J\_PanSios,  
 Hangul\_J\_KkogjiDalrinIeung, Hangul\_J\_YeorinHieuh, Korean\_Won,

Armenian\_eternity, Armenian\_section\_sign, Armenian\_full\_stop, Armenian\_verjaket, Armenian\_parenright, Armenian\_parenleft, Armenian\_guillemotright, Armenian\_guillemotleft, Armenian\_em\_dash, Armenian\_dot, Armenian\_mijaket, Armenian\_separation\_mark, Armenian\_but, Armenian\_comma, Armenian\_en\_dash, Armenian\_hyphen, Armenian\_yentamna, Armenian\_ellipsis, Armenian\_exclam, Armenian\_amanak, Armenian\_accent, Armenian\_shesht, Armenian\_question, Armenian\_paruyk, Armenian\_AYB, Armenian\_ayb, Armenian\_BEN, Armenian\_ben, Armenian\_GIM, Armenian\_gim, Armenian\_DA, Armenian\_da, Armenian\_YECH, Armenian\_yech, Armenian\_ZA, Armenian\_za, Armenian\_E, Armenian\_e, Armenian\_AT, Armenian\_at, Armenian\_TO, Armenian\_to, Armenian\_ZHE, Armenian\_zhe, Armenian\_INI, Armenian\_ini, Armenian\_LYUN, Armenian\_lyun, Armenian\_KHE, Armenian\_khe, Armenian\_TSA, Armenian\_tsa, Armenian\_KEN, Armenian\_ken, Armenian\_HO, Armenian\_ho, Armenian\_DZA, Armenian\_dza, Armenian\_GHAT, Armenian\_ghat, Armenian\_TCHE, Armenian\_tche, Armenian\_MEN, Armenian\_men, Armenian\_HI, Armenian\_hi, Armenian\_NU, Armenian\_nu, Armenian\_SHA, Armenian\_sha, Armenian\_VO, Armenian\_vo, Armenian\_CHA, Armenian\_cha, Armenian\_PE, Armenian\_pe, Armenian\_JE, Armenian\_je, Armenian\_RA, Armenian\_ra, Armenian\_SE, Armenian\_se, Armenian\_VEV, Armenian\_vev, Armenian\_TYUN, Armenian\_tyun, Armenian\_RE, Armenian\_re, Armenian\_TSO, Armenian\_tso, Armenian\_VYUN, Armenian\_vyun, Armenian\_PYUR, Armenian\_pyur, Armenian\_KE, Armenian\_ke, Armenian\_O, Armenian\_o, Armenian\_FE, Armenian\_fe, Armenian\_apostrophe, Armenian\_ligature\_ew,

Georgian\_an, Georgian\_ban, Georgian\_gan, Georgian\_don, Georgian\_en, Georgian\_vin, Georgian\_zen, Georgian\_tan, Georgian\_in, Georgian\_kan, Georgian\_las, Georgian\_man, Georgian\_nar, Georgian\_on, Georgian\_par, Georgian\_zhar, Georgian\_rae, Georgian\_san, Georgian\_tar, Georgian\_un, Georgian\_phar, Georgian\_khar, Georgian\_ghan, Georgian\_qar, Georgian\_shin, Georgian\_chin, Georgian\_can, Georgian\_jil, Georgian\_cil, Georgian\_char, Georgian\_xan, Georgian\_jhan, Georgian\_hae, Georgian\_he, Georgian\_hie, Georgian\_we, Georgian\_har, Georgian\_hoe, Georgian\_fi,

## 2.2. User Interface Options

Wing provides many options for customizing the user interface to your needs. Preferences can be set to control the number and type of windows, layout of tools and editors, text fonts and colors, type of toolbar, and the overall display style or "theme" (including white on black and many others).

### 2.2.1. Windowing Policies

Wing IDE can run in a variety of windowing modes. This is controlled by the **Windowing Policy** preference, which provides the following options:

- **Combined Tool Box and Editor Windows** -- This is the default, in which Wing opens a single window that combines the editor area with two tool box panels.
- **Separate Tool Box Windows** -- In this mode, Wing IDE moves all the tools out to a separate shared window.
- **One Window Per Editor** -- In this mode, Wing IDE creates one top-level window for each editor that is opened. Additionally, all tools are moved out to a separate shared tool box window and the toolbar and menu are moved out to a shared toolbar/menu window.

The windowing policy is used to describe the initial configuration and basic action of windows in the IDE. When it is changed, Wing will reconfigure your projects to match the windowing policy the first time they are used with the new setting.

However, it is possible to create additional IDE windows and to move editors and tools out to another window or among existing windows without changing from the default windowing policy. This is described below.

### 2.2.2. User Interface Layout

When working in the default windowing policy, Wing's main user interface area consists of two tool boxes (by default at bottom and right) and an area for source editors and integrated help.

Clicking on an already-active notebook tab will cause Wing to minimize the entire panel so that only the notebook tabs are visible. Clicking again will return the tool box to its former size. The F1 and F2 keys toggle between these modes. The command **Maximize Editor Area** in the **Tools** menu (Shift-F2) can also be used to quickly hide both tool areas and toolbar.

In other windowing modes, the tool boxes and editor area are presented in separate windows but share many of the configuration options described below.

#### Configuring the Toolbar

Wing's toolbar can be configured by altering the size and style of the toolbar icons in

the toolbar, and whether or not text is shown in addition to or instead of icons. This is controlled with the **Toolbar Icon Size** and **Toolbar Icon Style** preferences.

Alternatively, the toolbar can be hidden completely with the **Show Toolbar** preference.

## Configuring the Editor Area

The options drop down menu in the top right of the editor area allows for splitting and joining the editor into multiple independent panels. These can be arranged horizontally, vertically, or any combination thereof. When multiple splits are shown, all the open files within the window are available within each split, allowing work on any combination of files and/or different parts of the same file.

The options drop down menu can also be used to change between tabbed editors and editors that show a popup menu for selecting among files (the latter can be easier to manage with large number of files) and to move editors out to a separate window or among existing windows when multiple windows are open.

## Configuring Tool Boxes

Each of the tool boxes can be also be split or joined into any number of sub-panels along the long axis of the notebook by clicking on the options drop down icon in the tab area of the notebooks (right-clicking also works). The number of tool box splits Wing shows by default depends on your monitor size.

The options drop down menu can also be used to duplicate tools, or move them around among the splits or out to separate windows.

The size of each panel and the panel splits can also be altered by dragging on the dividers between them.

All available tools are enumerated in the Tools menu, which will display the most recently used tool of that type or will add one to your window at its default location, if none is already present.

## Creating Additional Windows

In addition to moving existing editors or tools to new windows, it is also possible to create new tool windows (initially with a single tool) and new document windows (with editor and toolbars if applicable to the selected windowing policy) from the Windows menu.

Wing IDE will remember the state of windows as part of your project file, so the same window layout and contents will be restored in subsequent work sessions.

### 2.2.3. Altering Text Display

Wing tries to find display fonts appropriate for each system on which it runs, but many users will want to customize the font style and size used in the editor and other user interface areas. This can be done with the **Source Code Font/Size** and **Display Font/Size** preferences.

The color of text for some file types in the editor can be controlled with the **Syntax Formatting** preference.

Note that when the **Source Code Background** preference is set to a color other than white, Wing will compute appropriately visible colors for text according to the chosen background color.

The color used for text selection can also be controlled with the **Text Selection Color** preference.

Changes in color preferences will often depend on the overall display theme that is chosen, as described in the next section.

### 2.2.4. Setting Overall Display Theme

Wing is based on GTK2, a cross-platform user interface toolkit that provides customizable **themes**, which control the overall look and feel of the user interface. Wing's default theme varies by platform (a Windows emulation theme is used on Windows, and an OS X like theme on OS X) and can be changed with the **Display Theme** preference.

In most cases, the new theme will instantly be applied to Wing's user interface. When switching back to default settings, a restart may be needed in some cases, as indicated by message dialog.

Some systems with slower graphics cards may not run as well using the more colorful 3D rendered themes. In this case, using the **Gtk Default** theme is the best option, as it involves no extra graphics-level processing.

#### System GTK on Linux

On Linux systems with GTK 2.6 or later installed, it is possible to run Wing with the system-wide GTK installation and system-defined themes. This is controlled with the **Use System GTK** preference or the **--system-gtk** or **--private-gtk command line arguments**. Wing works reasonably well with most 2.6.x GTK2 releases, but there still may be problems resulting from version differences. If you have any problems with Wing's stability or are seeing display glitches, you should use the private gtk option.

## 2.3. Preferences

Wing has many preferences that control features of the editor, debugger, and other tools.

To alter these, use the **Preferences** item in the **Edit** menu. This organizes all available preferences by category and provides access to documentation in tooltips that are displayed when mousing over the label area to the left of each preference. Any non-default values that are selected through the **Preferences Dialog** are stored in the user's preferences file, which is located in the **User Settings Directory**.

All preferences are documented in the **Preferences Reference**.

## 2.4. File Filters

Wing provides a way to define sets of files that can be used in various ways within the IDE, such as for searching particular batches of files and adding only certain kinds of files to a project.

To view or alter the defined file sets, use the **File Sets...** item in the File menu. This will display a file set editor within the Preferences manager.

When adding or editing a file set, the following information may be entered:

- **Name** -- The name of the file set
- **Includes** -- A list of inclusion criteria, each of which contains a type and a specification. A file will be included in the file set if any one of these include criteria matches it.
- **Excludes** -- A list of exclusion criteria, any of which can match to cause a file to be excluded from the file set even if one or more include matches were also found.

The following types of include and exclude criteria are supported:

- **Wildcard on Filename** -- The specification in this case is a wildcard that must match the file name. The wildcards supported are those provided by Python's [fnmatch](#) module.
- **Wildcard on Directory Name** -- The specification in this case is a wildcard that must match the directory name.
- **Mime Type** -- The specification in this case names a MIME type supported by Wing IDE. If additional file extensions need to be mapped to a MIME type, use the **Extra File Types** preference to define them.

Once defined, file sets are presented by name in the **Search in Files** tool's batch search facility and in the **Project tool**'s batch file addition features.

Any problems encountered in using the file sets are reported in the Messages area.



# Project Manager

The **Project manager** provides a convenient index of the files in your software project and collects information needed by Wing's debugger, source code analysis tools, version control integration, and other facilities.

To get the most out of Wing's debugger and source analysis engine, you may in some cases need to set up **Python Executable**, **Python Path**, and other values in **Project-Wide Properties** and/or **Per-File Properties**.

NOTE: Wing IDE Personal omits some of the project manager features.

## 3.1. Creating a Project

To create a new project, use the **New Project** item in the **Project** menu. This will prompt you to save any changes to your currently open project and will create a new untitled project.

When you create a new project, you will often want to alter some of the **Project Properties** to point Wing at the version of Python you want to use, set `PYTHONPATH` so Wing's source analyzer and debugger can find your files, and set any other necessary runtime environment for your code.

To add files to your project, use the following items in the **Project** menu:

- **Add Existing Directory** allows you to specify a directory to include in the project. In many cases, this is the only operation needed to set up a new project, and it is the recommended approach. You will be able to specify a filter of which files to include, whether to include hidden & temporary files, and whether to include subdirectories. The list of files in the project will be updated as files matching the criteria are added and removed from the disk.
- **Add Current File** will add the current editor file to the project if it is not already there.

- **Add Existing File** will prompt you to select a single file to add to the project view. This may also result in adding a new directory to the project manager window, if that file is the first to be added for a directory.
- **Add New File** is used to create a new file and simultaneously add it to your project.

A subset of these options can be accessed from the context menu that appears when right-clicking your mouse on the surface of the project manager window.

## 3.2. Removing Files and Directories

To remove a specific file or directory, select it and use the **Remove From Project** menu item in the right-click context menu from the surface of the Project Manager window, or by selecting an item on the project and using **Remove Selected Entry** in the Project menu.

If the removed file or directory is part of another directory that has been added to the project, the removal is remembered as an exclusion that can be cleared from **Directory Properties**, which are accessed by right clicking on the parent directory in the Project tool.

## 3.3. Saving the Project

To save a new project, use **Save Project As** in the **Project** menu. Once a project file has been saved the first time, it will be auto-saved whenever you close the project, start a debug session, or exit Wing.

You can also save a copy of your project to another location or name with **Save Project As...** in the **Project** menu.

### **Moving Project Files**

When moving a project file on disk, doing so in a file browser or from the command line may partially break the project if it is moved relative to the position of files that it includes. Using **Save Project As...** in Wing instead will properly update the relative paths that the project manager uses to locate files in the project.

## 3.4. Sorting the View

The project can be set to show your files in one of several modes, using the **Options** menu in the top right of the project view:

- **View As Tree** -- This displays the project files in true tree form. The tree structure is based on the partial relative path from the project file.
- **View As Flattened Tree** -- This view (the default) shows files organized according to their location on disk. Each directory is shown at the top level with path names shown as partial relative paths based on the location of the project file. If you alter the location of the project file with **Save Project As...**, these paths will be updated accordingly.

Several sorting options are available to sort items within their directory by name, mime type, or extension. The **List Files Before Directories** option may be used to control whether files or directories are shown first in the tree view.

## 3.5. Navigating to Files

Files can be opened from the project manager window by double clicking or middle clicking on the file name, or right-clicking and using the **Open in Wing IDE** menu item.

Files may also be opened using an external viewer or editor by right-clicking on the file and using the **Open in External Viewer** item. On Windows and Mac OS X, this opens the file as if you had double clicked on it. On Linux, the preferences **File Display Commands** and **Extra Mime Types** can be used to configure how files are opened.

You can also execute Makefiles, Python source code, and any executable files by selecting the **Execute Selected** item from the popup menu. This executes outside of the debugger with any input/output occurring in the **OS Commands** tool. Doing so also adds the command to the OS Commands tool, where its runtime environment can be configured.

### 3.5.1. Keyboard Navigation

Once it has the focus, the project manager tree view is navigable with the keyboard, using the up/down arrow keys, page up and page down, and home/end.

Use the right arrow key on a parent to display its children, or the left arrow key to hide them.

Whenever a file is selected, pressing enter will open that item into an editor in Wing IDE.

## 3.6. Project-wide Properties

Each project has a set of top-level properties that can be accessed and edited via the **Properties** item in the Project menu. These can be used to configure the Python environment used when debugging, executing, or testing code, and for the **source code analysis** engine, which drives Wing's auto completion, source index, and other capabilities. Project properties are also provided to set options for the project and to enable and configure extensions for revision control, Zope, and other tools.

Any string value for a property may contain environment and special variable references, as described in **Variable Expansion**.

### Environment

To get the most out of Wing, it is important to set these values in the Environment tab correctly for your project:

**Python Executable** -- When the **Custom** radio button is checked and the entered field is non-blank, this can be used to set the full path to the Python executable that should be used when debugging source code in this project. When **Use default** is selected, Wing tries to use the default Python obtained by typing `python` on the command line. On OS X, Wing prefers the latest Apple-provided Python. If this fails, Wing will search for Python in `/usr/local` and `/usr` (on Linux and OS X) or in the registry (on Windows).

**Python Path** -- The `PYTHONPATH` is used by Python to locate modules that are imported at runtime with the `import` statement. When the **Use default** checkbox in this area is checked, the inherited `PYTHONPATH` environment variable is used for debug sessions. Otherwise, when **Custom** is selected, the specified `PYTHONPATH` is used.

**Environment** -- This is used to specify values that should be added, modified, or removed from the environment that is inherited by debug processes started from Wing IDE and is used to expand environment variable references used in other properties. Each entry is in `var=value` form and must be specified one per line in the provided entry area. An entry in the form `var=` (without a value) will remove the given variable so it is undefined. Note that you are operating on the environment inherited by the IDE when it started and not modifying an empty environment. When the *Use inherited environment* choice is set, any entered values are ignored and the inherited environment is used without changes.

## Debug

The following properties are defined in the Debug tab:

**Main Debug File** -- This defines where execution starts when the debugger is launched from the IDE. The default is to start debugging in the current editor file. Alternatively, use this property to define a project-wide main debug file so that debug always started in that file regardless of which file is current in the editor.

**Initial Directory** -- When the **Use default** radio button is checked, the initial working directory set for each debug session will be the directory where the debugged file is located. When **Custom** is selected, the specified directory is used instead (use `$(WING:PROJECT_DIR)` for the project's directory). This property also sets the initial directory for the Python Shell, determines how Wing resolves partial paths on the Python Path for the purposes of static analysis, and is used for other features in the IDE that require a starting directory for a sub-process. For these, Wing will use the directory of the main debug file in the project as the default initial directory, or the directory of the project file if there is no main debug file defined.

**Python Options** -- This is used to select the command line options sent to the Python interpreter while debugging. The default of `-u` sets Python into unbuffered I/O mode, which ensures that the debug process output, including prompts shown for keyboard input, will appear in a timely fashion.

**Debug Server Port** -- This can be used to alter the TCP/IP port on which the debugger listens, on a per-project basis. In this way, multiple instances of Wing using different projects can concurrently accept externally initiated debug connections. See **Advanced Debugging Topics** for details.

## Options

These project options are provided:

**Default Encoding** sets the default text encoding to use for files when the encoding cannot be determined from the contents of the file. This applies to all files opened when the project is open, whether or not they are part of the project. By default, this falls back to the value set by the **Default Encoding** preference.

**Project Home Directory** sets the base directory for the project. This overrides the project file location as the directory on which to base relative paths shown in the Project view and elsewhere. It is also used as the directory in which the Python Shell sub-process is launched and for the starting directory when the **Default Directory Policy** preference is set to **Current Project**.

**Preferred Line Ending** and **Line Ending Policy** control whether or not the project prefers a particular line ending style (line feed, carriage return, or carriage return + line feed), and how to enforce that style, if at all. By default, projects do not enforce a line ending style but rather insert new lines to match any existing line endings in the file.

**Preferred Indent Style** and **Indent Style Policy** control whether or not the project prefers a particular type of indentation style for files (spaces only, tabs only, or mixed tabs and spaces), and how to enforce that style, if at all. By default, projects do not enforce an indent style but rather insert new lines to match any existing indentation in the file.

**Strip Trailing Whitespace** controls whether or not to automatically remove whitespace at the ends of lines when saving a file to disk.

## Extensions

The Extensions tab of Project Properties is used to control add-ons on a per-project basis:

**Enable Django Template Debugging** enables Django-specific functionality that makes it possible for Wing's debugger to stop at breakpoints and step through Django template files.

**Matplotlib Event Loop Support** enabled Matplotlib-specific functionality that updates plots continuously when working interactively in the Python Shell.

**Enable Zope2/Plone Support**, **Zope2 Instance Home**, and **Zope2 Host** enable legacy support for older Zope installations. They are needed because Zope 2.x implements import magic that works differently from Python's default `import` and thus adding the instance home directory to `PYTHONPATH` is not sufficient. Wing's source analyzer needs this extra clue to properly find and process the Zope instance-specific sources.

When this option is activated, Wing will also offer to add the relevant Zope2/Plone files to the project, and to install the control panel for configuring and initiating debug in Zope2/Plone. See the **Zope How-To** for details.

### 3.6.1. Variable Expansion

Any string value for a property may contain environment variable references using the `$(name)` or `$ {name}` notation. These will be replaced with the value of the environment variable when it is used by the IDE. If the environment variable is not set, the reference will

be replaced by an empty string. The system environment, as modified by the project-wide or per-file environment property (if defined), is used to expand variable references.

The following special variable names are also available for use in the `$(name)` or `${name}` form:

- `WING:FILENAME` -- full path of current file
- `WING:FILENAME_DIR` -- full path of the directory containing the current file
- `WING:LINENO` -- current line number in the current file
- `WING:SCOPE` -- x.y.z-formatted name of the current scope in the current file (if Python)
- `WING:PROJECT` full path of current project
- `WING:PROJECT_DIR` -- full path of the directory containing the current project
- `WING:PROJECT_HOME` -- full path of the project home directory
- `WING:SELECTION` -- the text selected on the current editor, if any

These may evaluate to an empty string when there is no current file name.

## 3.7. Per-file Properties

Per-file properties can be set by right-clicking on a source file and selecting the **Properties** menu item in the popup, by right-clicking on a file in the project view and selecting **File Properties**, or by opening a file and using the **Current File Properties...** item in the Source menu. For Debug and Python Settings, values entered here will override any corresponding project-wide values when the selected file is the current file or the main entry point for debugging.

Any string value for a property may contain environment and special variable references, as described in **Variable Expansion**.

### File Attributes

**File Type** -- This property specifies the file type for a given file, overriding the type determined automatically from its file extension and/or content. This property is recommended only when the **Extra File Types** preference cannot be used to specify encoding based on filename extension.

**Encoding** -- This can be used to specify the encoding with which a file will be saved. When it is altered for an already-open file, Wing will offer to reload the file using the new encoding, to only save subsequently using the new encoding, or to cancel the change. Choose to reload if the file was opened with the wrong encoding. For already-open files, the encoding attribute change is only saved if the file is saved. If it is closed without saving, the encoding attribute will revert to its previous setting. The encoding cannot be altered with this property if it is being defined by an encoding comment in a Python, HTML, XML, or gettext PO file. In this case, the file should be opened and the encoding comment changed. Wing will save the file under the newly specified encoding.

**Important:** Files saved under a different encoding without an encoding comment may not be readable by other editors because there is no way for them to determine the file's encoding if it differs from the system or disk default. Wing stores the selected encoding in the project file, but no mark is written in the file except for those encodings that naturally use a Byte Order Mark (BOM), such as utf\_16\_le, utf\_16\_be, utf\_32\_le, or utf\_32\_be. Note that standard builds of CPython cannot read source files encoded in utf16 or utf32.

**Line Ending Style** -- Specifies which type of line ending (line feed, carriage return, or carriage return and line feed) is used in the file. When altered, the file will be opened and changed in an editor. The change does not take effect until the file is saved to disk.

**Indent Style** -- This property can be used in non-Python files to change the type of indent entered into the file for newly added lines. For Python files, the only way to alter indentation in a file is with the **Indentation manager**.

**Read-only on Disk** -- This property reflects whether or not the file is marked read-only on disk. Altering it will change the file's disk protections for the owner of the file (on Posix, group/world permissions are never altered).

## Editor

These properties define how the file is displayed in the editor:

**Show Whitespace** -- This allows overriding the **Show White Space** preference on a per-file basis.

**Show EOL** -- This allows overriding the **Show EOL** preference on a per-file basis.

**Show Indent Guides** -- This allows overriding the **Show Indent Guides** preference on a per-file basis.

**Ignore Indent Errors** -- Wing normally reports potentially serious indentation inconsistency in Python files. This property can be used to disable this check on a per-file basis (it is also available in the warning dialog).



**Ignore EOL Errors** -- When the project's **Line Ending Policy** is set to warn about line ending mismatches, this property can be used to disable warnings for a particular file.

## Environment

These properties are the same as for the Python Settings defined in **Project-Wide Properties**. Values defined per-file override the corresponding project-wide property.

For the **Environment** attribute, note that the option menu area contains some additional choices. Use *Add to Project Values* to apply the values specified here to the runtime environment specified by the project, or *Add to System Environment* to bypass the project-wide values and apply the per-file values directly to the environment set by the operating system.

## Debug

The per-file debug properties dialog contains all the same fields described in **Project-Wide Properties**, with the following additions:

**Run Arguments** -- Enter any run arguments here. Wing does not interpret backslashes (") on the command line and passes them unchanged to the debug process. The only exceptions to this rule are ' and " (backslash followed by single or double quote), which allow inclusion of quotes inside quoted multi-word arguments.

**Show this dialog before each run** -- Check this checkbox if you want the debug options dialog to appear each time you start a debug session.

Values defined per-file override or modify the corresponding project-wide property.

When debugging, only per-file debug properties set on the *initially invoked file* are used. Even if other files with set properties are used in the debug session, any values set for them will be ignored.



# Source Code Editor

Wing IDE's source code editor is designed to make it easier to adopt the IDE even if you are used to other editors.

## Key things to know about the editor

- The editor has personalities that emulate other commonly used editors such as Visual Studio, VI/Vim, Emacs, and Brief.
- Context-appropriate auto-completion, goto-definition, and code index menus are available when working in Python code
- The editor supports a wide variety of file types for syntax colorization.
- Key mappings and many other behaviors are configurable.
- The editor supports structural folding for some file types

## 4.1. Syntax Colorization

The editor will attempt to colorize documents according to their MIME type, which is determined by the file extension, or content. For example, any file ending in `.py` will be colorized as a Python source code document. Any file whose MIME type cannot be determined will display all text in black normal font by default.

All the available colorization document types are listed in the **File Properties** dialog's File Attributes tab. If you have a file that is not being recognized automatically, you can use the **File Type** menu found there to alter the way the file is being displayed. Your selections from this menu are stored in your project file, so changes made are permanent in the context of that project.

If you have many files with an unrecognized extension, use the **Extra File Types** preference to add your extension.

Syntax coloring can be configured as described in the section **Syntax Coloring**.

## 4.2. Right-click Editor Menu

Right-clicking on the surface of the editor will display a context menu with commonly used commands such as Copy, Paste, Goto Definition, and commenting and indentation operations.

## 4.3. Navigating Source

The set of menus at the top of the editor can be used to navigate through your source code. Each menu indicates the scope of the current cursor selection in the file and may be used to navigate within the top-level scope, or within sub-scopes when they exist.

When editor tabs are hidden by clicking on the options drop down in the top right of the editor area, the left-most of these menus lists the currently open files by name.

You can also use the **Goto Definition** menu item in the editor context menu to click on a construct in your source and zoom to its point of definition. Alternatively, place the cursor or selection on a symbol and use the **Goto Selected Symbol Defn** item in the **Source** menu, or its keyboard equivalent.

When moving around source, the history buttons in the top left of the editor area can be used to move forward and backward through visited files and locations within a file in a manner similar to the forward and back buttons in a web browser.

Other commonly used ways to select among files that are open include the **Window** menu, which lists all open files, and the **Recent** sub-menu in the **File** menu.

Additionally, the **Open From Keyboard** command in the **File** menu can be a convenient way to find files quickly. This operates in a temporary input area at the bottom of the IDE window and offers auto-completion of file names as you type.

## 4.4. File status and read-only files

The editor tabs, or editor selection menu when the tabs are hidden, indicate the status of the file by appending **\*** when the file has been edited or **(r/o)** when the file is read-only. This information is mirrored for the current file in the status area at the bottom left of

each editor window. Edited status is also shown in the **Window** menu by appending \* to the file names found there.

Files that are read-only on disk are initially opened within a read-only editor. Use the file's context menu (right-click) to toggle between read-only and writable state. This alters both the editability of the editor and the writability of the disk file so may fail if you do not have the necessary access permissions to make this change.

## 4.5. Transient vs. Sticky Editors

Wing can open files in several modes that control how and when files are closed:

**Transient Mode** -- Files opened when searching, debugging, navigating to point of definition or point of use, and using the Project or Source Browser tools with the **Follow Selection** checkbox enabled are opened in transient mode and will be automatically closed when hidden.

The maximum number of non-visible transient files to keep open at any given time can be set with the **Editor / Advanced / Transient Threshold** preference.

**Sticky Mode** -- Files opened from the File menu, from the keyboard file selector, or by double clicking on items in the Project or Source Browser tools will be opened in sticky mode, and are kept open until they are explicitly closed.

A file can be switched between these modes by clicking on the stick pin icon in the upper right of the editor area.

Right-click on the stick pin icon to navigate to files that were recently visited in the associated editor or editor split. Blue items in the menu were visited in transient state and black items were sticky. Note that this differs from the Recent area in the File menu, which lists only sticky file visits and includes visits for all editors and editor splits.

Transient files that are edited are also automatically converted to sticky mode.

## 4.6. Auto-completion

Wing Personal and Professional display an auto-completer in the editor and shells. When the completer appears, type until the correct symbol is highlighted in the list, or use the up/down arrow keys, and then press the Tab key or double click on an item. Wing will fill in the remaining characters for the source symbol, correcting any spelling errors you might have made in the name.

To alter which keys cause auto-completion to occur, use the **Auto-completion Keys** preference. Ctrl-click on the list to select multiple keys. For printable keys such as '.', '(', '[', and ':' the key will be added to the editor and any relevant **auto-editing** operations will be applied. For '.' the completer will be shown again for the attributes of the completed symbol.

To cancel out of the auto-completion popup, press the **Esc** key or **Ctrl-G**. The auto-completer will also disappear when you exit the source symbol (for example, by pushing **space** or any other character that isn't a completion key and can't be contained in a source symbol), if you click elsewhere on the surface of the source code, or if you issue other keyboard-bound commands that are not accepted by the auto-completer (for example, **save** through keyboard equivalent).

The completer can be configured to display immediately, only after a specified number of characters, or after a time delay. Completion may be case sensitive or insensitive and the completer may be auto-hidden after a specified timeout. These and other configuration options are in the **Auto-completion preferences group**.

### **Turbo Completion Mode for Python (Experimental)**

When the **Python Turbo Mode** preference is enabled, Wing will use a different completion mode for Python files and in the shells. This treats any non-word key as being a completion key, in a context appropriate way. **Ctrl**, **Alt**, and **Command** act as cancel keys, in addition to **Esc**.

This mode can be considerably faster to use when the completer contains the desired text. Once the correct completion is selected in the completer, the next source code character can immediately be typed. The completion will be placed, the next key will be entered into the editor, any relevant **auto-editing** operations will be applied, and the completer shown again if appropriate.

In contexts where a new symbol is being defined, Wing disables Turbo mode depending on the character being pressed. For example, pressing = after a name at the start of a line, entering an argument name in a def, and entering a symbol after **for** all define a new symbol in most cases. In these contexts, **Tab** must be pressed to cause completion to occur.

The draw-back of operating in this mode is that Wing may fail to recognize some contexts where a new symbol is being defined, or may enter undesired completions when code is being typed before a referenced symbol has been defined. To make canceling from the completer more convenient in this case, **Ctrl**, **Alt**, and **Command** are also treated as cancel keys, in addition to **Esc**.

For the same reason, snippets do not participate in Turbo mode. To enter snippets found in the auto-completer, press **Tab** or **Enter**.

This mode is experimental. Please email feedback and suggestions to [support@wingware.com](mailto:support@wingware.com).

## How Auto-completion Works

The information shown in Wing's auto-completer comes from several sources: (1) Static analysis of Python code, (2) introspection of extension module contents, (3) inspection of keywords and builtins in the active Python version, (4) introspection of the runtime application state when the debugger is active or when working in the shells, (5) enumeration of relevant code snippets, and in some cases (6) user-provided interface description files. See **Source Code Analysis** for more information on how static analysis works and how you can help Wing determine the types of values.

Because static analysis can be defeated by Python's dynamic nature, it is sometimes more effective to work from live runtime state. This can be done by placing a breakpoint in the source code, running to it, and then working in the editor or (in Wing IDE Pro) in the Debug Probe.

In non-Python files, the auto-completer is limited to words found within similar contexts in the file, keywords defined for syntax highlighting that file type, and any snippets relevant to the editing context.

## 4.7. Source Assistant

The **Source Assistant** tool (in Wing IDE Personal and higher) can be used while viewing or editing source code to display additional information about the point of definition of source constructs located near the insertion caret's position.

The display will include links to the point of definition of a selection symbol, a guess at the symbol's type (when available) and a link to the type's point of definition, and docstrings and call signature when available.

For symbols in the Python standard library, Wing will attempt to compute a documentation URL whenever possible. These point to <http://docs.Python.org/> but can be redirected to another server with the **Source Analysis > Advanced > Python Docs URL Prefix preference**. To access locally stored documentation, a local http server must be used because # bookmark references do not work with file: URLs.

Note that the source assistant is also integrated with the auto-completer, and will show information as you scroll through the completion list. Similarly, it will updated as focus moves into the **Project** and **Source Browser** tools.

When working in the editor, auto-completer, project view, or source browser, the source assistant is fueled by Wing's Python source code analysis engine. Because of Python's

dynamic nature, Wing cannot always determine the types of all values, but presents as much information as it can glean from the source code.

When a debug process is active, or when working in the **Python Shell**, Wing also extracts relevant information from the live runtime state. Since this yields complete and correct type information even for code that Wing's static analysis engine cannot understand, it is often useful to run to a breakpoint before designing new code that is intended to work in that context.

For more hints on helping Wing understand your source code, see **Source Code Analysis** and **Helping Wing Analyze Code**.

## 4.8. Indentation

Since indentation is syntactically significant in Python, Wing provides a range of features for inspecting and managing indentation in source code.

### 4.8.1. How Indent Style is Determined

When an existing file is opened, it is scanned to determine what type of indentation is used in that file. If the file contains some indentation, this may override the tab size, indent size, and indent style values given in preferences and the file will be indented in a way that matches its existing content rather than with your configured defaults. If mixed forms of indentation are found, the most common form is used.

For non-Python files, you can change indentation style on the fly using the **Indent Style** property in the **File Properties** dialog (accessed by right-clicking on the editor). This allows creating files that intentionally mix indentation forms in different parts of the file. To ask Wing to return to the form of indentation it determines as most prominent in the file, select **Match Existing Indents**.

For Python files, the **Indent Style** cannot be altered without converting the whole file's indent style using the **Indentation Manager**, which can be accessed from the button next to the **Indent Style** property and from the Tools menu.



**Tab Size**

Tab size is automatically forced to 8 characters for all Python source files that contain some spaces in indentation. This is done because the Python interpreter defines tabs as 8 characters in size when used together with spaces. This version of Wing does not recognize vi style tab size comments, but it does apply the **Tab Size** preference when a file contains only tabs in indentation, or if it is a non-Python file.

## 4.8.2. Indentation Preferences

The following preferences affect how the indentation features behave:

- 1) The **Use Indent Analysis** preference is used to control whether analysis of current file content is used to determine the type of indentation placed during edits. It can be enabled for all files, only for Python files, or disabled. Note that disabling this preference for Python files can result in a potentially broken mix of indentation in the files. In general, indent styles should not be mixed within a single Python file.
- 2) The **Default Tab Size** preference defines the position of tab stops and is used to determine the rendering of files with tabs only, or non-Python files with mixed tab and space indentation. In Python files with mixed indents, this value is ignored and the file is always shown in the way that the Python interpreter would see it.
- 3) The **Default Indent Size** preference defines the default size of each level of indent, in spaces. This is used in new empty files or when indent analysis has been disabled. Wing may override this value in files that contain only tabs in indentation, in order to make it a multiple of the configured tab size.
- 4) The **Default Indent Style** preference defines the default indentation style, one of **spaces-only**, **tabs-only**, or **mixed**. This is used in new empty files or when indent analysis has been disabled. Mixed indentation replaces each tab-size spaces with one tab character.

These preferences define how indentation is handled by the editor:

- 5) The **Auto-Indent** preference controls whether or not each new line is automatically indented.
- 6) The **Show Indent Guides** preference controls whether or not to show indentation guides as light vertical lines. This value can be overridden on a file-by-file basis from Editor tab in **File Properties**.

- 7) The **Show Python Indent Warnings** preference can be used to enable or disable warnings for Python files that may contain confusing or damaged indentation.
- 8) The **Show Override Warnings** preference controls whether or not Wing shows a warnings when the user enters indentation that does not match the form already within a file. This is currently only possible in non-Python files, by altering the **Indent Style** attribute in **File Properties**.

### 4.8.3. Indentation Policy

The project manager also provides the ability to define the preferred indentation style (overriding the preference-defined style) and to specify a policy for enforcing line endings, on a per-project basis. This is accomplished with **Preferred Line Ending** and **Line Ending Policy** under the Options tab in Project Properties.

### 4.8.4. Auto-Indent

The IDE ships with auto-indent turned on. This causes leading white space to be added to each newly created line, as return or enter are pressed. Enough white space is inserted to match the indentation level of the previous line, possibly adding or removing a level of indentation if this is indicated by context in the source (such as `if`, `while`, or `return`).

Note that if preference **Auto-indent** is turned off, auto-indent does not occur until the tab key is pressed.

In Python code, Wing also auto-indents after typing a colon after `else`, `elif`, `except`, and `finally`. Indentation will go to the closest matching `if` or `try` statement. If there are multiple possible matching statements, the colon key can be pressed repeatedly to toggle through the possible positions for the line. Similarly, when **Smart Tab** is selected as the **Tab Key Action**, then pressing the Tab key repeatedly will toggle the line through the possible indent positions. This can also be accomplished with the **Indent to Match** toolbar and menu items (regardless of selected tab key action).

When pasting multiple lines into Python code and the caret is in the indent region or on a blank line, Wing will auto-indent pasted text as follows: (1) If the caret is in column zero, the text is indented to match the context, (2) If the caret is within the indent region but not in column zero, the text is indented to that position. If the auto-indent is incorrect, a single **Undo** will return the pasted text to its original indentation level, or the text can be selected and adjusted with the indentation toolbar or menu items or key equivalents.

### 4.8.5. The Tab Key

By default, the action of the tab key depends on the selected **Keyboard Personality**, file type, and position within the file as described under **Default for Personality** below.

To insert a real tab character regardless of the indentation mode or the position of the cursor on a line, type Ctrl-Tab or Ctrl-T.

The behavior of the tab key can be altered using the **Tab Key Action** preference, which provides the following options:

#### Default for Personality

This selects from the other tab key actions below according to the chosen keyboard personality, current file type, and in some cases the position of the caret within the file. In all non-Python files, the default is Move to Next Tab Stop. In Python files, the defaults are as follows by keyboard personality:

- **Normal**: Indent to Match
- **VI/VIM**: Move to Next Tab Stop
- **Emacs**: Indent to Match
- **Brief**: Smart Tab
- **Visual Studio**: Move to Next Tab Stop
- **OS X**: Smart Tab

#### Indent to Match

This indents the current line or selected lines to position them at the computed indent level for their context in the file.

#### Move to Next Tab Stop

This enters indentation characters matching the current file's style of indentation so that the caret reaches the next tab stop.

#### Indent Region

This enters indentation characters matching the current file's style of indentation to increase the indentation of the current line or selected lines by one level.

#### Insert Tab Character

This inserts a Tab character (`chr(9)`) into the file.

### Smart Tab

This option is available for Python files only. It implements the following behavior for the tab key:

- 1) When the caret is within a line or there is a non-empty selection, this performs **Indent to Match**. When the line or lines are already at the matching position, indentation is toggled between likely positions as follows:
  - (a) If a comment precedes the current line or selection, then indentation will match the position of the prior non-comment code line (if any).
  - (b) If multiple nested blocks match an `'else'`, `'elif'`, `'except'`, or `'finally'`, then indentation will match the position of the enclosing blocks (traversing each in outward order).
  - (b) In other cases, indentation is reduced by one level.
- 2) When the caret is at the end of a non-empty line and there is no selection, one indent level is inserted. The **Smart Tab End of Line Indents** preference can be used to alter the type of indentation used or to disable this aspect of the Smart Tab feature.

### 4.8.6. Checking Indentation

Wing IDE analyzes existing indentation whenever it opens a Python source file, and will indicate a potentially problematic mix of indentation styles, allowing you to attempt to repair the file. Files can be inspected more closely or repaired at any time using the **Indentation Manager**.

To turn off indentation warnings in Python files, use the **Show Python Indent Warnings** preference.

Wing also indicates suspiciously mismatched indentation in source code by underlining the indent area of the relevant lines in red or yellow. In this case, an error or warning message is displayed when the mouse hovers over the flagged area of code.

### 4.8.7. Changing Block Indentation

Wing provides **Indent** and **Outdent** commands in the **Indentation** portion of the Source menu, which increase or decrease the level of indentation for selected blocks of text. All

lines that are included in the current text selection are moved, even if the entire line isn't selected.

Indentation placed by these commands will contain either only spaces, only tabs, or a mixture of tabs and spaces, as determined by the method described in **Indentation**.

#### **Indenting to Match**

The command **Indent Lines to Match** (also in the **Indentation** sub-menu) will indent or outdent the current line or selected lines to the level as a unit so that the first line is positioned as it would have been positioned by Wing's auto-indentation facility. This is very useful when moving around blocks of code.

### **4.8.8. Indentation Manager**

The Indentation manager, accessible from the **Tools** menu, can be used to inspect and change indentation style in source files. It has two parts: (1) The indentation report, and (2) the indentation converter.

A report on the nature of existing indentation found in your source file is given above the horizontal divider. This includes the number of spaces-only, tabs-only, and mixed tabs-and-space indents found, information about whether indentation in the file may be problematic to the Python interpreter, and the tab and indent size computed for that file. The manager also provides information about where the computed tab and indent size value come from (for example, an empty file results in use of the defaults configured in preferences).

Conversion options for your file are given below the horizontal divider. The three tabs are used to select the type of conversion desired, and each tab contains information about the availability and action of that conversion, and a button to start the conversion. In some of the conversion options, the indent size field shown in the indentation report is made editable, to allow specification of the desired resulting indent size.

Once conversion is complete, the indentation manager updates to display the new status of the file, and action of any subsequent conversions.

Conversions can be undone by moving to the converted source file and selecting **Undo** from the **Edit** menu.

## 4.9. Structural Folding

The editor supports optional structural folding for Python, C, C++, Java, Javascript, HTML, Eiffel, Lisp, Ruby, and a number of other file formats. This allows you to visually collapse logical hierarchical sections of your code while you are working in other parts of the file.

You can turn Structural Folding on and off as a whole with the **Enable Folding** preference.

The **Fold Line Mode** preference can be used to determine whether or not a horizontal line is drawn at fold points, whether it is drawn above or below the fold point, and whether it is shown when the fold point is collapsed or expanded. **Fold Indicator Style** is used to select the look of the fold marks shown at fold points.

Once folding is turned on, an additional margin appears to the left of source files that can be folded. Left mouse click on one of the fold marks in this margin to collapse or expand that fold point. Right mouse clicking anywhere on the fold margin displays a context menu with the various folding operations.

You can also hold down the following key modifiers while left-clicking to modify the folding behavior:

- **Shift** -- Clicking on any fold point while holding down the shift key will expand that point and all its children recursively so that the maximum level of expansion is increased by one.
- **Ctrl** -- Clicking on any fold point while holding down the ctrl key will collapse that point and all its children recursively so that the maximum level of expansion is decreased by one.
- **Ctrl+Shift** -- On a currently expanded fold point, this will collapse all child fold points recursively to maximum depth, as well as just the outer one. When the fold point is subsequently re-expanded with a regular click, its children will appear collapsed. Ctrl-shift-click on a collapsed fold point will force re-expansion of all children recursively to maximum depth.

Fold commands are also available in the **Structural Folding** section of the **Source** menu, which indicates the key equivalents assigned to the operations:

- **Toggle Current Fold** -- Like clicking on the fold margin, this operates on the first fold point found in the current selection or on the current line.
- **Collapse Current More** -- Like ctrl-clicking, this collapses the current fold point one more level than it is now.

- **Expand Current More** -- Like shift-clicking, this expands the current fold point one more level than it is now.
- **Collapse Current Completely** -- Like shift-ctrl-clicking on an expanded node, this collapses all children recursively to maximum depth.
- **Expand Current Completely** -- Like shift-ctrl-clicking on a collapsed node, this ensures that all children are expanded recursively to maximum depth.
- **Collapse All** -- Unconditionally collapse the entire file recursively.
- **Expand All** -- Unconditionally expand the entire file recursively.
- **Fold Python Methods** -- Fold up all methods in all classes in the file.
- **Fold Python Classes** -- Fold up all classes in the file.
- **Fold Python Classes and Defs** -- Fold up all classes and any top-level function definitions in the file.

## 4.10. Brace Matching

Wing will highlight matching braces in green when the cursor is adjacent to a brace. Mismatched braces are highlighted in red.

You can cause Wing to select the entire contents of the innermost brace pair from the current cursor position with the Match Braces item in the Source menu.

Parenthesis, square brackets, and curly braces are matched in all files. Angle brackets (< and >) are matched also in HTML and XML files.

## 4.11. Support for files in .zip or .egg files

Source and other text files stored in .zip or .egg files may be loaded into the editor as readonly files. Wing is unable to write changes to a file within a .zip or .egg file or otherwise write to or create a .zip or .egg file.

When stepping through code, using goto definition, or using other methods to goto a line in a file, a file within a .zip or .egg file will be opened automatically. To open a file through the open file dialog, specify the name of the .zip or .egg file and add a / followed by the name of the file to open.

## 4.12. Keyboard Macros

The `Edit` menu contains items for starting and completing definition of a keyboard or command sequence macro, and for executing the most recently defined macro. Once macro recording is started, any keystroke or editor command is recorded as part of that macro, until macro recording is stopped again. Most commands may be included in macros, as well as all character insertions and deletions.

Macros can be quite powerful by combining keyboard-driven search (**Mini-search** in the `Edit` menu), cursor movements, and edits.

## 4.13. Notes on Copy/Paste

There are a number of ways to cut, copy, and paste text in the editor:

- Use the `Edit` menu items. This stores the copy/cut text in the system-wide clipboard and can be pasted into or copied from other applications.
- Use key equivalents as defined in the `Edit` menu.
- Right-click on the editor surface and use the items in the popup menu that appears.
- Select a range of text and drag it using the drag and drop feature. The default drag operation is to *copy* on Linux and OS X and *move* on Windows. Pressing the Control key after starting the drag toggles between moving or copying the text.
- On Linux, select text anywhere on the display and then click with the middle mouse button to insert it at the point of click.
- On Windows and Mac OS X, click with the middle mouse button to insert the current emacs private clipboard (if in emacs mode and the buffer is non-empty) or the contents of the system-wide clipboard (in all other cases). On Mac OS X, the middle mouse button is emulated by command or other key configured in the X11 Server's preferences. This behavior may be disabled via the **Middle Mouse Paste** preference

## 4.14. Auto-reloading Changed Files

Wing's editor detects when files have been changed outside of the IDE and can reload files automatically, or after prompting for permission. This is useful when working with an external editor, or when using code generation tools that rewrite files.



Wing's default behavior is to automatically reload externally changed files that have not yet been changed within Wing's source editor, and to prompt to reload files that have also been changed in the IDE.

You can change these behaviors by setting the value of the **Reload when Unchanged** and **Reload when Changed** preferences

On Windows, Wing uses a signal from the OS to detect changes so notification or reload is usually instant. On Linux and Unix, Wing polls the disk by default every 3 seconds; this frequency can be changed with the **External Check Freq** preference.

## 4.15. Auto-save

The source code editor auto-saves files to disk every few seconds. The auto-save files are placed in a subdirectory of your **User Settings Directory**.

If Wing ever crashes or is killed from the outside, you can use these files to manually recover any unsaved changes. Copy the auto-save files to overwrite the older unsaved files, doing a comparison first to verify that the auto-save file is what you want.



# Search/Replace

Wing provides a number of tools for search and replace in your source code. Which you use depends on the complexity of your search or replace task and what style of searching you are most familiar with.

## 5.1. Toolbar Quick Search

One way to do simple searches is to enter text in the search area of the toolbar. This scrolls as you type to the next match found after the current cursor position. Pressing **Enter** will search for each subsequent match, wrapping the search when the end of the file is reached.

Text matching during toolbar quick search is case-insensitive unless you enter a capital letter as part of your search string.

If focus is off the toolbar search area and it already contains a search string, clicking on it will immediately start searching in the current source editor for the next match. If you wish to search for another string instead, delete the text and type the desired search string. As you delete, the match position in the editor will proceed backward until it reaches your original search start position, so that after typing your new search string you will be presented with the first match after the original source editor cursor position.

## 5.2. Keyboard-driven Mini-Search/Replace

The Edit menu contains a Mini-Search sub-menu that enumerates the available keyboard-driven search options. These are normally initiated with the keyboard command sequences shown in the menu and can be controlled entirely by using the keyboard. All interaction with the mini-search manager occurs using data entry areas displayed on demand at the bottom of the IDE window.

The implementation of the mini-search manager is very close to the most commonly

used search and replace features found in Emacs, but it is available whether or not the Emacs editor personality is being used.

The following search and replace features are available in this facility:

- **Forward** and **Backward** -- These display a search string entry area at the bottom of the IDE window and interactively search forward or backward in the current source editor, starting from the current cursor position. The search takes place as you type and can be aborted with **Esc** or **Ctrl-G**, which returns the editor to its original cursor location and scroll position.

Searching is case-insensitive unless you enter a capital letter as part of your search string. To search repeatedly, press **Ctrl-U** (or **Ctrl-S** in emacs keyboard mode) to search forward and ‘‘**Ctrl-Shift-U** (or **Ctrl-R** in emacs mode) to search in reverse. The search direction can be changed any number of times and searching will wrap whenever the top or bottom of the file is reached. You can also enter **Ctrl-U** (or **Ctrl-S** in emacs mode) or **Ctrl-Shift-U** (or **Ctrl-R** in emacs mode) again initially while the search string is still blank in order to call up the most recently used search string and begin searching forward or backward with it.

Once the mini-search entry area is visible, **Ctrl-W** will add the current word in the editor to the search string. Pressing **Ctrl-W** more than once while the mini-search entry is visible adds additional words from the editor to the search string.

- **Selection Forward** and **Selection Backward** -- These work like the above but start with the selection in the current source editor.
- **Regex Forward** and **Regex Backward** -- These work like the above but treat the search string as a regular expression.
- **Query/Replace** and **Regex Query/Replace** -- This prompts for search and replace strings in an entry area at the bottom of the IDE window and prompts for replace on each individual match found after the cursor location in the current source editor. Press **y** to replace or **n** to skip a match and move on to the next one. The interaction can be canceled at any time with **Esc** or **-G**. Matching is case insensitive unless a capital letter is entered as part of the search string. Searching is always forward and stops when the end of the file is reached, without wrapping to any un-searched parts between the top of the file and the position from which the search was started.
- **Replace String** and **Replace Regex** -- This works like the above command but immediately replaces all matches without prompting.

## 5.3. Search Tool

The dockable **Search** tool can be used for more advanced search and replace tasks within the current editor. It provides the ability to customize case sensitivity and whole/part word matching, search in selection, and perform wildcard or regex search and replace.

The **Replace** field may be hidden and can be shown from the **Options** menu in the bottom right of the tool.

To the right of the **Search** and **Replace** fields, Wing makes available a popup that contains a history of previously used strings, options for inserting special characters, and an option for expanding the size of the entry area.

The following search options can be selected from the tool:

- **Case Sensitive** -- Check this option to show only exact matches of upper and lower case letters in the search string.
- **Whole Words** -- Check this option to require that matches are surrounded by white space (spaces, tabs, or line ends) or punctuation other than `_` (underscores).
- **In Selection** -- Search for matches only within the current selection on the editor.

The following additional options are available from the Options popup menu:

- **Show Replace** -- Whether or not the Replace field is visible in the tool.
- **Text Search** -- Select this to do a regular text search without wildcard or regex.
- **Wildcard Search** -- Select this to allow use of special characters for wildcarding in the search string (see **Wildcard Search Syntax** for details).
- **Regex Search** -- Select this to use regular expression style searching. This is a more powerful variant than wildcard search that allows for more complex specification of search matches and replacement values. For information on the syntax allowed for the search and replace strings, see Python's [Regular Expression Syntax](#) documentation. In this mode, the replace string can reference regex match groups with `\1`, `\2`, etc, as in the Python `re.sub()` call.
- **Wrap Search** -- Uncheck this to avoid wrapping around when the search reaches the top or bottom of a file.
- **Incremental** -- Check this to immediately start or restarted searching as you type or alter search options. When unchecked, use the forward/backward search buttons to initiate searching.

- **Find After Replace** -- Select this to automatically find the next search match after each Replace operation.

## 5.4. Search in Files Tool

The dockable **Search in Files** tool is used to search and replace within sets of files, or for searching Wing's documentation. It performs searches in batch and displays a result list for all found matches. This list can then be traversed to view the matches in the source editor, and is automatically updated as edits alter the search results. Searching may span the current editor, a single selected file, all open files, all project files, all of Wing's documentation, or sets of files on disk.

Files in a set may be filtered by file type, for example searching only through Python files in the project.

In addition the options also available in the **search tool**, the following choices are available in the **Options** popup menu:

- **Replace Operates On Disk** -- Check this to replace text in un-opened files directly on disk. Caution: see **Replace in Multiple Files** for details on this option.
- **Recursive Directory Search** -- Check this to search recursively within all sub-directories of the selected search directory.
- **Omit Binary Files** -- Check this to omit any file that appears to contain binary data.
- **Auto-restart Searches** -- Check this to restart searching immediately if it is interrupted because a search parameter or the set of files being searched is changed.
- **Open First Match** -- Check this to automatically open the first batch search match, even before the result list is clicked upon.
- **Show Line Numbers** -- Check this to include line numbers in the batch result area.
- **Result File Name** -- This is used to select the format of the result file name shown in the batch result area.

### 5.4.1. Replace in Multiple Files

For searches that operate on open files, replace always occurs in the open file editor and can be undone or saved to disk subsequently, as with any other edit operation.

When replacing text in batch mode, some of the files being searched may not currently be open in an editor. In this case, Wing will by default open all altered files and make changes in newly created editors that remain open until the user saves and closes them explicitly. This is the safest way to undertake multi-file global replace operations because it clearly shows which files have been altered and makes it possible to undo changes.

An alternative approach is available by selecting the **Replace Operates on Disk** option from the **Options** popup. This will cause Wing to change files directly on disk in cases when there is no currently open editor.

Because global replace operations can be tricky to do correctly, we *strongly* recommend using a revision control system or frequent backups and manually comparing file revisions before accepting files that have been altered.

## 5.5. Wildcard Search Syntax

For wild card searches in the Search tools, the following syntax is used:

**\*** can be used to match any sequence of characters except for line endings. For example, the search string **my\*value** would match anything within a single line of text starting with **my** and ending with **value**. Note that **\*** is "greedy" in that **myinstancevalue = myothervalue** would match as a whole rather than as two matches. To avoid this, use Regex Search instead with **.\*?** instead of **\***.

**?** can be used to match any single character except for line endings. For example, **my???value** would match any string starting with **my** followed by three characters, and ending with **value**.

**[ and ]** can be used to indicate sets of match characters. For example **[abcd]** matches any one of **a**, **b**, **c**, or **d**. Also, **[a-zA-Z]** matches any letter in the range from **a** to **z** (inclusive), either lower case or uppercase. Note that case specifications in character ranges will be ignored unless the **Case Sensitive** option is turned on.





# Interactive Python Shell

Wing provides an integrated Python Shell for execution of commands and experimental evaluation of expressions. The version of Python used in the Python Shell, and the environment it runs with, is configured in your project using **Project Properties**.

This shell runs a separate Python process that is independent of the IDE and functions without regard to the state of any running debug process.

Convenient ways to run parts of your source code in the shell include:

**Copy/Paste** part of a file -- Wing will automatically adjust leading indentation so the code can be executed in the shell.

**Drag and Drop** part of a file -- This works like Copy/Paste.

**Evaluate File in Python Shell** -- This command in the **Source** menu will evaluate the top level of the current file in the shell.

**Evaluate Selection in Python Shell** -- The command in the **Source** menu and editor's context menu (right-click) will evaluate the current selection in the shell.

**Options menu** This menu in the Python Shell tool contains items for evaluating the current file or selection

In the Python Shell, the **Up** and **Down** arrow keys will traverse the history of the code you have entered and the return key will either execute the code if it is complete or prompt for another line if it is not. **Ctrl-Up** and **Ctrl-Down** will move the cursor up and down and **Ctrl-Return** will insert a new line character at the cursor position.

To restart the Python Shell, select **Restart Shell** from the **Options** menu in the top right of the tool. This will terminate the external Python process and restart it, clearing and resetting the state of the shell.

To save the contents of the shell, use **Save a Copy** in the **Options** menu or right-click context menu. The right-click context menu also provides items for copying and pasting text in the shell.

To preload some code into the Python Shell when it is started, you can set the `PYTHON-STARTUP` environment variable, as supported by the Python Shell outside of Wing IDE.

## 6.1. Python Shell Auto-completion

Wing's Python Shell includes auto-completion, which can be a powerful tool for quickly finding and investigating functionality at runtime, for the purposes of code learning, or in the process of crafting new code. The Python Shell's completer is fueled by introspection of the runtime environment.

The **Source Assistant** will display details for the currently selected item in the auto-completer within the Python Shell. This provides quick access to the documentation and call signature of functions and methods that are being invoked.

Goto-definition will also work in the Python Shell, using a combination of live runtime state and static analysis to attempt to find the definition of the symbol or its type.

## 6.2. Python Shell Options

The **Options** menu in the Python Shell contains some settings that control how the Python Shell works:

- **Wrap Lines** causes the shell to wrap long output lines in the display
- **Filter history by entered prefix** controls whether the history will be filtered by the string between the prompt and the cursor. If history is filtered and **a** is entered at the prompt, the up arrow will find the most recent history item starting with **a**
- **Evaluate Whole Lines** causes Wing to round up the selection to the nearest line when evaluating selections, making it easier to select the desired range
- **Auto-restart when Evaluate File** causes Wing to automatically restart the shell before evaluating a file, so that each evaluation is made within a clean new environment.

# Debugger

Wing's debugger provides a powerful toolset for rapidly locating and fixing bugs in single-threaded or multi-threaded Python code.

The debugger is built around a TCP/IP client/server design that supports launching your application not just from Wing itself but also externally, as with CGI scripts or code running in an embedded scripting facility within a larger application. Remote (host to host) debugging is also provided.

Because the debugger core is written in optimized C, debug overhead is relatively low; however, you should expect your programs to run about 50% slower within the debugger.

## 7.1. Quick Start

Wing IDE can be used to debug all sorts of Python code, including scripts and stand-alone applications written with **wxPython**, Tkinter, **PyQt**, **PyGTK**, and **pygame**. Wing can also **debug web CGIs** including those running under **mod\_python**, code running under **Zope**, **Plone**, **Turbogears**, **Django**, **Paste/Pylons**, **Twisted**, and code running in an embedded Python interpreter.

This section describes how to debug stand-alone scripts and applications that can be launched from within Wing IDE. If you wish to debug web CGIs within the web server, web servlets, or embedded Python scripts, please refer to **Debugging Externally Launched Code** and, for remote host-to-host debugging, see **Remote Debugging**.

Before debugging, you will need to install Python on your system if you have not already done so. Python is available from [www.python.org](http://www.python.org).

To debug Python code with Wing, open up the Python file and select **Start / Continue** from the Debug menu. This will run to the first breakpoint, unhandled exception, or until the debug program completes. Select **Step Into** instead to run to the first line of code.

Use the Debug I/O tool to view your program's output, or to enter values for input to

the program. If your program depends on characteristics of the Windows Console or a particular Linux/Unix shell, see **External I/O Consoles** for more information.

In some cases, you may also need to enter a `PYTHONPATH` and other environment values using the **Project Properties** dialog available from the Project menu. This can also be used to specify which Python executable should be used to run with your debug process. Use this if Wing IDE cannot find Python on your system or if you have more than one version of Python installed.

To set breakpoints, just click on the leftmost part of the margin next to the source code.

## 7.2. Specifying Main Entry Point

Normally, Wing will start debugging in whatever file you have active in the frontmost editor. Depending on the nature of your project, you may wish to specify a file as the default debug entry point. This is done with **Set Current As Main Debug File** in the **Debug** menu, by right clicking on a file in the Project tool and selecting **Set As Main Debug File**, or by setting **Main Debug File** in Project Properties.

When a main debug entry point is specified, it is used whenever you start the debugger, except when using **Debug Current File** in the **Debug** menu, or when right-clicking on an entry in the project manager and choosing the **Debug Selected** context menu item.

Note that the path to the main debug file is highlighted in red in the project window.

The main entry point defined for a project is also used by the source code analysis engine to determine the python interpreter version and Python path to use for analysis. Thus, changing this value will cause all source files in your project to be reanalyzed from scratch. See section **Source Code Analysis** for details.

## 7.3. Debug Properties

In some cases, you may need to set project and per-file properties from the Project manager before you can debug your code. This is done to specify Python interpreter, `PYTHONPATH`, environment variables, command line arguments, start directory, and other values associated with the debug process. For details, see **Project-Wide Properties** and **Per-file Properties**.

## 7.4. Setting Breakpoints

Breakpoints can be set on source code by opening the source file and clicking on the breakpoint margin to the left of a line of source code. Right-clicking on the breakpoint margin will display a context menu with additional breakpoint operations and options. Alternatively, the **Debug** menu or the toolbar's breakpoint icons can be used to set or clear breakpoints at the current line of source (where the insertion cursor or selection is located).

## 7.5. Starting Debug

There are several ways in which to start a debug session from within Wing:

- Choose **Start / Continue** from the **Debug** menu or push the **Debug** icon in the toolbar. This will run the main debug file if one has been defined (described in **Setting a Main Debug File**), or otherwise the file open in the frontmost editor window. Execution stops at the first breakpoint or exception, or upon program completion.
- Choose **Step Into** from the **Debug** menu or push the **Step Into** icon in the toolbar. This will run the main debug file if one has been defined, or otherwise the file open in the frontmost editor window. Execution stops at the first line of code.
- Choose **Debug Current File** from the **Debug** menu or **Debug Selected** from the right-click popup menu on the Project tool to run a specific file regardless of whether a main debug file has been specified for your project. This will stop on the first breakpoint or exception, or upon program completion.
- Choose **Run to Cursor** from the **Debug** menu or toolbar. This will run the main debug file if one has been defined or otherwise the file open in the frontmost editor window. Execution continues until it reaches the line selected in the current source text window, until a breakpoint or exception is encountered, or until program completion.
- Use **Debug Recent** in the **Debug** menu to select a recently debugged file. This will stop on the first breakpoint or exception, or upon program completion.
- Use one of the key bindings given in the **Debug** menu.

Additional options exist for initiating a debug session from outside of Wing and for attaching to an already-running process. These are described in sections **Debugging Externally Launched Code** and **Attaching**, respectively.

Once a debug process has been started, the status indicator in the lower left of the window should change from white or grey to another color, as described in **Debugger Status**.

## 7.6. Debugger Status

The debugger status indicator in the lower left of editor Windows is used to display the state of the debugger. Mousing over the bug icon shows expanded debugger status information in a tool tip. The color of the bug icon summarizes the status of the debug process, as follows:

- **White** -- There is no debug process, but Wing is listening for a connection from an externally launched process.
- **Gray** -- There is no debug process and Wing is not allowing any external process to attach.
- **Green** -- The debug process is running.
- **Yellow** -- The debug process is paused or stopped at a breakpoint.
- **Red** -- The debug process is stopped at an exception.

The current debugger status is also appended to the Debugger status group in the IDE's **Messages** tool.

## 7.7. Flow Control

Once the debugger is running, the following commands are available for controlling further execution of the debug program from Wing. These are accessible from the tool bar and the **Debug** menu:

- At any time, a freely running debug program can be paused with the **Pause** item in the **Debug** menu or with the pause tool bar button. This will stop at the current point of execution of the debug program.
- At any time during a debug session, the **Stop Debugging** menu item or toolbar item can be used to force termination of the debug program. This option is disabled by default if the current process was launched outside of Wing. It may be enabled for all local processes by using the **Kill Externally Launched** preference.

When stopped on a given line of code, execution can be controlled as follows from the **Debug** menu or tool bar:

**Step Over** will step over a single byte code operation in Python. This may not leave the current line if it contains something like a list comprehension or single-line for loop.

**Step Into** will attempt to step into the next executed function on the current line of code. If there is no function or method to step into, this command acts like Step Over.

**Step Out** will complete execution of the current function or method and stop on the first instruction encountered after returning from the current function or method.

**Continue** will continue execution until the next breakpoint, exception, or program termination

**Run To Cursor** will run to the location of the cursor in the frontmost editor, or to the next breakpoint, exception, or program termination.

## 7.8. Viewing the Stack

Whenever the debug program is paused at a breakpoint or during manual stepping, the current stack is displayed in the **Call Stack** tool. This shows all program stack frames encountered between invocation of the program and the current run position. Outermost stack frames are higher up on the list.

When the debugger steps or stops at a breakpoint or exception, it selects the innermost stack frame by default. In order to visit other stack frames further up or down the stack, select them in the **Call Stack** tool. You may also change stack frames using the **Up Stack** and **Down Stack** items in the **Debug** menu, the up/down tool bar icons, the stack selector popup menus the other debugging tools.

When you change stack frames, all the tools in Wing that reference the current stack frame will be updated, and the current line of code at that stack frame is presented in an editor window.

To change the type of stack display, right-click on the **Call Stack** tool and select from the options for the display and positioning of the code line excerpted from the debug process.

When an exception has occurred, a backtrace is also captured by the **Exceptions** notification tool, where it can be accessed even after the debug process has exited.

## 7.9. Viewing Debug Data

Wing IDE allows you to inspect locals and globals using the **Stack Data** tool. This area displays values for the currently selected stack frame.

### Values Fetched on Demand

The variable data displayed by Wing is fetched from the debug server on the fly as you navigate. Because of this, you may experience a brief delay when a change in an expansion or stack frame results in a large data transfer.

For the same reason, leaving large amounts of debug data visible on screen may slow down stepping through code.

### 7.9.1. Stack Data View

The **Stack Data** debugger tool contains a popup menu for selecting thread (in multi-threaded processes) and accessing the current debug stack, a tree view area for browsing variable data in locals and globals, and a textual view area for inspecting large data values that are truncated on the tree display.

Simple values, such as strings and numbers, and values with a short string representation, will be displayed in the value column of the tree view area.

Strings are always contained in `"` (double quotes). Any value outside of quotes is a number or internally defined constant such as `None` or `Ellipsis`.

Integers can be displayed as decimal, hexadecimal, or octal, as controlled by the **Integer Display Mode** preference.

Complex values, such as instances, lists, and dictionaries, will be presented with an angle-bracketed type and memory address (for example, `<dict 0x80ce388>`) and can be expanded by clicking on the expansion indicator in the **Variable** column. The memory address uniquely identifies the construct. If you see the same address in two places, you are looking at two object references to the same instance.

If a complex value is short enough to be displayed in its entirety, the angle-bracketed form is replaced with its value, for example `{'a': 'b'}` for a small dictionary. These short complex values can still be expanded in the normal way.

Upon expansion of complex data, the position or name of each sub-entry will be displayed in the **Variable** column, and the value of each entry (possibly also complex values) will be displayed in the **Value** column. Nested complex values can be expanded indefinitely, even if this results in the traversal of cycles of object references.



Once you expand an entry, the debugger will continue to present that entry expanded, even after you step further or restart the debug session. Expansion state is saved for the duration of your Wing IDE session.

When the debugger encounters a long string, it will be truncated in the **Value** column. In this case, the full value of the string can be viewed in the textual display area at the bottom of the Stack Data tool, which is accessed by right-clicking on a value and selecting **Show Detail**. The contents of the detail area is updated when other items in the Stack Data tool are selected.

#### **Opaque Data**

Some data types, such as those defined only within C/C++ code, or those containing certain Python language internals, cannot be transferred over the network. These are denoted with Value entries in the form `<opaque 0x80ce784>` and cannot be expanded further.

#### **7.9.1.1. Popup Menu Options**

Right-clicking on the surface of the Stack Data view displays a popup menu with options for navigating data structures:

- **Show/Hide Detail** -- Used to quickly show and hide the split where Wing shows expanded copies of values that are truncated on the main debug data view (click on items to show their expanded form).
- **Expand More** -- When a complex data value is selected, this menu item will expand one additional level in the complex value. Since this expands a potentially large number of values, you may experience a delay before the operation completes.
- **Collapse More** -- When a complex data value is selected, this menu item will collapse its display by one additional level.
- **Force Reload** -- This forces Wing IDE to reload the displayed value from the debug process. This is useful in cases where Wing is showing an evaluation error or when the debug program contains instances that implement `__repr__` or similar special methods in a way that causes the value to change when subjected to repeated evaluation.

#### **7.9.1.2. Filtering Value Display**

There are a number of ways in which the variable displays can be configured:

- Wing lets you prune the variable display area by omitting all values by type, and variables or dictionary keys by name. This is done by setting the two preferences, **Omit Types** and **Omit Names**.
- You can also tell Wing to avoid probing certain values by data type. This is useful to avoid attempting expansion of data values defined in buggy extension modules, which can lead to crashing of the debug process as the debugger invokes code that isn't normally executed. This preference is also respected during introspection of the runtime state for auto-completion and other features in the IDE. To add values to avoid, set preference **Do Not Expand**.
- Wing provides control over size thresholds above which values are considered too large to move from the debug process into the variable display area. Values found to be too large are annotated as **huge** in the variable display area and cannot be expanded further. The data size thresholds are controlled with preferences **Huge List Threshold** and **Huge String Threshold**.
- By default Wing will display small items on a single line in the variable display areas, even if they are complex types like lists and maps. The size threshold used for this is controlled with preference **Line Threshold**. If you want all values to be shown uniformly, this preference should be set to 0.

### 7.9.2. Problems Handling Values

The Wing debugger tries to handle debug data as gently as possible to avoid entering into lengthy computations or triggering errors in the debug process while it is packaging debug data for transfer. Even so, not all debug data can be shown on the display. This section describes each of the reasons why this may happen:

**Wing may time out handling a value** -- Large data values may hang up the debug server process during packaging. Wing tries to avoid this by carefully probing an object's size before packing it up. In some cases, this does not work and Wing will wait for the data for the duration set by the **Network Timeout** preference and then will display the variable value as `<network timeout during evaluate>`.

**Wing may encounter values too large to handle** -- Wing will not package and transfer large sequences, arrays or strings that exceed the size limits set by **Huge List Threshold** and **Huge String Threshold** preferences. On the debugger display, oversized sequences and arrays are annotated as **huge** and `<truncated>` is prepended to large truncated strings.

To avoid this, increase the value of the threshold preferences, but be prepared for longer data transfer times. Note that setting these values too high will cause the debugger to time out if the **Network Timeout** value isn't also increased.

**Wing may encounter errors during data handling** -- Because Wing makes assignments and comparisons during packaging of debug data, and because it converts debug data into string form, it may execute special methods such as `__cmp__` and `__str__` in your code. If this code has bugs in it, the debugger may reveal those bugs at times when you would otherwise not see them.

The rare worst case scenario is crashing of the debug process if flawed C or C++ extension module code is invoked. In this case, the debug session is ended.

More common, but still rare, are cases where Wing encounters an unexpected Python exception while handling a debug data value. When this happens, Wing displays the value as `<error handling value>`.

These errors are not reported as normal program errors in the Exceptions tool. However, extra output that may contain the exception being raised can be obtained by setting the **Debug Internals Log File** preference.

### Stored Value Errors

Wing remembers errors it encounters on debug values and stores these in the project file. These values will not be refetched during subsequent debugging, even if Wing is quit and restarted.

To override this behavior for an individual value, use the **Force Reload** item in the right-click context menu on a data value.

To clear the list of all errors previously encountered so that all values are reloaded, use the **Clear Stored Value Errors** item in the **Debug** menu. This operates only on the list of errors known for the current debug file, if a debug session is active, or for the main debug file, if any, when no debug process is running.

## 7.10. Debug Process I/O

While running under the Wing debugger, any output from `print` or any writes to `stdout` or `stderr` will be seen in the **Debug I/O** tool. This is also where you enter keyboard input, if your debug program requests any with `input()` or `raw_input()` or by reading from `stdin`.

The code that services debug process I/O does two things: (1) any waits on `sys.stdin` are multiplexed with servicing of the debug network socket, so that the debug process

remains responsive to Wing IDE while waiting for keyboard input, and (2) in some cases, I/O is redirected to another window.

For a debug process launched from within Wing, keyboard I/O always occurs either in the Debug I/O tool or in a new external console that is created before the debug process is started. This can be controlled as described in **External I/O Consoles**. Using an external console is recommended when printing very large amounts of output from a debug process.

Debug processes launched outside of Wing, using `wingdbstub`, always do their keyboard I/O through the environment from which they were launched (whether that's a console window, web server, or any other I/O environment).

### 7.10.1. External I/O Consoles

In cases where the debug process requires specific characteristics provided by the Windows Console or specific Linux/Unix shell, or to better handle very large amounts of debug process output, you can redirect debug I/O to a new external window using the **Use External Console** preference.

The most effective way to keep the external console visible after the debug process exits is to place a breakpoint on the last line of your program. Alternatively, enable the **External Console Waits on Exit** preference. However, this can result in many external consoles being displayed at once if you do not press enter inside the consoles after each debug run.

On Linux/Unix it is possible to select which console applications will be tried for the external console by altering the **External Consoles** preference.

Windows always uses the standard DOS Console that comes with your version of Windows.

### 7.10.2. Disabling Debug Process I/O Multiplexing

Wing alters the I/O environment in order to make it possible to keep the debug process responsive while waiting for I/O. This code mimics the environment found outside of the debugger, so any code that uses only Python-level I/O does not need to worry about this change of environment.

There are however several cases that can affect users that bypass Python-level I/O by doing C/C++ level I/O from within an extension module:

- Any C/C++ extension module code that does standard I/O calls using the C-level

`stdin` or `stdout` will bypass Wing's I/O environment (which affects only Python-level `stdin` and `stdout`). This means that waiting on `stdin` in C or C++ code will make the debug process unresponsive to Wing, causing time out and termination of the debug session if you attempt to Pause or alter breakpoints at that time. In this case, redirection of I/O to the debugger I/O tool and Debug Probe (in Wing Pro only) will also not work.

- On all platforms, calling C-level `stdin` from multiple threads in a multi-threaded program may result in altered character read order when running under the Wing debugger.
- When debugging on win32, calling C-level `stdin`, even in a single-threaded program, can result in a race condition with Wing's I/O multiplexer that leads to out-of-order character reads. This is an unavoidable result of limitations on multiplexing keyboard and socket I/O on this platform.

If you run into a problem with keyboard I/O in Wing's debugger, you should:

- 1) Turn off Wing's I/O multiplexer by setting the **Use `sys.stdin` Wrapper** preference to **False**.
- 2) Turn on the **Use External Console** preference (for details see **External I/O Consoles**)

Once that is done, I/O should work properly in the external console, but the debug process will remain unresponsive to Pause or breakpoint commands from Wing IDE whenever it is waiting for input, either at the C/C++ or Python level.

## 7.11. Debugging Multi-threaded Code

Wing's debugger can debug multi-threaded code, as well as single-threaded code. By default, Wing will debug all threads and will stop all threads if a single thread stops. If multiple threads are present in the debug process, the Stack Data tool (and in Wing Pro the Debug Probe and Watch tools) will add a thread selector popup to the stack selector.

Even though Wing tries to stop all threads, some may continue running if they do not enter any Python code. In that case, the thread selector will list the thread as running. It also indicates which thread was the first one to stop.

When moving among threads in a multi-threaded program, the Show Position icon shown in the toolbar during debugging (between the up/down frame icons) is a convenient way to return to the original thread and stopping position.

Whenever debugging threaded code, please note that the debugger's actions may alter the order and duration that threads are run. This is a result of the small added overhead, which may influence timing, and the fact that the debugger communicates with the IDE through a TCP/IP connection.

### Selecting Threads to Debug

Currently, the only way to avoid stopping all threads in the debugger is to launch your debug process from outside Wing, import `wingdbstub`, and use the debugger API's `SetDebugThreads()` call to specify which threads to debug. All other threads will be entirely ignored. This is documented in **Debugging Externally Launched Code** and the API is described in **Debugger API**

An example of this can be seen in the file `DebugHttpServer.py` that ships with Wing's support for Zope and Plone. To see this, unpack the WingDBG archive found inside the `zope` directory in your Wing installation.

Note, however, that specifying a subset of threads to debug may cause problems in some cases. For example, if a non-debugged thread starts running and does not return control to any other threads, then Wing's debugger will cease to respond to the IDE and the connection to the debug process will eventually be closed. This is unavoidable as there is no way to preemptively force the debug-enabled threads to run again.

## 7.12. Managing Exceptions

By default, Wing's debugger stops at exceptions when they would be printed by the Python interpreter or when they are logged with `logging.exception`. Wing will also stop on all `AssertionError` exceptions, whether or not they are printed or logged, since these usually indicate a program error even if they are handled.

The **Debugger > Exceptions** preference group can be used to control how Wing approaches exception reporting. This includes the following preferences.

### Exception Reporting Mode

The overall strategy for identifying and reporting exceptions is configured with the **Report Exceptions** preference. The following choices are available:

**When Printed** (default) -- The debugger will stop on exceptions at the time that they would have been printed out by the Python interpreter.

For code with catch-all exceptions written in Python, Wing may fail to report unexpected

exceptions if the handlers do not print the exception. In this case, it is best to rewrite the catch-all handlers as described in **Trouble-shooting Failure to Stop on Exceptions**.

In this exception handling mode, any code in **finally** clauses, **except** clauses that reraise the exception, and **with** statement cleanup routines will be executed before the debugger stops because they execute before the traceback is printed.

**Always Immediately** -- The debugger will stop at every single exception immediately when it is raised. In most code this will be very often, since exceptions may be used internally to handle normal, acceptable runtime conditions. As a result, this option is usually only useful after already running close to code that requires further examination.

**At Process Termination** -- In this case, the debugger will make a best effort to stop and report exceptions that actually lead to process termination. This occurs just before or sometimes just after the process is terminated. The exception is also printed to **stderr**, as it would be when running outside of the debugger.

When working with an **Externally Launched Debug Process**, the **At Process Termination** mode may not be able to stop the debug process before it exits, and in some cases may even fail to show any post-mortem traceback at all (except as printed to **stderr** in the debug process).

Similarly, when working with wxPython, PyGTK, and similar environments that include a catch-all exception handler in C/C++ code, the **At Process Termination** mode will fail to report any unexpected exceptions occurring during the main loop because those exceptions do not actually lead to process termination.

**Immediately if Appear Unhandled** -- The debugger will attempt to detect unhandled exceptions as they are raised in your debug process, making it possible to view the program state that led to the exception and to step through subsequently reached **finally** clauses. This is done by looking up the stack for exception handlers written in Python, and reporting only exceptions for which there is no matching handler.

Because of changes in the Python implementation, this mode no longer works in Python versions 2.7+ and 3.0+.

The **Immediately if Appear Unhandled** mode works well with wxPython, PyGTK, and in most other code where unexpected exceptions either lead to program termination or are handled by catch-all exception handlers written in C/C++ extension module code.

In some cases, Wing's unhandled exception detector can report normal handled exceptions that are not seen outside of the debugger. This occurs when the exceptions are handled in C/C++ extension module code. Wing can be trained to ignore these by checking the **Ignore this exception location** check box in the debugger's **Exception** tool. Ignored exceptions are still reported if they actually lead to program

termination, and your selection is remembered in your project file so only needs to be made once. Use **Clear Ignored Exceptions** from the **Debug** menu at any time to reset the ignore list to blank.

## Reporting Logged Exceptions

The **Report Logged Exceptions in When Printed Mode** preference controls whether exceptions that are not printed but that are logged with a call to `logging.exception` will be reported by the default **When Printed** exception reporting mode. This preference is ignored in other exception reporting modes.

## Exception Type Filters

The **Never Report** and **Always Report** preferences can be used to specify that certain exception types should never be reported at all, or always reported regardless of whether they are printed or logged. For example, by default Wing will never stop on `SystemExit` or `GeneratorExit` since these occur during normal program behavior, and Wing will always stop on `AssertionError` since this usually indicates a bug in code even if it is handled.

In some code, adding `NameError` or `AttributeError` to the **Always Report** list may help uncover bugs; however, this may not work if these are treated as normal expected exceptions by the authors of the code and there are too many such cases to ignore them with the **Ignore this exception location** checkbox in the **Exceptions** tool.



# Advanced Debugging Topics

This chapter collects documentation of advanced debugging techniques, including debugging externally launched code, and using Wing's debugger together with a debugger for C/C++ code.

See also the collection of **How-Tos** for tips of working with specific third party libraries and frameworks for Python.

## 8.1. Debugging Externally Launched Code

This section describes how to start debugging from a process that is not launched by Wing. Examples of debug code that is launched externally include CGI scripts or web servlets running under a web server and embedded Python scripts running inside a larger application.

### 8.1.1. Importing the Debugger

The following step-by-step instructions can be used to start debugging in externally launched code that is running on the same machine as Wing IDE:

- 1) Copy `wingdbstub.py` from the Wing IDE installation directory into the same directory as your debug program.
- 2) In some cases, you will also need to copy the file `wingdebugpw` from your **User Settings Directory** into the same directory as `wingdbstub.py`. This is needed when running the debug process as a different user or in a way that prevents the debug process from reading the `wingdebugpw` file from within your User Settings Directory.
- 3) At the point where you want debugging to begin, insert the following source code: `import wingdbstub` Depending on your code base, you may need to

be cautious about whether this statement is reached by multiple processes. If this happens, the first process will connect to Wing and the second one will fail to connect and continue running without debug. If you are debugging code in an embedded Python instance, see the notes in **Debugging Embedded Python Code**.

- 4) Make sure the Wing IDE preference **Enable Passive Listen** is turned on, to allow connection from external processes.
- 5) Set any required breakpoints in your Python source code.
- 6) Initiate the debug program from outside Wing IDE, for example with a page load in your web browser, if the program is a web app. You should see the status indicator in the lower left of the main Wing IDE window to yellow, red, or green, as described in **Debugger Status**. Make sure that you are running the Python interpreter without the `-O` option. The debugger will not work when optimization is turned on.
- 7) The debugger should stop at the first breakpoint or exception found. If no breakpoint or exception is reached, the program will run to completion, or you can use the **Pause** command in the **Debug** menu.

#### **Enabling Process Termination**

In some cases, you may wish to enable termination of debug processes that were launched from outside of Wing IDE. By default, Wing recognizes externally launched processes and disables process termination in these cases unless the **Kill Externally Launched** preference is enabled.

If you have problems making this work, try setting `kLogFile` variable in `wingdbstub.py` for log additional diagnostic information.

#### **Behavior on Failure to Attach to IDE**

Whenever the debugger cannot contact Wing IDE (for example, if the IDE is not running or is listening on a different port), the debug program will be run without debugging. This is useful since debug-enabled CGIs and other programs should work normally when Wing is not present. However, you can force the debug process to exit in this case by setting the `kExitOnFailure` flag in `wingdbstub.py`.

### **8.1.2. Debug Server Configuration**

In some cases you may also need to alter other preset configuration values at the start of `wingdbstub.py`. These values completely replace any values set in Wing's Project

or File Properties, which are relevant only when the debug program is launched from within Wing. The following options are available:

- The debugger can be disabled entirely with `kWingDebugDisabled=1`. This is equivalent to setting the `WINGDB_DISABLED` environment variable before launching the debug program.
- Set `kWingHostPort` to specify the network location of Wing IDE, so the debugger can connect to it when it starts. This is equivalent to setting the `WINGDB_HOSTPORT` environment variable before launching the debug program. The default value is `localhost:50005`. See section **Remote Debugging** for details if you need to change this value.
- You can control whether or not the debugger's internal error messages are written to a log file by setting `kLogFile`. Use `<stdout>`, `<stderr>`, or a file name. If the given file doesn't exist, it is created if possible. Note that using `<stderr>` may cause problems on Windows if the debug process is not running in a console. This is equivalent to setting the `WINGDB_LOGFILE` environment variable before launching the debug program (use a value of `-` to turn off logging to file).
- Set `kEmbedded` to 1 when debugging embedded scripts. In this case, the debug connection will be maintained across script invocations instead of closing the debug connection when the script finishes. When this is set to 1, you may need to call `wingdbstub.debugger.ProgramQuit()` before your program exits, or before it discards an instance of Python, in order to cleanly close the debug connection to the IDE. This is equivalent to setting the environment variable `WINGDB_EMBEDDED`.
- Set `kAttachPort` to define the default port at which the debug process will listen for requests to attach (available in Wing IDE Professional only). This is equivalent to setting the `WINGDB_ATTACHPORT` environment variable before launching the debug program. If this value is less than 0, the debug process will never listen for attach requests. If it is greater than or equal to 0, this value is used when the debug process is running without being in contact with Wing IDE, as might happen if it initially fails to connect to the above-defined host and port, or if the IDE detaches from the process for a period of time.
- Set `kPWFilePath` and `kPWFileName` to define the search path and file name used to find a `wingdebugpw` file for the debugger. The environment variables `WINGDB_PWFILEPATH` and `WINGDB_PWFILENAME` will override these settings. The file path should be a Python list of strings if set in `wingdbstub.py` or a list of directories separated by the path separator (`os.pathsep`) when sent by environment variable. The string `$<winguserprofile>` may be used to specify Wing's **User Settings Directory** for the user that the debug process is running as. The password file name is usually `wingdebugpw` but may be changed in cases where this naming is inconvenient.

- Optionally, set `WINGHOME`, which is the Wing IDE installation directory (or on OS X `Contents/MacOS` within Wing's `.app` folder). This is set up during installation, but may need to be altered if you are running Wing from source or copied the debugger binaries over from another machine.

Setting any of the above-described environment variable equivalents will override the value given in the `wingdbstub.py` file.

### 8.1.3. Debugger API

A simple API can be used to control debugging more closely, once you have imported `wingdbstub.py` the first time, as was describe. This is useful in cases where you want to be able to start and stop debugging on the fly several times during a debug run, for example to avoid debug overhead except within a small sub-section of your code. It can also be useful in embedded scripting environments, particularly in those that alter the thread state or discard and recreate the Python instance across invocations.

To use the API, you must first onfigure and import `wingdbstub.py` as described in section **Importing the Debugger**.

#### High-Level API

The `wingdbstub.Ensure(require_connection=1, require_debugger=1)` function may be used to ensure the debugger is running and connected to the IDE. If `require_connection` is true, `ValueError` will be raised if a connection to the IDE cannot be made. If `require_debugger` is true, `ValueError` will be raised if the debugger binaries cannot be found or the debugger cannot be started.

#### Low-Level API

After importing `wingdbstub`, the following calls may be made on `wingdbstub.debugger` to control the debugger:

- `StopDebug()` - Stop debugging completely and disconnect from Wing IDE. The debug program continues executing in non-debug mode and must be restarted to resume debugging.
- `StartDebug(stophere=0, connect=1)` -- Start debugging, optionally connecting back to the IDE and/or stopping immediately afterwards.
- `Break()` -- This pauses the free-running debug program on the current line, as if at a breakpoint.

- `ProgramQuit()` - This must be called before the debug program is exited if `kEmbedded` was set to 1 in `wingdbstub.py` or if `autoquit=0` in the preceding `StartDebug()` API call (if any). This makes sure the debug connection to the IDE is closed cleanly.
- `SetDebugThreads(threads={}, default_policy=1)` - This can be used in multi-threaded code to tell Wing's debugger which threads to debug. Pass in a dictionary that maps from thread id (as obtained from `thread.get_ident()`) to one of the following values: 0 to ignore the thread (do not debug it), or 1 to debug the thread and immediately stop it if any thread stops. The `default_policy` sets the action to take when a thread is not found in the thread map.
- `SuspendDebug()` - This will leave the connection to the debug client intact but disables the debugger so that connection overhead is avoided during subsequent execution. This should be used only to exempt a particular section of code from debug overhead. In most cases `StopDebug` is preferable.
- `ResumeDebug()` - This will resume debugging using an existing connection to Wing.

Here is a simple usage example:

```
import wingdbstub
a = 1 # This line is debugged
wingdbstub.debugger.SuspendDebug()
x = 1 # This is executed without debugging
wingdbstub.debugger.ResumeDebug()
y = 2 # This line is debugged
```

`SuspendDebug()` and `ResumeDebug()` can be called as many times as desired, and nested calls will be handled so that debugging is only resumed when the number of `ResumeDebug()` calls matches the number of `SuspendDebug()` calls.

#### 8.1.4. Debugging Embedded Python Code

When Python code is run by an interpreter embedded in a larger application, you may need to craft special code to make debugging work properly.

If the host application is simply creating a single Python instance and reusing it for all script invocations, in most cases setting `kEmbedded=1` in `wingdbstub.py` will suffice.

In certain cases where the host application is manually creating or altering the thread state for each invocation of a script, you may need to use code as follows to reset the debugger and connection for each script invocation:

```
import wingdbstub
wingdbstub.Ensure()
```

In other cases where the host application uses an entirely different Python instance for each invocation, you may need to arrange that the **Debugger API** function `ProgramQuit` is called before each instance is destroyed and may also want to leave `kEmbedded=0` in `wingdbstub.py`. In this case you may also need to unset the environment variable `WINGDB_ACTIVE` before importing `wingdbstub`, as this may be left in the environment by the host application and will prevent `wingdbstub` from initiating debug in the second or later Python instance.

## 8.2. Remote Debugging

Since remote debugging is fairly complicated to configure, we currently recommend using remote display of the IDE via X Windows (Linux/Unix) or Remote Desktop (Windows) when possible, instead of setting up the IDE on a separate host from the debug process.

When this is not an option, you can also ask the debugger to connect remotely over the network. In order to do this, take the following steps (see also **Remote Debugging Example**):

- (1) First set up Wing IDE to successfully accept connections from another process within the same machine, as described in section **Importing the Debugger**. You can use any Python script for testing this until you have values that work.
- (2) Optionally, alter the **Server Host** preference to the name or IP address of the network interface on which the IDE listens for debug connections. The default server is `None`, which indicates that the IDE should listen on all the valid network interfaces on the host.
- (3) Optionally, alter the preference **Server Port** to the TCP/IP port on which the IDE should listen for debug connections. This value may need to be changed if multiple copies of Wing IDE are running on the same host.
- (4) Set the **Allowed Hosts** preference to include the host on which the debug process will be run. For security purposes, Wing will reject connections if the host isn't included here.
- (5) Configure any firewall on the system that Wing IDE is running on to accept a connection on the server port from the system that the debug process will run on.
- (6) Next install Wing IDE on the machine on which you plan to run your debug program. Creating an entire Wing IDE installation is the easiest approach. Alternatives are to copy only the debug server code out of a Wing installation on the same type of OS or to

compile the debugger core from source code. For details, see **Installing the Debugger Core**.

(7) Next, transfer copies of all your debug code so that the source files are available on the host where Wing IDE will be running and at least the `*.pyc` files are available on the debug host.

During debugging, the client and server copies of your source files must match or the debugger will either fail to stop at breakpoints or stop at the wrong place, and stepping through code may not work properly.

Since there is no mechanism in Wing IDE for transferring your code, you need to use NFS, Samba, FTP or some other file sharing or file transfer mechanism to keep the remote files up to date as you edit them in Wing.

If files appear in different disk locations on the two machines, you will also need to set up a file location map, as described in **File Location Maps**.

(8) On your debug host, copy `wingdbstub.py` into the same directory as your source files and import it in your Python source as described in **Debugging Externally Launched Code**.

(9) If you didn't copy `wingdbstub.py` out of a complete installation of Wing IDE on the debug host, you will need to set `kWingHome` to match the location where you have copied the debug server code on your debug host.

(10) In `wingdbstub.py` on your debug host, set `kWingHostPort`. The host in this value must be the IP address of the machine where Wing IDE is running. The port must match the port configured with the **Server Port** preference on the host where Wing IDE is running.

(11) Then restart Wing and try running your program on the debug host. You should see the Wing IDE debugger status icon change to indicate that a debug process has attached.

If you have problems making this work, try setting `kLogFile` variable in `wingdbstub.py` for log additional diagnostic information.

### 8.2.1. File Location Maps

In cases where the full path to your source is not the same on both machines, you also need to set up a mapping that tells Wing where it can find your source files on each machine.

This is done with the **Location Map** preference, which lists corresponding local and remote directory locations for each remote host's dotted quad IP address.

Each host IP address in the location map is paired with one or more (**remote\_prefix**, **local\_prefix**) tuples. The remote file prefix will be a full path on the debug server's file system. The local file prefix is usually the full path of a local directory, though it may also be a file: url.

The best way to understand this is to look at the **Location Map Examples**.

When running Wing IDE on Windows XP, UNC formatted file names such as `\\machine\path\to\file` may be used. On other Windows systems, you must map remote drives to a drive letter such as `F:`. In cases where setting up a persistent drive mapping is a problem, use a `cmd.exe` script with a `net use` command to map the drive on demand.

Note that making symbolic links on the client or server will not work as an alternative to using this mapping. This is a side-effect of functionality in the debugger that ensures that debugging works right when symbolic links are present: Internally, source file names are always resolved to their actual full path location.

#### 8.2.1.1. File Location Map Examples

The best way to understand location mapping is to inspect a few examples.

##### Defaults Explained

The default value for the **Location Map** preference contains one entry for 127.0.0.1 where the mapping is set to **None** (in Python this is represented as `{'127.0.0.1':None}`). This is equivalent to the more verbose Python representation of `{'127.0.0.1':[(('/',, '')]}`. It converts full paths on the debug server to the client-side URLs without altering any part of the full path.

##### Two Linux/Unix Hosts

Here is an example setting for `debug.location-map` that would be used if running Wing on `desktop1` and debugging some code on `server1` with IP address 192.168.1.1:

```
debug.location-map={
    '127.0.0.1':None,
    '192.168.1.1':[( '/home/apache/cgi', '/svr1/home/apache/cgi' )]
}
```

In this example, the files located in `/home/apache/cgi` on `server1` are the same files seen in `/server1/home/apache/cgi` on `desktop1` because the entire file system on `server1` is being shared via NFS and mounted on `desktop1` under `/svr1`.



To enter this value in Preferences, you would add 192.168.1.1 as a new Remote IP Address and a single local/remote mapping pair containing /home/apache/cgi and /svr1/home/apache/cgi.

### IDE on Linux/Unix with Debug Process on Windows

If you are debugging between Windows and Linux or Unix, some care is needed in specifying the conversion paths because of the different path name conventions on each platform. The following entry would be used when running Wing IDE on a Linux/Unix host and the debug process on a Windows host with ip address 192.168.1.1:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[(r'e:\src', '/home/myuser/src')],
}
```

In this example the Linux/Unix directory /home/myuser is being shared via Samba to the Windows machine and mapped to the e: drive.

In the Preferences GUI, you would add 192.168.1.1 as a new Remote IP Address and a single local/remote mapping pair containing e:\src and /home/myuser/src.

### IDE on Windows with Debug Process on Linux/Unix

If running Wing IDE on a Windows host and the debug process on a Linux/Unix host with IP address 192.168.1.1, the following would be used instead for the same file locations:

```
debug.location-map={
  '127.0.0.1':None,
  '192.168.1.1':[( '/home/myuser/src', 'e:/src')],
}
```

Again, note the use of forward slashes in the URL even though the file is on a Windows machine.

In the Preferences GUI, you would add 192.168.1.1 as a new Remote IP Address and a single local/remote mapping pair containing /home/myuser/src and e:/src.

### Two Windows Hosts

If running Wing IDE on Windows and the debug process on another Windows machine with IP address 192.168.1.1, the following would be used:

```
debug.location-map={
```

```

    '127.0.0.1':None,
    '192.168.1.1':[(r'c:\src', 'e:/src')],
}

```

In this case, the host where Wing is running has mapped the entire remote (debug process) host's `c:` drive to `e:`.

In the Preferences GUI, you would add 192.168.1.1 as a new Remote IP Address and a single local/remote mapping pair containing `c:\src` and `e:/src`.

### Two Windows Hosts using UNC Share

A UNC style path name can be used on Windows XP as follows:

```

debug.location-map={
    '127.0.0.1':None,
    '192.168.1.1':[(r'c:\src', '\\server\share\dir')],
}

```

In this case, `c:src` on the remote host, where the debug process is running, can be accessed as `\servershare\dir` on the machine where Wing IDE is running.

In the Preferences GUI, you would add 192.168.1.1 as a new Remote IP Address and a single local/remote mapping pair containing `c:\src` and `\\server\share\dir`.

## 8.2.2. Remote Debugging Example

Here is a simple example that enables debugging a process running on a Linux/Unix host (192.168.1.200) using Wing IDE running on a Windows machine (192.168.1.210). This example is for wingdbstub users only. If you are using the WingDBG product to debug Zope code, please refer to the **Zope Debugging How-To** (also included in the WingDBG control panel's Help tab).

On the Windows machine, the following preferences must be specified:

- **Enable Passive Listen** should be checked
- **Server Host** should be set to **All Interfaces** (this is the default)
- **Server Port** should be set to 50005 (this is the default)
- **Allowed Hosts** should be altered by adding 192.168.1.200

On the Linux/Unix machine, the following value is needed in `wingdbstub.py`:

```
kWingHostPort='192.168.1.210:50005'
```

Once this is done and Wing has been restarted, you should be able to run code that imports `wingdbstub` on the Linux/Unix machine and see the debug connection establish on the Windows machine.

Then you will need to set up file sharing between the two machines (for example, using Samba) and will need to establish a location map in your Wing IDE preferences on the Windows machine.

If your source code on the Linux/Unix machine is in `/home/myuser/mysource` and you map `/home/myuser` to `e:` on the Windows machine, then you would enter this location map via the Preferences GUI by adding `192.168.1.200` as a new Remote Host IP and entering a single mapping pair with `/home/myuser/mysource` and `e:/mysource`.

See **Location Map Examples** for additional examples.

### 8.2.3. Installing the Debugger Core

When Wing is used to debug a Python program remotely, the Wing debugger core must be installed on the remote machine. The easiest way to do that is just to install Wing IDE there. If that is not possible, there are two options: (1) Copy just the debugger files from a Wing IDE installation on the same type of machine, or (2) compile the debugger core from sources (available for Wing IDE Professional only).

#### Copying from Wing IDE Installation

When copying from an existing Wing IDE installation on another machine, you will need to copy all of the following files and directories under `WINGHOME`, which is the Wing IDE installation directory or on OS X the `Contents/MacOS` directory within the `.app` folder:

~~

```
wingdbstub.py bin/wingdb.py bin/##/src/debug/tserver bin/##/src.zip/debug/tserver
(only Python 2.5) bin/##/opensource/schannel (Python versions other
than 2.5) bin/##/opensource.zip/schannel (only Python 2.5)
```

Replace `##` with each version Python you wish to debug under (for example, `2.5`). You can omit the directories for the versions that you are not using.

The directories within zip files (used only in Python 2.5 and later) can either be copied by moving the entire zip file or by creating a subset that contains only the necessary directories.

Be sure to copy these directories from a Wing installation on the same type of host, so that on Linux/Unix you include \*.so extension modules, on Windows \*.pyd extension modules, and so forth.

### Compiling from Source

On machines for which there is no Wing IDE installer, the debugger core can be installed from source code. This is only available to Wing IDE Professional customers, and requires signing a [non-disclosure agreement](#). The compilation instructions are located in build-files/README.DBG-SRC/txt in the source distribution that you will be provided with.

## 8.3. OS X Debugging Notes

### System-Provided Python

The copy of Python in /Library/Python on OS X does not include source files for the standard libraries, so Wing's editor will not offer autocompletion values for those modules. To work around this, use Python from within /Library/Frameworks/Python.frameworks instead or copy of Python installed from the standard source distribution.

### MacPorts Python

At least some versions of the MacPorts packaging of Python are known not to work with Wing's debugger because it contains an \_md5 module that won't load. To work around this, use a different distribution of Python instead.

### Debugging 32-bit Python on a 64-bit System

On 64-bit OS X systems, you can set up a shell script with the following contents and set it as the Python Executable in Project Properties, in order to facilitate debugging Python in 32-bit mode:

```
#!/bin/bash
arch -i386 python "$@"
```

This should only be necessary if your code needs 32-bit libraries. Wing's debugger works in either 64-bit or 32-bit mode.

## 8.4. Debugger Limitations

There are certain situations that the debugger cannot handle, because of the way the Python programming language works. If you are having problems getting the debugger to stop at breakpoints or to display source as you step through your code, one or more of these may apply.

Always read the **Trouble-shooting Failure to Debug** section first. If that fails to uncover your problem, refer to the following detailed documentation of debugger limitations (many of which are extremely rare and esoteric):

- (1) Your source files must be stored on disk and accessible to the IDE. If you are trying to debug code fragments, try writing them to disk temporarily and setting the `__file__` variable in the module name space before invoking Python's `exec` or `eval`. This will allow Wing's debugger to map code objects to the source you've temporarily written to disk.
- (2) Running without saving will lead to incorrect display of breakpoints and run position because the debug process runs against the on-disk version of the source file. Wing will indicate in the Messages tool and Stack Data status indicator that some files are out of `sync` so this case should only occur if you ignore its warnings.
- (3) You cannot run the debug program using the `-O` or `-OO` optimization options for the Python interpreter. This removes information about line numbers and source file names, making it impossible to stop at breakpoints or step through code.
- (4) There are several cases where Wing may fail to stop at breakpoints or exceptions, or may fail to find source files corresponding with breakpoints or exception points. All of these are caused by storage of incorrect file names in `*.pyc` files:

- Moving `*.pyc` files on disk after they are generated invalidates the file name stored in the file if it is a partial relative path. This happens if your `PYTHONPATH` or `sys.path` contains partial relative path names.
- A similar problem may result from use of `compileall.py` and some other utilities that don't record a correct filename in the `*.pyc` file.
- If you run the same code twice using different paths to the same working directory, as is possible on Linux/Unix with symbolic links, the file names left in `*.pyc` may contain a mix of each of these paths. If the symbolic link that was used is subsequently removed, some of the file names become invalid.

The fix for all of these problems is to remove the `*.pyc` files and let Python regenerate them from the corresponding `*.py` files with the correct file name information.

Hint: You can open \*.pyc files in most text editors to inspect the stored file names.

(5) For code that spends much of its time in C/C++ without calling Python at all, for example as in a GUI main loop, the debugger may not reliably stop at breakpoints added during a run session, and may not respond to Pause requests. See section **Debugging Non-Python Mainloops** for more information.

(6) You cannot use `pdb` or other debuggers in code that you are running within the Wing debugger. The two debuggers conflict because they attempt to use the same debugger hooks in the Python interpreter.

(7) If you override `__import__` in your code, you will break the debugger's ability to stop at breakpoints unless you call the original `__import__` as part of your code whenever a module is actually imported. If you cannot call the original `__import__` for some reason, it may be possible to instead use `wingdbstub` and then call `wingdbstub.debugger.NotifyImport(mod)` from your import handler (where `mod` is the module that was just imported).

(8) If you set `__file__` in a module's name space to a value other than its original, Wing will be unable to stop at breakpoints in the module and may fail to report exceptions to the IDE's user interface.

(9) If you use an extension module to call C/C++ level `stdio` calls instead of using the Python-level facilities, the debug process will remain unresponsive to Wing IDE while waiting for keyboard input, I/O redirection to the Debug Probe in Wing Pro will fail, and you may run into out-of-order character reads in some cases. Details can be found in **Debug Process I/O**.

(10) Using partial path names in module `__file__` attribute can in rare cases cause Wing to fail to stop on breakpoints and exceptions, to fail to display source files, or to confuse source files of the same name.

A partial path name may end up in `__file__` only when (a) invoking Python code with a partial path name, for example with `python myfile.py` instead of `python /path/to/myfile.py`, (b) sending partial path names into `exec`, (c) using partial path names in your `PYTHONPATH` or `sys.path`, or (d) using `compileall.py` or similar tool to compile modules with a partial path name.

Because Wing does everything possible to avoid this problem in practice, it actually only occurs in the following rare cases:

- When modules are loaded with partial path names and `os.chdir()` is called before debugging is started. This is only possible when using `wingdbstub` or otherwise starting debug after your debug process is started.
- When modules are loaded with partial path names and `os.chdir()`

is called after `wingdbstub.debugger.SuspendDebug()` and before `wingdbstub.debugger.ResumeDebug()`.

- When modules are loaded with partial path names and removed from `sys.modules` before the debugger is started or while debugging is suspended.
- When code objects are created on the fly using `compile()`, the C API, or the new module, a relative filename or an incorrect filename are used for the filename argument, and `os.chdir()` is called before the code is executed.

(11) Wing tries to identify when source code in the IDE matches or does not match the code that is running in the debug process. There are certain very rare cases where this will fail, which may lead to failure to stop on breakpoints and other problems even when files are identified by the IDE as being synchronized:

Using `execfile()`, `eval()`, or `exec` with a `globals` dict that contains `__file__` will cause Wing to incorrectly assert that the specified file has been reloaded. In practice, this scenario usually occurs when `execfile()` is called from the top level of a module, in which case the module is in fact being loaded or reloaded (so no mis-identification of module load status occurs). However, in cases where a module load takes a long time or involves a long-running loop at the top level, the `execfile()`, `eval()`, or `exec` may occur **after** edits to the module have been made and saved. In this case, Wing will mis-identify the module as having been reloaded with the new edits.

This problem can also be triggered if a `globals` with `__file__` is explicitly passed to `execfile()`, `eval()`, or `exec`. However, it will only occur in this case when the code object file name is `?`, and `locals` and `globals` dictionaries are the same, as they are by default for these calls.

(12) In very rare cases, when using the `wingdbstub.py`, if you set `sys.exitfunc` after debugging has been started, the IDE will time out on a broken network connection after the debug program exits on an exception. This only happens in some exception handling modes with exceptions that look like they will be handled because a `try/except` block is present that might handle the exception, but where the exception is not in the end handled and the debug program exits without calling `StopDebug()`. Work-arounds include setting `sys.exitfunc` before importing `wingdbstub.py` or adding a top-level `try/except` clause that always calls `StopDebug()` before exiting the debug program.

(13) Naming a file `<string>` will prevent the debugger from debugging that file because it is confused with the default file name used in Python for code that is not located in a file.

(14) The debugger may fail to step or start after stopping at a breakpoint if the floating point mode is set to single precision (24 bit) on Intel x86 and potentially other processors.

This is sometimes done by graphics libraries such as DirectX or by other code that optimizes floating point calculations.

(15) When using Stackless Python, overriding `stackless.tasklet.__call__` without calling the Wing debugger's `__call__` will break the debugger.



# Source Code Analysis

Wing's auto-completer, source assistant, source index menu, goto-definition capability, find uses, refactoring, and other features all rely on a central engine that reads and analyzes your source code in the background as you add files to your project or alter your code in the source code editor. This engine can also load and inspect extension modules used by your code, can make use of live runtime state when available in a debug process or in the integrated Python Shell, and can read user-provided interface description files.

## 9.1. How Analysis Works

In analysing your source, Wing will use the Python interpreter and `PYTHONPATH` that you have specified in your **Project Properties**. If you have indicated a main debug file for your project, the values from that file's properties are used; otherwise the project-wide values are used. Whenever any of these values changes, Wing will re-analyze some or all of your source code.

You can view the Python interpreter and `PYTHONPATH` that are being used by the source code analysis engine, by selecting the Show Analysis Stats item in the Source menu. The values shown in the resulting dialog window are read-only but may be changed by pushing the Settings button. See **Project-wide Properties** for details on changing these values.

Be aware that if you use multiple versions of the Python interpreter or different `PYTHONPATH` values for different source files in your project, Wing will analyse all files in the project using the one interpreter version and `PYTHONPATH` it finds through the main debug file or project-wide debug properties settings. This may lead to incorrect or incomplete analysis of some source, so it is best to use only one version of Python with each Wing IDE project file.

When Wing tries to find analysis information for a particular module or file, it takes the following steps:

- The path and same directory as the referencing module are searched for an importable module
- If the module is Python code, Wing statically analyses the code to extract information from it
- If the module is an extension module, Wing looks for a `*.pi` interface description file as described later in this section
- If the module cannot be found, Wing tries to import it in a separate process space in order to analyze its contents
- If a debug process is active, Wing tries to read relevant type information from the live runtime state associated with the source code

## 9.2. Static Analysis Limitations

The following are known limitations affecting features based on static source analysis:

- Argument number, name, and type is not determined for functions and methods in extension modules.
- Analysis sometimes fails to identify the type of a construct because Python code doesn't always provide clues to determine the data type.
- Types of elements in lists, tuples, and dictionaries are not identified.
- Analysis information may be out of date if you edit a file externally with another editor and don't reload it in Wing. See section **Auto-reloading Changed Files** for reload options.
- From time to time, as Python changes, some newer Python language constructs and possible type inferencing cases are not supported.

A good way to work around these limitations, when they arise, is to place a breakpoint in the code where you are working, run to it, and then auto-completion and other information presented by the IDE will be based on the actual runtime state rather than static analysis.

See **Helping Wing Analyze Code** for more information.

## 9.3. Helping Wing Analyze Code

There are a number of ways of assisting Wing's static source analyzer in determining the type of values in Python code.

### Using Live Runtime State

When a debug process is active, or when working in the **Python Shell**, Wing extracts relevant type information from the live runtime state associated with your Python code. Since this yields complete and correct type information even for code that Wing's static analysis engine cannot understand, it is often useful to run to a breakpoint before designing new code that is intended to work in that context. In the editor, the cog icon in the auto-completer indicates that type information was found in the live runtime state. In Wing IDE Professional, the **Debug Probe** can be used to immediately try out new code in the runtime environment for which it is being designed.

### Using `isinstance()` to Assist Analysis

One way to inform the code analysis facility of the type of a variable is to add an `isinstance` call in your code. For example `isinstance(obj, CMyClass)` or `assert isinstance(obj, CMyClass)` when runtime type checking is desired. The code analyzer will pick up on these and present more complete information for the asserted values.

In cases where doing this introduces a circular import, you can use a conditional to allow Wing's static analyser to process the code without causing problems when it is executed:

```
if 0:
    import othermodule
    assert isinstance(myvariable, othermodule.COtherClass)
```

In most code, a few `isinstance` calls go a long way to making code faster and easier to edit and navigate.

### Using \*.pi files to Assist Analysis

Wing's source analyser can only read Python code and does not contain support for understanding C/C++ extension module code other than by attempting to import the extension module and introspecting its contents (which yields only a limited amount of information and cannot determine argument number, name, or types). Also, since Python is a dynamic language, it is possible to craft code that Wing's static analysis engine cannot understand.

For both of these cases, it is possible to create a \*.pi (Python interface) file that describes the contents of a module. This file is simply a Python skeleton with the appropriate structure and call signature to match the functions, attributes, classes, and methods

defined in a module. Wing IDE will read this file and merge its contents with any information it can obtain through static analysis or by loading an extension module. In some cases, as for Python bindings for GUI and other toolkits, these `*.pi` files can be auto-generated from interface description files.

For a module imported as `mymodule`, the interface file is called `mymodule.pi`. Wing will search for `*.pi` files first in the same directory as it finds the Python module (or the extension module source code if it has not yet been compiled and the source code's directory is on your configured Python Path). If not found, Wing will look in the directory path set with the **Interfaces Path** preference. Next, Wing will look in the `resources/builtin-pi-files` directory within your Wing IDE installation. Finally, Wing will look in `resources/packages-pi-files`, which is used to ship some `*.pi` files for commonly used third party packages.

When searching on the interfaces path or in the `resources` directories, the top level of the directory is checked first for a matching `*.pi` file. Then, Wing tries looking in a sub-directory `##` named according to the major and minor version of Python being used with your source base, and subsequently in each lower major/minor version back to 2.0.

For example, if `c:\share\pi\pi-files` is on the interfaces path and Python 2.7 is being used, Wing will check first in `c:\share\pi\pi-files`, then in `c:\share\pi\pi-files\2.7`, then in `c:\share\pi\pi-files\2.6`, and so forth.

Example `*.pi` files used by Wing internally to produce autocompletion information for builtins can be seen in the directory `resources/builtin-pi-files` inside your Wing IDE installation. This also illustrates the above-described version number fallback mechanism.

In cases where Wing cannot find a `*.pi` at all, it will attempt to load the module by name (in a separate process space) so that it can introspect its contents. The results of this operation are stored in `pi-cache` within the **User Settings Directory** and used subsequently. This file is regenerated only if the `*.pyd` or `*.so` for the loaded module changes.

## 9.4. Analysis Disk Cache

The source code analyzer writes information about files it has recently examined into the Cache Directory that is listed in Wing's About box, which is accessed from the **Help** menu.

Cache size may be controlled with the **Max Cache Size** preference. However, Wing does not perform well if the space available for the cache is smaller than the space needed for

a single project's source analysis information. If you see excessive sluggishness, either increase the size of the cache or disable it entirely by setting its size to 0.

If the same cache will be used by more than one computer, make sure the clocks of the two computers are synchronized. The caching mechanism uses time stamps, and may become confused if this is not done.

The analysis cache may be removed in its entirety. Wing IDE will reanalyze your code and recreate the cache as necessary.



# Trouble-shooting Guide

This chapter describes what to do if you are having trouble installing or using Wing IDE.

We welcome feedback and bug reports, both of which can be submitted directly from Wing IDE using the Submit Feedback and Submit Bug Report items in the Help menu, or by emailing us at [support at wingware.com](mailto:support@wingware.com).

## 10.1. Trouble-shooting Failure to Start

If you are having trouble getting Wing to start at all, read through this section for information on diagnosing the problem.

**On OS X**, Wing requires that you install and launch an X11 Server before starting Wing IDE. If the launcher fails to start X11 or Wing, try starting X11 Server manually and then running `wing-personal4.1` from within the Wing IDE application folder (which can be entered using a terminal window in X11). See the **OS X How-To** for details.

**On Windows**, the user's temporary directory sometimes becomes full, which prevents Wing from starting. Check whether the directory contains more than 65,534 files. Some versions of Acrobat Reader will leave large numbers of lock files in this directory. These files are named `Acrxxxx.tmp`. Other applications may do this as well.

**On Fedora Core 5 and other Linuxes with SELinux**, Wing won't start because permissions are denied on one of the shared libraries needed by it. The solution is to go into `bin/2.4/external/pyscintilla2` and issue the following command:

```
chcon -t texrel_shlib_t _scintilla.so
```

**On Linux**, in some cases, Wing will not run with its own private GTK installation because of incompatibilities with the system. To test this, run Wing with the `--system-`

**gtk** command line option after making sure your Linux system has the GTK packages installed. If this works, you can set the **Use System Gtk** preference.

Note, however, that there are known problems running system-provided Qt emulation when using the system GTK option. Some of these themes contain bugs that can cause crashing. If you need to use the system GTK and experience crashes, we recommend using a theme other than a Qt theme.

**On Linux**, if Wing fails to start after the **Use System Gtk** preference has been set, use the **--private-gtk** command line option to get Wing running again so that the preference can be turned off.

**On Linux, OS X, or other posix systems**, in some cases when the `~/.cache` directory or the cache directory set by the `$XDG_CACHE_DIR` is located on an NFS or other remote file server, Wing can't obtain a lock on a database file. To use slower, dotfile locking set the **Use sqlite dotfile locking** preference to enabled or run Wing with the **--use-sqlite-dotfile-locking** command line option. Note that all Wing processes, regardless of the system they're running on, that use the same cache directory need to either use or not use dotfile locking.

**To rule out problems with a project file or preferences**, try renaming your **User Settings Directory** and restart Wing. If this works, you can copy over files from the renamed directory one at a time to isolate the problem -- or email support at wingware dot com for help.

**Under a Windows terminal server**, Wing may not be able to set up the environment variables it uses internally and will not start up. In this case, you can get Wing to start with the following commands:

```
set PYTHONOPTIMIZE=1
set PYTHONHOME=D:\Program Files\WingIDE\bin\PyCore
wing.exe
```

Alter PYTHONHOME according to the location at which you've installed Wing IDE.

**Constant Guard** from Comcast can prevent Wing IDE from starting without showing any dialog or message that it is doing so

**In other cases**, refer to **Obtaining Diagnostic Output**.

## 10.2. Issues on Microsoft Windows

Wing has a few problems and limitations on Microsoft Windows systems



1) The TortoiseHg shell extension and a few of the demo shell extension COM objects from `win32all` can cause Wing to crash if they are registered. The crash occurs when the file open, save, and add files to project dialog boxes are used. TortoiseHg may be removed via the Windows control panel. The demo extensions may be disabled by using ShellExView (<http://www.snapfiles.com/get/shellexview.html>) or a similar program to find and disable them. They can also be uninstalled by running the `.py` file with an `--unregister` argument.

2) The nVidia Desktop Manager may cause the system to freeze on some versions of Windows (apparently the card becomes very sluggish while the system CPU utilization remains near 0%). The problem appears more frequently when using Wing in multi-window modes but may occur in all cases. Disabling the manager prevents the freeze from occurring.

There may be other display issues (such as failure to draw window contents when un-minimizing from Windows task bar) specifically with some nVidia cards, even if the desktop manager is disabled.

3) Pasting will sometimes fail when remote desktop or another application that tracks the contents of the clipboard is used.

4) Windows drag-n-drop currently doesn't work for transferring text between Wing and other applications.

5) Wing won't be able to launch python on Vista if the python executable is configured to run as administrator and `wing.exe` is not. It is recommended that neither Wing nor python is set to run as administrator by default.

6) Changing the `os.environ['TZ']` value in a program will not affect the timezone used by the `time.localtime()` and other time functions, at least in Python 2.5. This is due to Windows C runtime library capturing the TZ value the first time a time related function is called; Wing's debugger calls `time.time()` before any code in the program being debugged is run. The workaround is to set TZ in the environment via the Project Properties dialog so the value is set before the process starts.

7) Some anti-virus / security programs can block access to the clipboard, but usually can be configured to allow cut and paste to work. Please contact Wingware support if you are having problems with cut or paste.

## 10.3. Issues on Linux

Wing has a few problems and limitations on Linux systems

1) Wing can consume all memory or crash if run with some versions of KDE or, more

specifically, the gtk theme engine that implements qt look and feel. To work around this, go to KDE's setting module and in the System Settings/Application Appearance/GTK+ Appearance module change from the default to another setting and restart Wing

2) For other persistent problems on a 64-bit Linux system, such as GUI update glitches or crashing, try installing the 32-bit compatibility libraries and 32-bit Wing IDE as documented at the end of **Linux Installation Detail**.

## 10.4. Trouble-shooting Failure to Debug

If you have trouble debugging with Wing IDE, select which of the following most closely describes the problem you are seeing:

- **Debugging fails to start**
- **Debugger doesn't stop on breakpoints**
- **Debugger doesn't stop on exceptions**
- **Debugger reports exceptions not seen outside Wing**

### 10.4.1. Failure to Start Debug

Wing may fail to start the debug process in certain cases. If this happens, it often helps to try debugging a small test such as the following:

```
print("test1")
print("test2")
```

Use the **Start / Continue** command from the Debug menu to cause Wing IDE to attempt to run only as far as the first line of your code. This rules out possible problems caused by specific code.

Then check through the following common problems. For information on obtaining additional information from the debug sub-system, refer to the **Diagnostic Output** section:

1) Wing's debugger uses a TCP/IP protocol to communicate with the IDE. Make sure that TCP/IP is installed and configured on your machine. If you are running a custom-built copy of Python, verify that the **socket** module is available.

2) If Wing says it can't find Python or if you've got multiple versions of Python on your system, make sure you've got your **Project Properties** set up to contain a valid interpreter (see Source / Show Python Environment menu item to verify that the right interpreter is being found).

3) Enter any necessary PYTHONPATH for your debug process in Project Properties if not already defined in the environment.

4) If you set PYTHONHOME or PYTHONPATH environment variables, these may cause the debug process to fail if they do not match the particular Python interpreter that Wing is launching. You can either change the interpreter used so it matches, or unset or alter these environment values from the outside or via Project Properties from the Project menu.

PYTHONHOME is a problem in all cases when it doesn't match the Python interpreter reported in the Source menu's Show Python Environment dialog.

PYTHONPATH is only a problem if it contains directories that are part of a Python installation. When this doesn't match the interpreter version, this leads to import errors because Python tries to import incompatible modules.

5) On Windows, check that you don't have Hummingbird Socks Client installed on your machine. Some versions and configurations of this product are known to incorrectly route network packets in such a way that slows down the Wing IDE debugger enough to make it time out during initialization.

6) All forms of the Python binary distribution (TAR, RPM, and Windows installer) are known to have problems when a newer version of Python is installed directly over an older one on disk.

In this case, most Python programs will appear to work fine outside of Wing IDE but will not work within the Wing IDE debugger. This occurs because the debug support code uses sockets and other functionality that is not necessarily exercised by your debug program outside of the Wing debugger.

If you try to run a debug session in Wing IDE and it fails, you may be having this problem. The following test script can be used to confirm that the problem exists in your Python installation:

```
import sys
print('sys.version =', sys.version)
print('sys.executable =', sys.executable)
print('sys.version_info =', sys.version_info)
import socket
print('socket =', socket)
print('socket._socket =', socket._socket)
```

```
import select
print('select =', select)
import cPickle
print('cPickle =', cPickle)
```

To solve this problem, try uninstalling Python, manually removing any remaining files, and installing again. Or install Python into a new location on disk.

Once this is done, be sure to confirm that Wing is configured to use the new Python installation from the Project Properties dialog in the Project menu and that the Show Python Environment item in the Source menu displays the correct interpreter.

7) Wing's debugger is unable to debug games written with pygame when they are running in full screen mode. Use window mode instead. This is a problem also for other IDEs; we have not yet investigated the cause.

### 10.4.2. Failure to Stop on Breakpoints or Show Source Code

The most common cause of failure to stop on breakpoints or to bring up source windows while stopping or stepping through code is a mismatch between the file name that is stored in the \*.pyc file and the actual location of the \*.py source file.

This can be caused by (1) not saving before you run in the debugger, (2) using partial path names on PYTHONPATH or when invoking a script from the command line (the partial path stored in the \*.pyc file may become invalid if current directory changes), (3) moving around the \*.pyc file after they are created, or (4) using `compileall.py` to create \*.pyc files from source. The easiest way to solve this is to use only full paths on PYTHONPATH and remove any suspect \*.pyc files.

Wing may fail to stop when debugging an application that gets invoked repeatedly in separate processes, for example a CGI script invoked multiple times from a browser as part of a page load. This is because the debugger can currently only debug one process at a time. If the debugger is already connected to one process, the second and later processes will not be debugged and thus may miss breakpoints.

Less common causes of this problem are (1) running Python with the `-O` optimization option, (2) running Python with `psyco` or other optimizer, (3) overriding the Python `__import__` routine, (4) adding breakpoints after you've started debugging an application that spends much of its time in C/C++ or other non-Python code, and (5) on win32, using symbolic links to directories that contain your source code files (Posix platforms handle symbolic links just fine).

For more information, see the **Debugger Limitations** section.

### 10.4.3. Failure to Stop on Exceptions

Failure to stop on exceptions is most commonly caused by the same factors that can cause **failure to stop on breakpoints**. The rest of this section covers additional possible causes of failure to stop on exceptions.

By default, Wing only stops on exceptions for which a traceback is printed when the code is run outside of the debugger. If your code runs within a catch-all try/except clause written in Python (as in some GUI main loops or in an environment like Zope), Wing may not report all exceptions encountered in your debug process.

In some cases, altering the **Exception Reporting** preference will work. In others, it may suffice to set a breakpoint in the top-level exception handler.

An alternative is to recode your app by adding the following code to catch-all exception handlers:

```
import os, sys
if 'WINGDB_ACTIVE' in os.environ:
    sys.excepthook(*sys.exc_info())
```

The above only works with the default exception handling configuration. If you are not using the **When Printed** exception handling mode (as set by the **Report Exceptions** preference) then the above will not cause the debugger to stop. In that case, the following variant can be used instead:

```
import os

# No handler when running in Wing's debugger
if 'WINGDB_ACTIVE' in os.environ:
    dosomething()

# Handle unexpected exceptions gracefully at other times
else:
    try:
        dosomething()
    except:
        # handler here
```

Note that environments such as wxPython, PyGTK, and others include catch-all handlers for unexpected exceptions raised in the main loop, but those handlers cause the exception traceback to be printed and thus will be reported correctly by Wing without any modification to the handler.

#### 10.4.4. Extra Debugger Exceptions

This section is only relevant if you have set the **Exception Reporting** preference to **Immediately if Appears Unhandled**.

When Wing's debugger is running in this exception handling mode, it sometimes appears to reveal bugs that are not seen when running outside of the debugger. This is a result of how this mode decides which exceptions should be shown to the user -- it is inspecting exceptions as they are raised and making decisions about whether or not the exception is unexpected or part of normal operation.

You can train Wing to ignore unwanted exception reports with the checkbox in the **Exceptions** tool.

You can also change the way Wing reports debug process exceptions with the **Exception Reporting** preference.

For more information, see **Managing Exceptions**.

### 10.5. Obtaining Diagnostic Output

Wing IDE and your debug code run in separate processes, each of which can independently be configured to collect additional diagnostic log information.

#### Diagnosing General IDE Problems

A quick way to diagnose problems seen while working with Wing IDE is to submit a bug report from the **Help** menu. Please include a description of the problem and check the **Include error log** checkbox so we can diagnose and fix the problem.

To diagnose other problems, such as failure to start, try looking at the file **error-log** in your **User Settings Directory**.

Alternatively, run `console_wing.exe` (on Windows) or `wing-personal4.1 --verbose` (on Linux/Unix and OS X) from the command line to display diagnostic output.

Email this output to [support at wingware.com](mailto:support@wingware.com) along with your system type and version, version of Wing IDE, version of Python, and any other potentially relevant details.

#### Diagnosing Debugger Problems

To diagnose debugger problems, set preference **Debug Internals Log File** to a value other than **No logging** and turn on preferences **Use External Console** and **External**

**Console Waits on Exit.** When you try again, Wing will display a debug console with diagnostics.

Alternatively, copy `wingdbstub.py` out of your Wing IDE installation, set `WINGDB_LOGFILE` environment variable to `<stderr>` or the name of a log file on disk (or alter `kLogFile` inside `wingdbstub.py`), turn on the **Enable Passive Listen** preference, and try launching the following script from the command line:

```
import wingdbstub
print("test1")
print("test2")
```

This prints diagnostic output that may be easier to capture in some cases.

Email this output to [support at wingware.com](mailto:support@wingware.com). Please include also the contents of the file `error-log` in your **User Settings Directory**, and also your system version, version of Wing IDE, version of Python, and any other potentially relevant details.

## 10.6. Speeding up Wing

Wing should present a responsive, snappy user interface even on relatively slow hardware. In some cases, Wing may appear sluggish:

- Try using a different Display Theme from preferences -- the pixmap manipulations in Wing's default themes sometimes fail to be accelerated on certain display hardware. Oddly, this seems worse on faster hardware than on slower hardware.
- If you have nVidia desktop manager, disable it for Wing.
- The first time you set up a project file, Wing analyzes all source files for the source code browser and auto-completion facilities. During this time, the browser's class-oriented views will display only the source constructs from files of which analysis information has already been obtained. The user interface may also appear to be sluggish and Wing will consume substantial amounts of CPU time.

To avoid this in subsequent sessions, Wing stores its source analysis information to disk in a cache within your **User Settings Directory**.

- In wxPython and other code that uses `from xxx import *` style imports, the auto-completer may initially be slow to appear if it needs to process many hundreds of symbols. This should only happen the first time it appears, however.

- On Windows, if Wing is started while operating via Remote Desktop Connection, performance is terrible, even after quitting the RDC session and working directly on the machine that is running Wing. However, if Wing is started on the machine on which it runs, performance is very lively on that machine and acceptable if switched to operating via RDC without quitting Wing.
- If you are displaying Wing remotely via X11, try turning off anti-aliased fonts by placing [this file](#) in `~/.fonts.conf` on the display machine and then restarting the X server.
- If you are working on a multi-core virtual machine and Wing runs slowly you may be able to improve performance by setting the processor affinity for Wing. This is done with `schedtool -a 0x1 -e wing-personal4.1` on Linux (the `schedtool` package needs to be installed if not already present) and with `START /AFFINITY 01 "Wing IDE" "C:\Program Files\Wing IDE 4.1\bin\wing.exe"` on Windows. Although Wing runs on only one core, this technique has been reported to improve performance.

## 10.7. Failure to Detect HTTP Proxy and Connect to wingware.com

Wing will try to open an http connection to `wingware.com` when you activate a license, check for product updates, or submit feedback or a bug report. If you are running in an environment with an http proxy, Wing will try to auto-detect your proxy settings. If this fails you will need to configure your proxy manually using Wing's **HTTP Proxy Server** preference. To determine the correct settings to use, ask your network administrator or see [how to determine proxy settings](#).

## 10.8. Trouble-shooting Failure to Open Filenames Containing Spaces

**On Windows:** When using Windows File Types or Open With to cause Python files to be opened with Wing, some versions of Windows set up the wrong command line for opening the file. You can fix this using *regedt32.exe*, *regedit.exe*, or similar tool to edit the following registry location:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Classes\Applications\wing.exe\shell\open\command
```



The problem is that the association stored there is missing quotes around the `%1` argument. It should instead read as follows:

```
"C:\Program Files\Wing IDE\bin\wing.exe" "%1" %*
```

**On Linux:** KDE's Konqueror has the same problem that file names passed on the command line to applications bound to a file type are not enclosed with quotes, so the command line is not parsed correctly. We do not currently have a work-around for this problem.

## 10.9. Trouble-shooting Failure to Print

This section provides some hints to get printing working if it doesn't work "out of the box".

### On Windows

Wing has trouble printing with some printer drivers. One known issue is failure to transfer the correct font to the printer. The symptom is correctly printed header and footer but gibberish in the body of the source code. The problem can be solved in the Advanced menu under Print Properties in Windows by changing TrueTypeFont from "substitute with device font" to "download as soft font".

### On Linux

For Python files, Wing prints PDF formatted output directly to the printer. This does not work on at least some Linux distributions and can be worked around by setting the **Print Spool Command** preference to `pdf2ps %s - | kprinter --stdin`.

Wing uses `kprinter` by default on Linux when it is present. Another problem on Linux occurs when using a buggy version of `kprinter`. To rule that out, try `pdf2ps %s - | lpr` or simply `lpr %s` instead for the **Print Spool Command** preference

Turning on the **Print Python as Text** preference may also solve some printing problems, although on some systems with plainer output for Python files. When this is enabled, Python files are also passed through the the command given in the **Text Print Cmd** preference instead of generating syntax highlighted PDF. In all cases, all non-Python files are passed through this command.



# Preferences Reference

This chapter documents the entire set of available preferences for Wing IDE Professional. Note that this includes preferences that are ignored and unused in Wing IDE Personal and Wing IDE 101.

Most preferences can be set from the **Preferences GUI** but some users may wish to build preference files manually to control different instances of Wing IDE (see details in **Preferences Customization**).

## User Interface

### Display Theme

Configures the overall display style, or theme, used by Wing IDE. Additional GTK2 themes may be downloaded from <http://art.gnome.org/themes> and placed into WINGHOME/bin/gtk-bin/share/themes or USER\_SETTINGS\_DIR/themes. These will be added to the choices below. However, only the pixbuf, metal, and redmond95 theme engines are supported.

Internal Name: `gui.display-theme`

Data Specification: [H20-gtk2-Sapphire, H20-gtk2-Emerald, H20-gtk2-Amber, AluminumAlloy-Toxic, Redmond95, Smooth-2000, H20-gtk2-Amythist, HighContrastLargePrint, AluminumAlloy-Cryogenic, HighContrast, AluminumAlloy-Volcanic, LowContrast, LargePrint, HighContrastLargePrintInverse, AluminumAlloy-Smog, HighContrastInverse, Smokey-Blue, Glider, Smooth-Sea-Ice, Default, Glossy P, Redmond, None, Smooth-Retro, Smooth-Desert, H20-gtk2-Ruby, LowContrastLargePrint, Black-Background, GnuBubble]

Default Value: None

### Display Language

The language to use for the user interface. Either the default for this system, or set to a specific supported language.

Internal Name: `main.display-language`

Data Specification: `[None, de, en, fr]`

Default Value: `None`

### **Display Font/Size**

The base font and size to use for the user interface's menus and labels

Internal Name: `gui.default-font`

Data Specification: `[None or <type str>]`

Default Value: `None`

### **Source Code Font/Size**

The base font and size to use for the source code editor, Python Shell, Debug Probe, Source Assistant, and other tools that display source code.

Internal Name: `edit.default-font`

Data Specification: `[None or <type str>]`

Default Value: `None`

### **Use System Gtk**

Use the system wide gtk library (requires gtk 2.2 or later). Wing comes with its own private copy of the gtk libraries for which it is built and tested. Use the system gtk option to better integrate with the gnome or other desktop environment, however on some systems this may result in random crashing or other bugs resulting from binary incompatibilities in library versions. This preference may be overridden on the command line with the `--system-gtk` and `--private-gtk` command line options.

Internal Name: `gui.use-system-gtk`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

- **Layout**

### **Windowing Policy**

Policy to use for window creation: combined-window mode places toolboxes into editor windows, separate-toolbox-window mode creates separate tool box windows, and one-window-per-editor mode also creates a new window for each editor.

Internal Name: `gui.windowing-policy`

Data Specification: `[combined-window, one-window-per-editor, separate-toolbox-window]`

Default Value: `combined-window`

### **Show Editor Notebook Tabs**

Controls whether or not Wing shows notebook tabs for switching between editors. When false, a popup menu is used instead.

Internal Name: `gui.use-notebook-editors`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

### **Enable Tooltips**

Controls whether or not tooltips containing help are shown when the mouse hovers over areas of the user interface.

Internal Name: `gui.enable-tooltips`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

## **• Toolbars**

### **Show Toolbar**

Whether toolbar is shown in any window.

Internal Name: `gui.show-toolbar`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

### **Toolbar Size**

Sets size of the toolbar icons. One of "small", "medium", "large", "xlarge", or use "default" to select the system-wide settings.

Internal Name: `gui.toolbar-icon-size`

Data Specification: `[medium, default, xlarge, text-height, large, small]`

Default Value: `small`

### **Toolbar Style**

Select style of toolbar icons to use. One of "icon-only", "text-only", "text-below", "text-right", or use "default" to select the system-wide settings.

Internal Name: `gui.toolbar-icon-style`

Data Specification: `[medium, default, xlarge, text-height, large, small]`

Default Value: `icon-only`

## **• Colors**

### **Text Selection Color**

The color used to indicate the current text selection on editable text.

Internal Name: `gui.text-selection-color`

Data Specification: `[None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]`

Default Value: `(253, 253, 104)`

### **Source Code Background**

Background color to use on the source editor, Python Shell, Debug Probe, Source Assistant, and other tools that display source code. Foreground colors for text may be altered automatically to make them stand out on the selected background color.

Internal Name: `edit.background-color`

Data Specification: `[None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]`

Default Value: `None`

### **Debugger Run Marker Color**

The color of the text highlight used for the run position during debugging

Internal Name: `debug.run-marker-color`

Data Specification: [None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]

Default Value: (255, 163, 163)

### **Debugger Run Marker Alpha**

Select transparency of the text highlight used for the run position during debugging

Internal Name: `debug.run-marker-alpha`

Data Specification: [None or <type int>]

Default Value: None

### **Syntax Formatting**

Formatting options for syntax coloring in editors. Colors are relative to a white background and will be transformed if the background color is set to a color other than white.

Internal Name: `.edit.syntax-formatting`

Data Specification: [dict; keys: <type str>, values: [dict; keys: [italic, back, fore, bold], values: [one of: None, <type str>, <boolean: 0 or 1>]]]

Default Value: {}

### **Highlight Builtins**

Highlight Python builtins

Internal Name: `edit.highlight-builtins`

Data Specification: <boolean: 0 or 1>

Default Value: True

## **• Keyboard**

### **Personality**

Selects editor personality

Internal Name: `edit.personality`

Data Specification: [normal, vi, eclipse, brief, emacs, visualstudio]

Default Value: `normal`

### **Tab Key Action**

Defines the action of the Tab key when it is bound to the tab-key command. Possible actions are: "Default for Personality" to emulate the chosen Keyboard Personality. "Indent To Match" to indent the current line or selected line(s) to match the computed indent level for this context, "Move to Next Tab Stop" to enter indentation characters so the caret reaches the next tab stop, "Indent Region" to increase the indentation of the selected line(s) one level, or "Insert Tab Character" to insert a Tab character (`chr(9)`). For Python files, "Smart Tab" is an option that varies the tab key action according to the location of the caret within the line.

Internal Name: `edit.tab-key-action`

Data Specification: `[dict; keys: <type str>, values: <type str>]`

Default Value: `{'*': '--default--', 'text/x-python': '--default--'}`

### **Smart Tab End of Line Indents**

Select type of indentation that Smart Tab will place at the end of a line.

Internal Name: `edit.smart-tab-eol-indents`

Data Specification: `[None, 1, 2, 3, 4]`

Default Value: `4`

### **Custom Key Bindings**

Override key bindings in the keymap. To enter the key, place focus on the entry area and type the key combination desired. The command is one of those documented in the user manual's Command Reference, or the name of any user scripts that have been loaded into the IDE. Leave the command name blank to remove the default binding for a key (this is useful when adding multi-key bindings that conflict with a default).

Internal Name: `gui.keymap-override`

Data Specification: `[dict; keys: <type str>, values: <type str>]`

Default Value: `{}`

### **Typing Group Timeout**

Sets the timeout in seconds to use for typing, after which keys pressed are considered a separate group of characters. This is used for typing-to-select on lists and in other



GUI areas. Before the timeout subsequent keys are added to previous ones to refine the selection during keyboard navigation.

Internal Name: `gui.typing-group-timeout`

Data Specification: `<type float>`, `<type int>`

Default Value: 1

## **VI Mode Ctrl-C/X/V**

Controls the behavior of the Ctrl-X/C/V key bindings in vi mode. Either always use these for cut/copy/paste, use them for vi native actions such as `initiate-numeric-repeat` and `start-select-block`, or use the default by system (clipboard on win32 and other commands elsewhere).

Internal Name: `vi-mode.clipboard-bindings`

Data Specification: `[other, clipboard, system-default]`

Default Value: `system-default`

- **Other**

## **Show Splash Screen**

Controls whether or not the splash screen is shown at startup.

Internal Name: `main.show-splash-screen`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

## **Auto-Focus Tools**

Controls whether to automatically move keyboard focus from the editor to tools when they are revealed.

Internal Name: `gui.auto-focus-tools`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

## **Case Sensitive Sorting**

Controls whether names are sorted case sensitively (with all caps preceding small letters) or case insensitively

Internal Name: `gui.sort-case-sensitive`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **Auto-Show Bug Report Dialog**

Whether the error bug reporting dialog (also available from the Help menu) is shown automatically when an unexpected exception is encountered inside Wing IDE.

Internal Name: `gui.show-report-error-dialog`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **Auto-check for Product Updates**

Automatically attempt to connect to wingware.com to check for updates once every day after Wing is started.

Internal Name: `main.auto-check-updates`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Show Support+Upgrades Reminders**

Show a reminder when Support+Upgrades for the active license is expired or will expire soon.

Internal Name: `main.monitor-support-upgrades`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

## **• Advanced**

### **Display Area**

Rectangle to use for the IDE work area on screen. All windows open within this area. Format is (x, y, width, height), or use `None` for full screen.

Internal Name: `gui.work-area-rect`

Data Specification: [None or [tuple length 4 of: <type int>, <type int>, <type int>, <type int>]]

Default Value: None

### Max Error Log Size

The number of bytes at which the error log file (USER\_SETTINGS\_DIR/error-log) is truncated. This file can be sent to technical support to help diagnose problems with the IDE.

Internal Name: `main.max-error-log-size`

Data Specification: [from 10000 to 10000000]

Default Value: 500000

### Key Map File

Defines location of the keymap override file. Use None for default according to configured editor personality. See the Wing IDE Manual for details on building your keymap override file -- in general this is used only in development or debugging keymaps; use the keymap-override preference instead for better tracking across Wing versions.

Internal Name: `gui.keymap`

Data Specification: [None or <type str>]

Default Value: None

### Messages

Controls the format and verbosity of messages shown to the user for each message domain in the message area. Each domain specifies the format (in Python 2.3 logging.Formatter format), and the minimum logging level that should be shown in the display. If a message domain is left unspecified, then the parent domain settings are used instead ("" is the parent of all domains).

Internal Name: `gui.message-config`

Data Specification: [dict; keys: [search, debugger, analysis, general, project, editor, scripts, browser], values: [tuple length 3 of: <type str>, [0, 40, 30], <type int>]]

Default Value: {'': ('%(message)s', 0, 100000)}

### Document Text Styles

Defines text styles used in data and document display. Each style is specified as a list of

(name, value) tuples. The names and values must be valid Pango text attribute names and values. To set default values that apply to all styles, use the “default” style name (for example adding (“size”, 14) changes default display size to 14 points. Note that size of menus, buttons, labels, and other basic GUI elements are set using system-wide theme configuration and not from this preference. The source editor is also configured separately.

Internal Name: `main.text-styles`

Data Specification: [dict; keys: [one of: <type str>, [admonition-title, danger, footnote, citation, admonition, calltip-doc, title-4, calltip-strong, caution, title-3, title-0, title-1, image-link, calltip-type, calltip-poc, hint, calltip-arg-current, tip, literal, note, field, emphasis, title-2, calltip-class-symbol, attention, calltip-def-symbol, link, strong, marked-list-items, calltip-def, list-items, default, docinfo-header, transition, calltip-arg, caption, warning, error, navigation-link, navigation]], values: [tuple of: [one of: [tuple length 2 of: [foreground], [None or <type str>]], [tuple length 2 of: [style], [None or [oblique, italic, normal]]], [tuple length 2 of: [justification], [None or [right, fill, center, left]]], [tuple length 2 of: [font\_desc], [None or <type str>]], [tuple length 2 of: [weight], [None or [one of: <type int>, [heavy, bold, ultrabold, normal, light, ultralight]]]], [tuple length 2 of: [right\_margin], [None or [1]]], [tuple length 2 of: [stretch], [None or [condensed, expanded, normal, semicondensed, extracondensed, extraexpanded, semiexpanded, ultracondensed, ultraexpanded]]], [tuple length 2 of: [strikethrough], [None or <boolean: 0 or 1>]], [tuple length 2 of: [rise], [None or [from -100000 to 100000]]], [tuple length 2 of: [variant], [None or [small-caps, normal]]], [tuple length 2 of: [underline], [None or [double, single, low, none]]], [tuple length 2 of: [ypad], [None or [1]]], [tuple length 2 of: [background], [None or <type str>]], [tuple length 2 of: [indent], [None or [1]]], [tuple length 2 of: [left\_margin], [None or [1]]], [tuple length 2 of: [font\_family], [None or <type str>]], [tuple length 2 of: [xpad], [None or [1]]], [tuple length 2 of: [size], [None or [one of: [from 0 to 1000000], [medium, x-large, xx-large, large, small, xx-small, x-small]]]]]]]]]

Default Value: {'calltip-strong': (('font\_family', 'sans'), ('weight', 'bold'), ('foreground', '#000066')), 'danger': (('background', '#ffffdd'),), 'footnote': (('weight', 'bold'),), 'navigation-link': (('foreground', '#909090'), ('style', 'italic'), ('weight', 'bold')), 'citation': (('weight', 'bold'),), 'admonition': (), 'list-items': (('xpad', '1'), ('ypad', '1')), 'title-4': (('size', 'small'), ('underline', 'single'), ('foreground', '#000066')), 'warn-

```

ing': (('background', '#ffffdd'),), 'caution': (('background',
'ffffdd'),), 'title-3': (('size', 'small'), ('weight', 'bold'),
('foreground', '#000066')), 'title-0': (('size', 'xx-large'),
('weight', 'bold'), ('foreground', '#000066')), 'title-1': (('size',
'large'), ('weight', 'bold'), ('foreground', '#000066')), 'image-
link': (), 'calltip-type': (('font_family', 'sans'),), 'calltip-poc':
(('font_family', 'sans'),), 'hint': (('background', '#ffffdd'),),
'admonition-title': (('weight', 'bold'),), 'tip': (('background',
'ffffdd'),), 'literal': (('foreground', '#227722'), ('weight',
'bold')), 'note': (), 'field': (('weight', 'bold'),), 'emphasis':
(('style', 'italic'),), 'calltip-class-symbol': (('font_family',
'sans'), ('weight', 'bold'), ('foreground', '#0000ff')), 'attention':
(('background', '#dddfff'),), 'calltip-def-symbol': (('font_family',
'sans'), ('weight', 'bold'), ('foreground', '#007f7f')), 'link': (('un-
derline', 'single'), ('foreground', '#3333ff')), 'strong': (('weight',
'bold'), ('foreground', '#000066')), 'marked-list-items': (('weight',
'bold'), ('foreground', '#ff3333')), 'calltip-def': (('font_family',
'sans'), ('weight', 'bold'), ('foreground', '#00007f')), 'calltip-
doc': (('font_family', 'sans'),), 'default': (), 'docinfo-header':
(('weight', 'bold'),), 'transition': (('justification', 'left'),),
'calltip-arg': (('font_family', 'sans'),), 'calltip-arg-current':
(('font_family', 'sans'), ('background', '#ffbbbb')), 'caption':
(('style', 'italic'),), 'error': (('background', '#ffdddd'),),
'title-2': (('size', 'medium'), ('weight', 'bold'), ('foreground',
'#000066')), 'navigation': (('foreground', '#909090'), ('style',
'italic'))}

```

## Files

### Default Directory Policy

Defines how Wing determines the starting directory to use when prompting for a file name: Either based on location of the resource at current focus, location of the current project home directory, the last directory visited for file selection, the current directory at startup (or selected since), or always the specific fixed directory entered here.

Internal Name: `main.start-dir-policy`

Data Specification: [tuple length 2 of: [current-project, current-directory, recent-directory, current-focus, selected-directory], <type str>]

Default Value: ('current-focus', '')

## Title Style

Format used for titles of source files: Use “basename” to display just the file name, “prepend-relative” to use partial relative path from the project file location, “append-relative” to append partial relative path from project file location after the base file name, “prepend-fullpath” to use full path, or “append-fullpath” to append fullpath after the base file name.

Internal Name: `gui.source-title-style`

Data Specification: [append-relative, basename, prepend-fullpath, append-fullpath, prepend-relative]

Default Value: `append-relative`

## Default Encoding

The default encoding to use for text files opened in the source editor and other tools, when an encoding for that file cannot be determined by reading the file. Other encodings may also be tried. This also sets the encoding to use for newly created files.

Internal Name: `edit.default-encoding`

Data Specification: [None or [Central and Eastern European iso8859-2, Japanese iso-2022-jp-2004, Hebrew cp856, Japanese euc-jp, Vietnamese cp1258, Greek cp1253, Baltic Languages cp1257, Korean johab, System default (UTF-8), Baltic Languages cp775, Japanese iso-2022-jp-ext, Korean iso-2022-kr, Icelandic cp861, Hebrew cp424, Cyrillic Languages cp1251, Turkish iso8859-9, Unicode (UTF-16, little endian) utf-16-le, Western European cp500, Chinese (PRC) gb18030, Greek cp875, Arabic cp864, Icelandic mac-iceland, Chinese (PRC) gbk, Turkish mac-turkish, Greek iso8859-7, Baltic Languages iso8859-13, Cyrillic Languages mac-cyrillic, Greek cp869, Japanese iso-2022-jp-1, Central and Eastern European cp852, Japanese iso-2022-jp-2, Chinese (ROC) big5, Urdu cp1006, Hebrew iso8859-8, Celtic Languages iso8859-14, Thai cp874, Cyrillic Languages cp855, Western European iso8859-15, Greek mac-greek, Western European cp1252, Ukrainian koi8-u, Hebrew cp1255, Danish, Norwegian cp865, Cyrillic Languages iso8859-5, Turkish cp1026, Western European mac-roman, Western European cp1140, Chinese (PRC) hz, Japanese shift-jisx0213, Portuguese cp860, Chinese (ROC) cp950, US, Canada, and Others cp037, Japanese shift-jis-2004, Turkish cp1254, Japanese iso-2022-jp-3, Hebrew cp862, Western European latin-1, Japanese euc-jisx0213, Unicode (UTF-16, big endian) utf-16-be, Japanese euc-jis-2004, None, Central and Eastern European cp1250, Baltic Languages iso8859-4, English ascii, Japanese shift-jis, Arabic iso8859-6, Canadian English/French cp863, Russian koi8-r, Japanese iso-2022-jp, Unicode (UTF-8) utf-8,

Greek cp737, Nordic Languages iso8859-10, Central and Eastern European mac-latin2, Chinese (PRC) gb2312, Unicode (UTF-7) utf-7, Arabic cp1256, Chinese (PRC) big5hkscs, Western European cp850, Esperanto and Maltese iso8859-3, Turkish cp857, Korean cp949, US, Australia, New Zealand, S. Africa cp437, Unicode (UTF-16) utf-16, Japanese cp932]]

Default Value: None

### **New File EOL**

Default end-of-line to use: One of “lf”, “cr”, or “crlf” for each entry. Note that Wing matches existing line endings in non-blank files and uses this preference only when a file contains no end-of-line characters.

Internal Name: `edit.new-file-eol-style`

Data Specification: [lf, cr, crlf]

Default Value: lf

### **New File Extension**

Default file extension for newly created files

Internal Name: `edit.new-file-extension`

Data Specification: <type str>

Default Value: .py

### **Auto-save Files**

Controls whether or not all edited files are autosaved before a debug run, before starting unit tests, or before a file or build process is executed.

Internal Name: `gui.auto-save-before-action`

Data Specification: <boolean: 0 or 1>

Default Value: 0

### **Max Recent Items**

Maximum number of items to display in the Recent menus.

Internal Name: `gui.max-recent-files`

Data Specification: [from 3 to 200]

Default Value: 20

**Maximum File Size (MB)**

Maximum size of files that Wing will try to open, in MB.

Internal Name: `gui.max-file-size`

Data Specification: [from 1 to 100000]

Default Value: 100

**Always Use Full Path in Tooltips**

Set to True to always show the full path of a file name in the tooltips shown from the editor tabs and file selection menus. When False, the configured Source Title Style is used instead.

Internal Name: `gui.full-path-in-tooltips`

Data Specification: <boolean: 0 or 1>

Default Value: True

- **File Types**

**Extra File Types**

This is a map from file extension or wildcard to mime type. It adds additional file type mappings to those built into Wing IDE. File extensions can be specified alone without dot or wildcard, for example “`xcf`” or using wildcards containing “`*`” and/or “`?`”, for example “`Makefile*`”. The mime type to use for Python files is “`text/x-python`”.

Internal Name: `main.extra-mime-types`

Data Specification: [dict; keys: <type str>, values: [text/x-smalltalk, text/x-sql, text/x-pov, text/x-ave, text/x-pl-sql, text/x-bash, text/x-lua-source, text/x-eiffel, text/x-vxml, text/xml, text/x-errorlist, text/x-caml, text/x-octave, text/x-erlang, text/x-php-source, text/x-cython, application/x-tex, text/x-dos-batch, text/x-bullant, text/x-baan, text/x-python, text/x-mako, text/x-mmixal, text/x-nncrontab, text/postscript, text/x-django, text/x-asn1, text/x-javascript, text/x-fortran, text/x-vhdl, text/x-escript, text/x-lisp, text/x-makefile, text/x-diff, text/x-ms-idl, text/x-cpp-source, text/x-asm, text/x-ruby, text/x-abaqus, text/x-ada, text/x-d, text/x-idl, text/x-nsis, text/x-scriptol, text/x-perl, text/x-java-source, text/x-docbook, text/x-rc, text/x-verilog, text/x-c-source, text/plain, text/x-spice, text/x-zopept, text/x-lout, text/x-matlab, text/x-inno-setup, text/html, text/x-forth, text/x-tcl, text/x-qss, text/x-vb-source, text/x-pascal, text/x-



```
yaml, text/x-conf, text/x-ms-makefile, text/x-properties, text/css,
text/x-r]]
```

Default Value: {}

## File Sets

Defines file sets by specifying filters to apply to file names for inclusion and exclusion from a larger set (such as scanned disk files or all project files).

Each file set is named and contains one list of inclusion patterns and one list of exclusion patterns. The patterns can be a wildcard on the file name, wildcard on a directory name, or a mime type name.

Only a single pattern needs to be matched for inclusion or exclusion. Exclusion patterns take precedence over inclusion patterns, so any match on an exclusion pattern will always exclude a file from the set. File sets are used in constraining search, adding project files, and for other operations on collections of files.

Internal Name: `main.file-sets`

Data Specification: [dict; keys: <type str>, values: [tuple length 2 of: [tuple of: [tuple length 2 of: [wildcard-filename, wildcard-directory, mime-type], <type str>]], [tuple of: [tuple length 2 of: [wildcard-filename, wildcard-directory, mime-type], <type str>]]]]

Default Value: {u'All Source Files': (((), (('wildcard-filename', '\*.o'), ('wildcard-filename', '\*.obj'), ('wildcard-filename', '\*.a'), ('wildcard-filename', '\*.lib'), ('wildcard-filename', '\*.so'), ('wildcard-filename', '\*.dll'), ('wildcard-filename', '\*.exe'), ('wildcard-filename', '\*.ilk'), ('wildcard-filename', '\*.pdb'), ('wildcard-filename', '\*.pyc'), ('wildcard-filename', '\*.pyo'), ('wildcard-filename', '\*.pyd'), ('wildcard-filename', '\*\$py.class'), ('wildcard-filename', 'core'), ('wildcard-filename', '\*.bak'), ('wildcard-filename', '\*.tmp'), ('wildcard-filename', '\*.temp'), ('wildcard-filename', '\*-old'), ('wildcard-filename', '\*.old'), ('wildcard-filename', '\*.wpr'), ('wildcard-filename', '\*.wpu'), ('wildcard-filename', '\*.zip'), ('wildcard-filename', '\*.tgz'), ('wildcard-filename', '\*.tar.gz'), ('wildcard-filename', '\*.dsp'), ('wildcard-filename', '\*.dsw'), ('wildcard-filename', '\*.sln'), ('wildcard-filename', '\*.suo'), ('wildcard-filename', '\*.vcproj'), ('wildcard-filename', '\*.user'), ('wildcard-filename', '\*.manifest'), ('wildcard-filename', '\*.ncb'), ('wildcard-filename', '\*.bsc'), ('wildcard-filename', '\*.sbr'), ('wildcard-filename',

```

 '*.log'), ('wildcard-filename', '*~'), ('wildcard-filename', '###'),
 ('wildcard-filename', '*.#*'), ('wildcard-filename', '*.svn-base'),
 ('wildcard-filename', '.coverage'), ('wildcard-directory', '__py-
 cache__'), ('wildcard-directory', '.bzip'), ('wildcard-directory',
 'CVS'), ('wildcard-directory', '.hg'), ('wildcard-directory', '.git'),
 ('wildcard-directory', '.svn'), ('wildcard-directory', '_svn'),
 ('wildcard-directory', '.xvpics'))), u'HTML and XML Files': (((('mime-
 type', 'text/html'), ('mime-type', 'text/xml'), ('mime-type', 'text/x-
 zope-pt')), (('wildcard-filename', '*~'), ('wildcard-filename', '###'),
 ('wildcard-filename', '*.#*'), ('wildcard-filename', '*.svn-base'),
 ('wildcard-filename', '.coverage'), ('wildcard-directory', '__py-
 cache__'), ('wildcard-directory', '.bzip'), ('wildcard-directory',
 'CVS'), ('wildcard-directory', '.hg'), ('wildcard-directory', '.git'),
 ('wildcard-directory', '.svn'), ('wildcard-directory', '_svn'),
 ('wildcard-directory', '.xvpics'))), u'C/C++ Files': (((('mime-type',
 'text/x-c-source'), ('mime-type', 'text/x-cpp-source')), (('wildcard-
 filename', '*~'), ('wildcard-filename', '###'), ('wildcard-filename',
 '*.#*'), ('wildcard-filename', '*.svn-base'), ('wildcard-filename',
 '.coverage'), ('wildcard-directory', '__pycache__'), ('wildcard-
 directory', '.bzip'), ('wildcard-directory', 'CVS'), ('wildcard-
 directory', '.hg'), ('wildcard-directory', '.git'), ('wildcard-
 directory', '.svn'), ('wildcard-directory', '_svn'), ('wildcard-
 directory', '.xvpics'))), u'Hidden & Temporary Files': (((('wildcard-
 filename', '*.o'), ('wildcard-filename', '*.obj'), ('wildcard-
 filename', '*.a'), ('wildcard-filename', '*.lib'), ('wildcard-
 filename', '*.so'), ('wildcard-filename', '*.dll'), ('wildcard-
 filename', '*.exe'), ('wildcard-filename', '*.ilk'), ('wildcard-
 filename', '*.pdb'), ('wildcard-filename', '*.pyc'), ('wildcard-
 filename', '*.pyo'), ('wildcard-filename', '*.pyd'), ('wildcard-
 filename', '*$py.class'), ('wildcard-filename', 'core'), ('wildcard-
 filename', '*.bak'), ('wildcard-filename', '*.tmp'), ('wildcard-
 filename', '*.temp'), ('wildcard-filename', '*-old'), ('wildcard-
 filename', '*.old'), ('wildcard-filename', '*.wpr'), ('wildcard-
 filename', '*.wpu'), ('wildcard-filename', '*.zip'), ('wildcard-
 filename', '*.tgz'), ('wildcard-filename', '*.tar.gz'), ('wildcard-
 filename', '*.ncb'), ('wildcard-filename', '*.bsc'), ('wildcard-
 filename', '*.sbr'), ('wildcard-filename', '*~'), ('wildcard-filename',
 '###'), ('wildcard-filename', '*.#*'), ('wildcard-filename', '*.svn-
 base'), ('wildcard-filename', '.coverage'), ('wildcard-directory',
 '__pycache__'), ('wildcard-directory', '.bzip'), ('wildcard-directory',
 'CVS'), ('wildcard-directory', '.hg'), ('wildcard-directory', '.git'),
 ('wildcard-directory', '.svn'), ('wildcard-directory', '_svn'),
 ('wildcard-directory', '.xvpics')), ()), u'Python Files': (((('mime-
 type', 'text/x-python'), ('mime-type', 'text/x-cython')), (('wildcard-

```

```
filename', '*~'), ('wildcard-filename', '###'), ('wildcard-filename',
'.#*'), ('wildcard-filename', '*.svn-base'), ('wildcard-filename',
'.coverage'), ('wildcard-directory', '__pycache__'), ('wildcard-
directory', '.bzip'), ('wildcard-directory', 'CVS'), ('wildcard-
directory', '.hg'), ('wildcard-directory', '.git'), ('wildcard-
directory', '.svn'), ('wildcard-directory', '_svn'), ('wildcard-
directory', '.xvpics'))))}
```

- **Reloading**

### **External Check Freq**

Time in seconds indicating the frequency with which the IDE should check the disk for files that have changed externally. Set to 0 to disable entirely.

Internal Name: `cache.external-check-freq`

Data Specification: `<type float>`, `<type int>`

Default Value: 5

### **Reload when Unchanged**

Selects action to perform on files found to be externally changed but unaltered within the IDE. One of “auto-reload” to automatically reload these files, “request-reload” to ask via a dialog box upon detection, “edit-reload” to ask only if the unchanged file is edited within the IDE subsequently, or “never-reload” to ignore external changes (although you will still be warned if you try to save over an externally changed file)

Internal Name: `cache.unchanged-reload-policy`

Data Specification: `[never-reload, auto-reload, request-reload, edit-reload]`

Default Value: `auto-reload`

### **Reload when Changed**

Selects action to perform on files found to be externally changed and that also have been altered in the IDE. One of “request-reload” to ask via a dialog box upon detection, “edit-reload” to ask if the file is edited further, or “never-reload” to ignore external changes (although you will always be warned if you try to save over an externally changed file)

Internal Name: `cache.changed-reload-policy`

Data Specification: `[never-reload, request-reload, edit-reload]`

Default Value: `request-reload`

- **Projects**

**Auto-reopen Last Project**

Controls whether most recent project is reopened at startup, in the absence of any other project on the command line.

Internal Name: `main.auto-reopen-last-project`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

**Close Files with Project**

Controls whether any files open in an editor are also closed when a project file is closed

Internal Name: `proj.close-also-windows`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

**Open Projects as Text**

Controls whether project files are opened as project or as text when opened from the File menu. This does not affect opening from the Project menu.

Internal Name: `gui.open-projects-as-text`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

**Confirm Drag Copy/Move**

Controls whether or not the IDE will confirm file copy/move operations initiated by dragging items around on the Project view.

Internal Name: `proj.confirm-file-drags`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

- **External Display**

**File Display Commands**

Posix only: The commands used to display or edit local disk files selected from the Help menu or project files selected for external display. This is a map from mime type to a list of display commands; each display command is tried in order of the list until one works. The mime type “\*” can be used to set a generic viewer, such as a web browser. Use %s to place the file name on the command lines. If unspecified then Wing will use the configured URL viewer in the environment (specified by BROWSER environment variable or by searching the path for common browsers). On Windows and OS X, the system-wide configured default viewer for the file type is used instead so this preference is ignored.

Internal Name: `gui.file-display-cmds`

Data Specification: `[dict; keys: <type str>, values: [list of: <type str>]]`

Default Value: `{}`

## Url Display Commands

Posix only: The commands used to display URLs. This is a map from protocol type to a list of display commands; each display command is tried in order of the list until one works. The protocol “\*” can be used to set a generic viewer, such as a multi-protocol web browser. Use %s to place the URL on the command lines. If unspecified then Wing will use the configured URL viewer in the environment (specified by BROWSER environment variable or by searching the path for common browsers). On Windows and OS X, the system-wide configured default web browser is used instead so this preference is ignored.

Internal Name: `gui.url-display-cmds`

Data Specification: `[dict; keys: <type str>, values: [list of: <type str>]]`

Default Value: `{}`

# Editor

## Error Indicators

Controls whether Wing will show error and/or warning indicators on the editor as red and yellow underlines. When shown, hovering the mouse over the indicator shows the error or warning detail in a tooltip.

Internal Name: `edit.error-display`

Data Specification: [show-errors, show-none, show-all]

Default Value: show-all

### **Show Line Numbers**

Shows or hides line numbers on the editor.

Internal Name: edit.show-line-numbers

Data Specification: <boolean: 0 or 1>

Default Value: 0

### **Show Whitespace**

Set to true to show whitespace with visible characters by default

Internal Name: edit.show-whitespace

Data Specification: <boolean: 0 or 1>

Default Value: 0

### **Show EOL**

Set to true to show end-of-line with visible characters by default

Internal Name: edit.show-eol

Data Specification: <boolean: 0 or 1>

Default Value: 0

### **Split Reuse Policy**

Policy for reusing splits in editors when new files are opened: Either always open in current split, reuse already visible editor falling back on current split, reuse already visible editor falling back on adjacent split, or always open in an adjacent split. This only has an effect when more than one editor split is visible.

Internal Name: gui.split-reuse-policy

Data Specification: [current, reuse-adjacent, reuse-current, adjacent]

Default Value: current

### **Strip Trailing White Space**

Controls whether to automatically strip trailing white space in the editor. May be enabled for any file or only files that are part of the current project.

Internal Name: `main.auto-rstrip-on-save`

Data Specification: [tuple length 2 of: [disabled, on-save-project, on-save], <type str>]

Default Value: `disabled`

### **Input Method**

Input method used for typing characters. This is important primarily for non-Western European languages.

Internal Name: `edit.gtk-input-method`

Data Specification: []

Default Value: `default`

### **Block Comment Style**

Style of commenting to use when commenting out blocks of Python code.

Internal Name: `gui.block-comment-style`

Data Specification: [indented, block]

Default Value: `indented`

### **Scroll Past End**

Set this to allow scrolling the editor past the last line.

Internal Name: `edit.scroll-past-end`

Data Specification: <boolean: 0 or 1>

Default Value: `True`

- **Caret**

### **Caret Width**

Width of the blinking insertion caret on the editor, in pixels. Currently limited to a value between 1 and 3.

Internal Name: `edit.caret-width`

Data Specification: [from 1 to 3]

Default Value: 1

### **Caret Line Highlight**

Selects whether to highlight the line the caret is currently on. When enabled, a highlight color and alpha (transparency) can be set.

Internal Name: `edit.caret-line-highlight`

Data Specification: [None or [tuple length 2 of: [None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]], <type int>]]

Default Value: None

### **Caret Flash Rate (ms)**

Sets the time in milliseconds between showing and hiding the caret when it is flashing; use 0 to disable flashing entirely

Internal Name: `edit.caret-flash-rate`

Data Specification: [from 0 to 2000]

Default Value: 500

## **• Indentation**

### **Use Indent Analysis**

Select when to use indent analysis (examination of current file contents) in order to determine tab size and indent size. Either always in all files, only in Python files, or never.

Internal Name: `edit.use-indent-analysis`

Data Specification: [always, never, python-only]

Default Value: always

### **Default Tab Size**

Set size of tabs in the editor.

Internal Name: `edit.tab-size`

Data Specification: [from 0 to 80]



Default Value: 8

### **Default Indent Size**

Sets size of an indent, in spaces, used in new files. This is overridden in non-empty files, according to the actual contents of the file. In files with tab-only indentation, this value may be modified so it is a multiple of the configured tab size. Use the Indentation Manager to alter indentation in existing files.

Internal Name: `edit.indent-size`

Data Specification: [from 0 to 80]

Default Value: 4

### **Default Indent Style**

Set the style of indentation used in new files. This is overridden in non-empty files, according to the actual contents of the file. Use the Indentation Manager to alter indentation in existing files. Indentation style choices are “tabs-only” for tabs only, “spaces-only” for spaces only, or “mixed” to use a tab whenever tab-size spaces have been seen

Internal Name: `edit.indent-style`

Data Specification: [mixed, spaces-only, tabs-only]

Default Value: spaces-only

### **Auto Indent**

Controls when Wing automatically indents when return or enter is typed.

Internal Name: `edit.auto-indent`

Data Specification: [0, 1, blank-only]

Default Value: 1

### **Adjust Indent After Paste**

Controls whether Wing automatically adjusts indents after multi-line text is pasted. When enabled, a single undo will remove any alterations in indentation.

Internal Name: `edit.adjust-indent-after-paste`

Data Specification: <boolean: 0 or 1>

Default Value: True

### **Show Indent Guides**

Set to true to show indent guides by default

Internal Name: `edit.show-indent-guides`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **Show Python Indent Warning Dialog**

Set to show a warning dialog when opening a Python file that contains potentially problematic indentation: Either inconsistent and possibly confusing indentation, a mix of indent styles in a single file, or mixed tab and space indentation (which is not recommended for Python).

Internal Name: `edit.show-python-indent-warnings`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Show Override Warning Dialog**

Show indent mismatch warning dialog when user selects an indent style that is incompatible with existing file content. This only applies to non-Python files since Wing disallows overriding the indent style in all Python files.

Internal Name: `edit.show-non-py-indent-warning`

Data Specification: `<boolean: 0 or 1>`

Default Value: True

## **• Line Wrapping**

### **Wrap Long Lines**

Set to true to wrap long source lines on the editor display.

Internal Name: `edit.wrap-lines`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **Edge Markers**

Tuple that defines how edge markers are shown: (mode, column, color) where mode is 0 to turn off markers, 1 to show a line, or 2 to highlight text that extends past the edge;

column is the column at which to draw the marker, if on; and color is the color for the marker (r,g,b) tuple with values from 0x00 to 0xff: (0xff,0xff,0xff) is white.

Internal Name: `edit.show-edge-markers`

Data Specification: [tuple length 3 of: [0, 1, 2], [from 0 to 10000], [None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]]

Default Value: (0, 80, (251, 8, 8))

### **Reformatting Wrap Column**

Column at which text should be wrapped by commands that automatically rearrange text

Internal Name: `edit.text-wrap-column`

Data Specification: `<type int>`

Default Value: 77

### **• Occurrences**

#### **Highlight Occurrences**

Selects when to automatically highlight other occurrences of the current selection on the editor

Internal Name: `edit.highlight-occurrences`

Data Specification: [always, never, words]

Default Value: words

#### **Match Case**

Disable to allow occurrences highlighting also where case does not match.

Internal Name: `edit.match-case-occurrences`

Data Specification: `<boolean: 0 or 1>`

Default Value: True

#### **Occurrences Indicator Style**

The style of indicator to use for highlighting other occurrences of the current selection on the editor.

Internal Name: `edit.occurrence-indicator-style`

Data Specification: `[box, block]`

Default Value: `box`

- **Folding**

### **Enable Folding**

Set to true to enable structural folding on source, false to disable

Internal Name: `edit.enable-folding`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

### **Line Mode**

Set to “above-expanded”, “below-expanded”, “above-collapsed”, “below-collapsed”, or “none” to indicate where fold lines are shown and whether they are above or below the line where the fold point is located.

Internal Name: `edit.fold-line-mode`

Data Specification: `[above-collapsed, above-expanded, none, below-collapsed, below-expanded]`

Default Value: `below-collapsed`

### **Indicator Style**

Set to 0 to use arrow indicators, 1 to use plus/minus indicators, 2 to rounded tree indicators, and 3 to use square tree indicators.

Internal Name: `edit.fold-indicator-style`

Data Specification: `[from 0 to 3]`

Default Value: `1`

### **Fold Trailing White Space**

Controls whether or not trailing white space after a block of code is folded up along with the block, for a more compact folded display.

Internal Name: `edit.fold-trailing-whitespace`

Data Specification: <boolean: 0 or 1>

Default Value: 1

- **Auto-completion**

### **Auto-show Completer**

Controls whether or not the completer is always shown automatically during typing, never auto-shown, or shown only after a certain number of characters are in the completion fragment. When auto-show is disabled, the auto-completer can still be shown on demand with the Show Completer item in the Source menu.

Internal Name: `edit.autocomplete-autoshow-option`

Data Specification: [always, never]

Default Value: always

### **Auto-completer Height**

The maximum number of lines to show in the auto-completer at once.

Internal Name: `edit.autocompleter-height`

Data Specification: <type int>

Default Value: 7

### **Auto-complete Delay (sec)**

Delay in seconds from last key press to wait before the auto-completer is shown. If 0.0, the auto-completer is shown immediately.

Internal Name: `edit.autocomplete-delay`

Data Specification: <type int>, <type float>

Default Value: 0.0

### **Auto-complete Timeout**

Timeout in seconds from last key press after which the auto-completer is automatically hidden. If 0.0, the auto-completer does not time out.

Internal Name: `edit.autocomplete-timeout`

Data Specification: <type int>, <type float>

Default Value: 0

### **Completion Keys**

Controls which keys will enter selected completion value into the editor. Shift or Ctrl click to select multiple items.

Internal Name: `edit.autocomplete-keys`

Data Specification: [tuple of: [f1, f3, return, space, period, bracketleft, tab, f12, colon, f10, parenleft]]

Default Value: ['tab']

### **Completion Mode**

Selects how completion is done in the editor: Either insert the completion at the cursor, replace any symbols that match the leading part of the completion (and insert in other cases), or replace any existing symbol with the new symbol.

Internal Name: `edit.autocomplete-mode`

Data Specification: [replace-matching, insert, replace]

Default Value: `insert`

### **Case Insensitive Matching**

Controls whether matching in the completer is case sensitive or not. The correct case is always used when a completion is chosen.

Internal Name: `edit.autocomplete-case-insensitive`

Data Specification: <boolean: 0 or 1>

Default Value: `True`

### **Include Snippets in Completer**

Whether or not to include code snippets in the auto-completer.

Internal Name: `edit.snippets-in-autocompleter`

Data Specification: <boolean: 0 or 1>

Default Value: `True`

### **Python Turbo Mode (Experimental)**

When enabled, the Python auto-completer enters the completion automatically whenever

a key other than a valid symbol name key is pressed. When disabled, only the configured completion keys enter the completion into the editor.

Internal Name: `edit.autocomplete-turbo-mode`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **Non-Python Completion**

Controls whether or not use the completer in non-Python files, where it uses a simple word list generated from the existing contents of the file. If enabled, the number of characters required before the completer is shown may be specified here. This value overrides any character threshold set above.

Internal Name: `edit.autocomplete-non-python-option`

Data Specification: `[always, never]`

Default Value: 3

### **Non-Python Word Size**

Sets the minimum size of words to add to the completion list for non-Python files. This affects only words found in the file, and not words included because they are keywords for that file type.

Internal Name: `edit.autocomplete-non-python-word-size`

Data Specification: `<type int>`

Default Value: 4

## **• Printing**

### **Print Font**

(Posix only) Set the font name used to print Python files. One of Courier, Helvetica, or Times-Roman.

Internal Name: `edit.print-font`

Data Specification: `[Times-Roman, Helvetica, Courier]`

Default Value: Courier

### **Print Size**

(Posix only) Set the font size used to print Python files.

Internal Name: `edit.print-size`

Data Specification: [from 0 to 120]

Default Value: 10

## **Paper**

(Posix only) Set the paper size for printing. One of Letter, Legal, A3, A4, A5, B4, or B5

Internal Name: `edit.print-paper`

Data Specification: [A3, A5, Legal, Letter, A4]

Default Value: `Letter`

## **Print Spool Cmd**

(Posix only) Sets the command used to spool output produced by Wing's printing facility. Format is text with embedded %s to indicate where the printed output file's name should be inserted. Set to None to use internal defaults. If the default is not working for you and your system does not accept PDF files for printing, try "pdf2ps %s - | kprinter --stdin". To rule out problems with buggy versions of kprinter, try "pdf2ps %s - | lpr" or simply "lpr %s" instead.

Internal Name: `edit.print-spool-cmd`

Data Specification: [one of: None, <type str>]

Default Value: `None`

## **Print Python as Text**

(Posix only) Set to true to print Python files faster but without syntax highlighting. Otherwise, the internal Python pretty printing facility is used.

Internal Name: `edit.print-python-as-text`

Data Specification: <boolean: 0 or 1>

Default Value: 0

## **Text Print Cmd**

(Posix only) Sets the command that is issued to print non-Python text files. Format is text with embedded %s to indicate where the printed file's name should be inserted



Internal Name: `edit.text-print-cmd`

Data Specification: `<type str>`

Default Value: `enscript -E %s`

- **Advanced**

## Brace Highlighting

Set to true to automatically highlight the matching braces next to the cursor or as they are typed.

Internal Name: `edit.auto-brace-match`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

## Transient Threshold

Maximum number of transient (non-sticky) editors to keep open at one time, in addition to any that are visible on screen

Internal Name: `gui.max-non-sticky-editors`

Data Specification: `<type int>`

Default Value: 1

## Selection Policy

This is a map from actions to policy for leaving a range selected after the action takes place. Possible actions are “indent-region”, “outdent-region”, “indent-to-match”, “comment-out-region”, and “uncomment-out-region”. Possible policies for each are “always-select”, which always leaves a selection, “retain-select” which leaves a selection only if there was one to begin with, and “never-select” which never leaves a selection.

Internal Name: `edit.select-policy`

Data Specification: `[dict; keys: [(u'Indent Region', 'indent-region'), (u'Indent To Match', 'indent-to-match'), (u'Uncomment out Region', 'uncomment-out-region'), (u'Outdent Region', 'outdent-region'), (u'Comment out Region', 'comment-out-region')], values: [(u'Never Select', 'never-select'), (u'Retain Select', 'retain-select'), (u'Always Select', 'always-select')]]`

Default Value: `{'uncomment-out-region': 'retain-select', 'outdent-`

```
region': 'retain-select', 'comment-out-region': 'retain-select',
'indent-region': 'retain-select', 'indent-to-match': 'retain-select'}
```

### **Middle Mouse Paste**

Paste text into the editor from the clipboard when the middle mouse button is pressed. Disabling this is mainly useful for wheel mice with a soft wheel that causes pasting of text before wheel scrolling starts.

Internal Name: `edit.middle-mouse-paste`

Data Specification: `<boolean: 0 or 1>`

Default Value: `True`

### **Default Drag-n-Drop Action**

Default drag-n-drop action. This is the default and can always be overridden by pressing shift or ctrl while dragging

Internal Name: `edit.default-drop-action`

Data Specification: `[os-default, copy, move]`

Default Value: `os-default`

### **Mini-search Case Sensitivity**

Whether or not mini-search is case sensitive. Can be 'match-mode' to use the default for the current keyboard personality ('never' in vi mode or 'if-upper' for emacs and other personalities), 'if-upper' to be case sensitive only if an upper case character is typed, 'always' to always match case sensitively, and 'never' to always match case insensitively.

Internal Name: `edit.minisearch-case-sensitive`

Data Specification: `[always, never, if-upper, match-mode]`

Default Value: `match-mode`

### **Symbol Menu Max Length**

The maximum number of names allowed on a single symbol menu

Internal Name: `.edit.max-symbol-menu-name-count`

Data Specification: `<type int>`

Default Value: `200`

# Debugger

## Ignore Unsynchronized Files

Controls whether or not Wing ignores files that were not saved before starting debug or that have changed since they were loaded by the debug process. Wing normally will warn of unsynchronized files since breakpoints may not be reached and stepping through the files may not work properly if lines have moved. Checking this option turns off these warnings.

Internal Name: `gui.ignore-unsaved-before-action`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

## Raise Source From Tools

Controls whether the debugger raises source files to indicate exception locations encountered when working in the Debug Probe, and other debugger tools.

Internal Name: `debug.raise-from-tools`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

## Default Watch Style

Sets the tracking style used when a value is double clicked in order to watch it: Use “symbolic” to track by symbolic name, “parent-ref” to track parent by object reference and attribute by name, and “ref” to track using an object reference directly to the value

Internal Name: `debug.default-watch-style`

Data Specification: `[ref, parent-ref, symbolic]`

Default Value: `symbolic`

## Integer Display Mode

This sets the display style for integer values to one of “dec”, “hex”, or “oct”.

Internal Name: `debug.default-integer-mode`

Data Specification: `[dec, hex, oct]`

Default Value: `dec`

**Hover Over Symbols**

Set to display debug data value of any symbol on the editor when the mouse cursor hovers over it.

Internal Name: `debug.hover-over-symbols`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

**Hover Over Selection**

Controls whether debug values are shown when the mouse hovers over a selection in the editor. This may be disabled, enabled for symbols (like x.y.z) only, or enabled for all selections including function or methods calls. **WARNING:** Enabling evaluation of any selection may result in function or methods calls that have side effects such as altering the program state or even making unintended database or disk accesses!

Internal Name: `debug.hover-over-selections`

Data Specification: `[0, 1, all]`

Default Value: 1

**Line Threshold**

Defines the character length threshold under which a value will always be shown on a single line, even if the value is a complex type like a list or map

Internal Name: `debug.line-threshold`

Data Specification: `<type int>`

Default Value: 65

- **Exceptions**

**Report Exceptions**

Controls how Wing reports exceptions that are raised by your debug process. By default, Wing shows exceptions at the time that the exception traceback would normally be printed. Alternatively, Wing can try to predict which exceptions are unhandled, and stop immediately when unhandled exceptions are raised so that any finally clauses can be stepped through in the debugger. Wing can also stop on all exceptions (even if handled) immediately when they are raised, or it can wait to report fatal exceptions as the debug process terminates. In the latter case Wing makes a best effort to stop before the debug process exits or at least to report the exception post-mortem, but one

or both may fail if working with externally launched debug processes. In that case, we recommend using When Printed exception reporting mode.

Internal Name: `debug.exception-mode`

Data Specification: `[unhandled, always, never, printed]`

Default Value: `printed`

### **Report Logged Exceptions In When Printed Mode**

Controls whether to stop on exceptions logged with `logging.exception` if the exception mode is set to 'When Printed'

Internal Name: `debug.stop-on-logged-exception`

Data Specification: `<boolean: 0 or 1>`

Default Value: `True`

### **Never Report**

Names of builtin exceptions to never report, even if the exception is not handled. This list takes precedence over the always report list and the default reporting mode, but is not used if the exception reporting mode is set to always.

Internal Name: `debug.never-stop-exceptions`

Data Specification: `[tuple of: <type str>]`

Default Value: `['SystemExit', 'GeneratorExit']`

### **Always Report**

Names of builtin exceptions to (nearly) always report. These exceptions are not reported only if they explicitly caught by the specific subclass in the frame they are raised in.

Internal Name: `debug.always-stop-exceptions`

Data Specification: `[tuple of: <type str>]`

Default Value: `['AssertionError']`

## **• I/O**

### **Debug I/O Encoding**

Encoding of input/output in the Debug I/O panel

Internal Name: `debug.debug-io-encoding`

Data Specification: [None or [Central and Eastern European iso8859-2, Japanese iso-2022-jp-2004, Hebrew cp856, Japanese euc-jp, Vietnamese cp1258, Greek cp1253, Baltic Languages cp1257, Korean johab, Western European cp1252, Baltic Languages cp775, Japanese iso-2022-jp-ext, Korean iso-2022-kr, Icelandic cp861, Hebrew cp424, Cyrillic Languages cp1251, Turkish iso8859-9, Unicode (UTF-16, little endian) utf-16-le, Western European cp500, Chinese (PRC) gb18030, Greek cp875, Arabic cp864, Icelandic mac-iceland, Chinese (PRC) gbk, Turkish mac-turkish, Greek iso8859-7, Baltic Languages iso8859-13, Cyrillic Languages mac-cyrillic, Greek cp869, Turkish cp1254, Japanese iso-2022-jp-1, Central and Eastern European cp852, Japanese iso-2022-jp-2, Chinese (ROC) big5, Urdu cp1006, Hebrew iso8859-8, Celtic Languages iso8859-14, Thai cp874, Cyrillic Languages cp855, Western European iso8859-15, Greek mac-greek, Ukrainian koi8-u, Hebrew cp1255, Danish, Norwegian cp865, Cyrillic Languages iso8859-5, Turkish cp1026, Western European mac-roman, Western European cp1140, Chinese (PRC) hz, Japanese shift-jisx0213, Portuguese cp860, Chinese (ROC) cp950, Unicode (UTF-16, big endian) utf-16-be, Japanese shift-jis-2004, Console default (UTF-8), Japanese iso-2022-jp-3, Hebrew cp862, Western European latin-1, Japanese euc-jisx0213, US, Canada, and Others cp037, Japanese euc-jis-2004, None, Central and Eastern European cp1250, Baltic Languages iso8859-4, English ascii, Japanese shift-jis, Arabic iso8859-6, Canadian English/French cp863, Russian koi8-r, Japanese iso-2022-jp, Unicode (UTF-8) utf-8, Greek cp737, Nordic Languages iso8859-10, Central and Eastern European mac-latin2, Chinese (PRC) gb2312, Unicode (UTF-7) utf-7, Arabic cp1256, Chinese (PRC) big5hkscs, Western European cp850, Esperanto and Maltese iso8859-3, Turkish cp857, Korean cp949, US, Australia, New Zealand, S. Africa cp437, Unicode (UTF-16) utf-16, Japanese cp932]]

Default Value: None

## Shell Encoding

Encoding of input/output in the integrated Python Shell

Internal Name: `debug.debug-probe-encoding`

Data Specification: [None or [Central and Eastern European iso8859-2, Japanese iso-2022-jp-2004, Hebrew cp856, Japanese euc-jp, Vietnamese cp1258, Greek cp1253, Baltic Languages cp1257, Korean johab, Western European cp1252, Baltic Languages cp775, Japanese iso-2022-jp-ext, Korean iso-2022-kr, Icelandic cp861, Hebrew cp424, Cyrillic Languages cp1251, Turkish iso8859-9, Unicode (UTF-16, little endian) utf-16-le, West-

ern European cp500, Chinese (PRC) gb18030, Greek cp875, Arabic cp864, Icelandic mac-iceland, Chinese (PRC) gbk, Turkish mac-turkish, Greek iso8859-7, Baltic Languages iso8859-13, Cyrillic Languages mac-cyrillic, Greek cp869, Central and Eastern European mac-latin2, Japanese iso-2022-jp-1, Central and Eastern European cp852, Japanese iso-2022-jp-2, Chinese (ROC) big5, Urdu cp1006, Hebrew iso8859-8, Celtic Languages iso8859-14, Thai cp874, Cyrillic Languages cp855, Western European iso8859-15, Greek mac-greek, Ukrainian koi8-u, Hebrew cp1255, Danish, Norwegian cp865, Cyrillic Languages iso8859-5, Turkish cp1026, Western European mac-roman, Western European cp1140, Chinese (PRC) hz, Japanese shift-jisx0213, Portuguese cp860, Chinese (ROC) cp950, US, Canada, and Others cp037, Japanese shift-jis-2004, Turkish cp1254, Japanese iso-2022-jp-3, Hebrew cp862, Western European latin-1, Japanese euc-jisx0213, Unicode (UTF-16, big endian) utf-16-be, Japanese euc-jis-2004, None, Central and Eastern European cp1250, Baltic Languages iso8859-4, English ascii, Japanese shift-jis, Use default stdin / stdout encoding, Canadian English/French cp863, Russian koi8-r, Japanese iso-2022-jp, Unicode (UTF-8) utf-8, Greek cp737, Nordic Languages iso8859-10, Arabic iso8859-6, Chinese (PRC) gb2312, Unicode (UTF-7) utf-7, Arabic cp1256, Chinese (PRC) big5hkscs, Western European cp850, Esperanto and Maltese iso8859-3, Turkish cp857, Korean cp949, US, Australia, New Zealand, S. Africa cp437, Unicode (UTF-16) utf-16, Japanese cp932]]

Default Value: None

### **Pretty Print in Shells**

Set to using pprint.pprint to display values in the Python Shell and Debug Probe.

Internal Name: debug.pretty-print-in-shells

Data Specification: <boolean: 0 or 1>

Default Value: False

### **Use External Console**

Selects whether to use the integrated I/O panel for debug process input/output or an external terminal window. Use an external window if your debug process depends on details of the command prompt environment for cursor movement, color text, etc.

Internal Name: debug.external-console

Data Specification: <boolean: 0 or 1>

Default Value: 0

### External Console Waits on Exit

Set to true to leave up the console after normal program exit, or false to close the console right away in all cases. This is only relevant when running with an external native console instead of using the integrated debug I/O panel.

Internal Name: `debug.persist-console`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### External Consoles

A list of the xterm-compatible X windows terminal programs that are used with debug processes when running with an external console. Each is tried in turn until one is found to exist. If just the name is given, Wing will look for each first on the PATH and then in likely places. Specify the full path (starting with “/”) to use a specific executable. If program arguments are specified, they must end with the argument that indicates that the rest of arguments are the program to run in the terminal

Internal Name: `debug.x-terminal`

Data Specification: `[tuple of: <type str>]`

Default Value: `('gnome-terminal "--title=Wing Debug Console" -x', 'xterm -T "Wing Debug Console" -e', 'konsole -T "Wing Debug Console" -e', 'rxvt -T "Wing Debug Console" -e')`

- **Data Filters**

### Huge List Threshold

Defines the length threshold over which a list, map, or other complex type will be considered too large to show in the normal debugger. If this is set too large, the debugger will time out (see network-timeout preference)

Internal Name: `debug.huge-list-threshold`

Data Specification: `<type int>`

Default Value: 2000

### Huge String Threshold

Defines the length over which a string is considered too large to fetch for display in the debugger. If this is set too large, the debugger will time out (see network-timeout preference).



Internal Name: `debug.huge-string-threshold`

Data Specification: `<type int>`

Default Value: 64000

### **Omit Types**

Defines types for which values are never shown by the debugger.

Internal Name: `debug.omit-types`

Data Specification: `[tuple of: <type str>]`

Default Value: `('function', 'builtin_function_or_method', 'class', 'classobj', 'instance method', 'type', 'module', 'ufunc', 'wrapper_descriptor', 'method_descriptor', 'member_descriptor')`

### **Omit Names**

Defines variable/key names for which values are never shown by the debugger.

Internal Name: `debug.omit-names`

Data Specification: `[tuple of: <type str>]`

Default Value: `()`

### **Do Not Expand**

Defines types for which values should never be probed for contents. These are types that are known to crash when the debugger probes them because they contain buggy data value extraction code. These values are instead shown as an opaque value with hex object instance id and are never accessed for runtime introspection.

Internal Name: `debug.no-probe-types`

Data Specification: `[tuple of: <type str>]`

Default Value: `('GdkColormap', 'IOBTree', 'JPackage')`

- **External/Remote**

### **Enable Passive Listen**

Controls whether or not the debugger listens passively for connections from an externally launched program (false to disable; true to enable). This should be on when the debug program is not launched by the IDE (e.g., as with a CGI script).

Internal Name: `debug.passive-listen`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **Allowed Hosts**

Sets which hosts are allowed to connect to the debugger when it is listening passively for externally launched programs.

Internal Name: `debug.passive-hosts`

Data Specification: `[tuple of: <type str>]`

Default Value: `('127.0.0.1',)`

### **Server Host**

Determines the network interface on which the debugger listens for connections. This can be a symbolic name, an ip address, or left unspecified (use None) to indicate that the debugger should listen on all valid network interfaces on the machine. Note that when a debug session is launched from within the IDE (with the Run button), it always connects from the loopback interface (127.0.0.1)

Internal Name: `debug.network-server`

Data Specification: `[None or <type str>]`

Default Value: None

### **Server Port**

Determines the TCP/IP port on which the IDE will listen for the connection from the debug process. This needs to be unique for each developer working on a given host. The debug process, if launched from outside of the IDE, needs to be told the value specified here using `kWingHostPort` inside `wingdbstub.py` or by `WINGDB_HOSTPORT` environment variable before importing `wingdbstub` in the debug process.

Internal Name: `debug.network-port`

Data Specification: `[from 0 to 65535]`

Default Value: 50005

### **Location Map**

Defines a mapping between the remote and local locations of files for host-to-host debugging. Each mapping key is the ip address of the remote location and the mapping

values are arrays of tuples where each tuple is a (remote\_prefix, local\_prefix) pair. This should be used when files on the remote host are updated via ftp, NFS, Samba, or other method from master copies on the local host, but the full path file system locations on the local and remote hosts do not match.

Internal Name: `debug.location-map`

Data Specification: [dict; keys: <ip4 address #.#.#.#>, values: [None or [list of: [tuple length 2 of: <type str>, <type str>]]]]

Default Value: {'127.0.0.1': None}

### **Kill Externally Launched**

Enable or disable the Kill command for debug processes that were launched from outside of the IDE

Internal Name: `debug.enable-kill-external`

Data Specification: <boolean: 0 or 1>

Default Value: 0

### **Common Attach Hosts**

List of host/port combinations that should be included by default in the attach request list shown with Attach to Process in the Debug menu, in addition to those that are registered at runtime. These are used primarily with externally launched debug processes, since Wing automatically shows IDE-launched processes for attach when appropriate. This value corresponds with `kAttachPort` configured in `wingdbstub.py` or by `WINGDB_ATTACHPORT` environment variable before importing `wingdbstub` in the debug process.

Internal Name: `debug.attach-defaults`

Data Specification: [tuple of: [tuple length 2 of: <type str>, [from 0 to 65535]]]

Default Value: (('127.0.0.1', 50015),)

- **Advanced**

### **Network Timeout**

Controls the amount of time that the debug client will wait for the debug server to respond before it gives up. This protects the IDE from freezing up if your program running within the debug server crashes (or if the server itself becomes unavailable). It

must also be taken into account when network connections are slow or if sending large data values (see the `huge-list-threshold` and `huge-string-threshold` preferences).

Internal Name: `debug.network-timeout`

Data Specification: `<type float>`, `<type int>`

Default Value: 10

### **Resolve Properties**

Set to show property values in the debug data views. This should be used with caution. It enables invocation of the `fget()` method on the property, which in some code bases can execute unwanted code, make unexpected changes to runtime state, hang on lengthy computations, trigger thread deadlocks, or crash on buggy user code while debug data is being displayed in the IDE.

Internal Name: `debug.resolve-properties`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **Allow dynamic introspection**

Set to allow Python code and other dynamic calls to be invoked while introspecting values in the debugger, for display in the auto-completer, shells, and source assistant. This should be used with caution. In some code bases, enabling this can execute unwanted code, make unexpected changes to runtime state, hang on lengthy computations, trigger thread deadlocks, or crash on buggy user code, while working in the IDE.

Internal Name: `debug.allow-dynamic-introspection`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **Show Data Warnings**

Controls whether or not time out, huge value, and error handling value errors are displayed by the debugger the first time they are encountered in each run of Wing.

Internal Name: `debug.show-debug-data-warnings`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Use sys.stdin Wrapper**

Whether `sys.stdin` should be set a wrapper object for user input in the program being debugged. The wrapper allows debug commands, such as pause, to be executed while the program is waiting for user input. The wrapper may cause problems with multi-threaded programs that use C stdio functions to read directly from stdin and will be slower than the normal file object. However, turning this preference off means that your debug process will not pause or accept breakpoint changes while waiting for keyboard input, and any keyboard input that occurs as a side effect of commands typed in the Debug Probe will happen in unmodified stdin instead (even though output will still appear in the Debug Probe as always).

Internal Name: `debug.use-stdin-wrapper`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Shells Ignore Editor Modes**

Set to False so that shells will act modal in the same way as editors when working with a modal key bindings such as that for VI. When True, the shells always act as if in Insert mode.

Internal Name: `debug.shells-ignore-editor-modes`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Execute Pasted Lines in Shells Immediately**

Whether to always execute immediately after text is pasted into a shell. Note that if the number of lines exceed the pasted line threshold, the lines are immediately executed.

Internal Name: `debug.shell-always-execute-on-paste`

Data Specification: `<boolean: 0 or 1>`

Default Value: False

- **Diagnostics**

### **Debug Internals Log File**

This is used to obtain verbose information about debugger internals in cases where you are having problems getting debugging working. When set to non-None value, debugger activity is logged to the given file name. Alternatively, “<stdout>” or “<stderr>” can be used.

Internal Name: `debug.logfile`

Data Specification: [one of: `None`, [`<stdout>`, `<stderr>`], `<type str>`]

Default Value: `None`

### **Extremely Verbose Internal Log**

This is used to turn on very verbose and detailed logging from the debugger. Only recommended when debugging the debugger.

Internal Name: `debug.very-verbose-log`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **Python Shell Debug Log**

This is used to obtain verbose information about the Python Shell internals in cases where you are having problems getting it working. When set to non-None value, debugger activity is logged to the given file name. Alternatively, “`<stdout>`” or “`<stderr>`” can be used to send output to the Python Shell tool.

Internal Name: `debug.shell-logfile`

Data Specification: [one of: `None`, [`<stdout>`, `<stderr>`], `<type str>`]

Default Value: `None`

### **Extremely Verbose Python Shell Debug Log**

This is used to turn on very verbose and detailed logging from the Python Shell internals. Only recommended when debugging the Python Shell.

Internal Name: `debug.very-verbose-shell-log`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

## **Source Analysis**

### **Analyze in Background**

Whether Wing should try to analyze python source in the background.

Internal Name: `pysource.analyze-in-background`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Introspect Live Runtime**

Set to introspect live Python runtimes for information displayed in autocompletion, the Source Assistant, and debug data value tooltips. Runtimes introspected include the Python Shell and live debug processes stopped at an exception or breakpoint.

Internal Name: `debug.introspect-in-shells`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **Max Cache Size (MB)**

The maximum size of the disk cache in megabytes

Internal Name: `pysource.max-disk-cache-size`

Data Specification: `[from 1 to 1000]`

Default Value: 500

### **Max Memory Buffers**

The maximum # of analysis info buffers that can be in-memory at once for files that are not open.

Internal Name: `pysource.max-background-buffers`

Data Specification: `[from 1 to 100]`

Default Value: 80

### **Typing Suspend Timeout**

Number of seconds between last key press and when analysis is re-enabled if analysis is to be suspended while typing occurs. If  $\leq 0$ , analysis is not suspended.

Internal Name: `edit.suspend-analysis-timeout`

Data Specification: `<type float>, <type int>`

Default Value: 3

- **Advanced**

### Interface File Path

Path to search for interface files for extension modules. If directory name is relative, it will be interpreted as relative to the user settings directory (USER\_SETTINGS\_DIR)

Internal Name: `pysource.interfaces-path`

Data Specification: [tuple of: <type str>]

Default Value: ('pi-files',)

### Scrape Extension Modules

Set this to False to disable automatic loading of extension modules and other modules that cannot be statically analysed. These modules are loaded in another process space and 'scraped' to obtain at least some analysis of the module's contents.

Internal Name: `pysource.scrape-modules`

Data Specification: <boolean: 0 or 1>

Default Value: True

### Scraping Helper Snippets

This is a dictionary from module name to Python code that should be executed before attempting to load extension modules for scraping. This is needed in cases where the extension modules are designed to be loaded only after some configuration magic is performed. For most extension modules, no extra configuration should be needed.

Internal Name: `pysource.scrape-config`

Data Specification: [dict; keys: <type str>, values: <type str>]

Default Value: `{'QtSvg': 'try:\n from PyQt4 import QtSvg\nexcept:\n from PySide import QtSvg\n', 'wxpython': 'pass', 'QtHelp': 'try:\n from PyQt4 import QtHelp\nexcept:\n from PySide import QtHelp\n', 'gdk': 'import pygtk\nvers = pygtk._get_available_versions().keys()\nvers.sort()\nvers.r\n v in vers:\n try:\n pygtk.require(v)\n break\n except:\n pass\n', 'Qt-Gui': 'try:\n from PyQt4 import QtGui\nexcept:\n from PySide import Qt-Gui\n', '_gst': 'from gst import _gst', 'gtk': 'import pygtk\nvers = pygtk._get_available_versions().keys()\nvers.sort()\nvers.reverse()\nfor v in vers:\n try:\n pygtk.require(v)\n break\n except:\n pass\n', 'QtXml': 'try:\n from PyQt4 import QtXml\nexcept:\n from PySide import QtXml\n', 'QtWebKit': 'try:\n from PyQt4 import QtWebKit\nexcept:\n from PySide import QtWebKit\n', 'QtScriptTools': 'try:\n from PyQt4`



```

import QtScriptTools\nexcept:\n from PySide import QtScriptTools\n',
'QtSql': 'try:\n from PyQt4 import QSql\nexcept:\n from PySide im-
port QSql\n', 'Qt': 'try:\n from PyQt4 import Qt\nexcept:\n from Py-
Side import Qt\n', 'QtAssistant': 'try:\n from PyQt4 import QtAs-
sistant\nexcept:\n from PySide import QtAssistant\n', 'QtXmlPat-
terns': 'try:\n from PyQt4 import QtXmlPatterns\nexcept:\n from Py-
Side import QtXmlPatterns\n', 'QtDeclarative': 'try:\n from PyQt4 im-
port QtDeclarative\nexcept:\n from PySide import QtDeclarative\n',
'QtDesigner': 'try:\n from PyQt4 import QtDesigner\nexcept:\n
from PySide import QtDesigner\n', 'pango': 'import pygtk\nvers =
pygtk._get_available_versions().keys()\nvers.sort()\nvers.reverse()\nfor
v in vers:\n try:\n pygtk.require(v)\n break\n except:\n pass\n',
'QtOpenGL': 'try:\n from PyQt4 import QtOpenGL\nexcept:\n from Py-
Side import QtOpenGL\n', 'QtUiTools': 'try:\n from PyQt4 import QtU-
iTools\nexcept:\n from PySide import QtUiTools\n', 'QSci': 'try:\n from
PyQt4 import QSci\nexcept:\n from PySide import QSci\n', 'atk': 'import
pygtk\nvers = pygtk._get_available_versions().keys()\nvers.sort()\nvers.reverse()\nfor
v in vers:\n try:\n pygtk.require(v)\n break\n except:\n pass\n',
'QtTest': 'try:\n from PyQt4 import QtTest\nexcept:\n from Py-
Side import QtTest\n', 'QtScript': 'try:\n from PyQt4 import
QtScript\nexcept:\n from PySide import QtScript\n', 'gobject': 'import
pygtk\nvers = pygtk._get_available_versions().keys()\nvers.sort()\nvers.reverse()\nfor
v in vers:\n try:\n pygtk.require(v)\n break\n except:\n pass\n', 'Qt-
Core': 'try:\n from PyQt4 import QtCore\nexcept:\n from PySide import
QtCore\n', 'QtNetwork': 'try:\n from PyQt4 import QtNetwork\nexcept:\n
from PySide import QtNetwork\n'}

```

## Python Docs URL Prefix

Prefix for Python Standard Library Documentation. This should be in the form <http://docs.python.org/library/> and Wing will append module and symbol specific to the given URL. To use locally stored documentation, you must run a local web server since # bookmarks do not work in file: URLs.

Internal Name: pysource.python-doc-url-prefix

Data Specification: [None or <type int>]

Default Value: None

## Use sqlite dotfile locking

Use slower, dotfile locking for sqllite databases to work around buggy remote file servers. Only needed if the user cache directory is on a remote file system or can be accessed via a remote file system. It is recommended that the user cache directory be on the local file system for performance reasons.

Internal Name: `pysource.use-sqlite-dotfile-locking`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

## Network

### HTTP Proxy Server

Allows manual configuration of an http proxy to be used for feedback, bug reports, and license activation, all of which result in Wing connecting to wingware.com via http. Leave user name and password blank if not required.

Internal Name: `main.http-proxy`

Data Specification: `[None or [tuple length 4 of: <type str>, <type int>, <type str>, <type str>]]`

Default Value: `None`

## Internal Preferences

### Core Preferences

#### **main.auto-reload-scripts**

When enabled, Wing will automatically reload scripts that extend the IDE when they are edited and saved from the IDE. This makes developing extension scripts for the IDE very fast, and should work in most cases. Disable this when working on extension scripts that do not reload properly, such as those that reach through the scripting API extensively.

Internal Name: `main.auto-reload-scripts`

Data Specification: `<boolean: 0 or 1>`

Default Value: `True`

#### **main.debug-break-on-critical**

If True and a gtk, gdk, or glib critical message is logged, Wing tries to start a C debugger and break at the current execution point

Internal Name: `main.debug-break-on-critical`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **main.documentation-language**

The language to use for the documentation, when available (not all documentation is translated into all supported languages).

Internal Name: `main.documentation-language`

Data Specification: `[None, de, en, fr]`

Default Value: `en`

### **main.extra-mime-type-comments**

This is a map from mime type to tuple of start/end comment characters for each mime type. One entry should be added for each new mime type added with the `main.extra-mime-types` preference.

Internal Name: `main.extra-mime-type-comments`

Data Specification: `[dict; keys: <type str>, values: [tuple length 2 of: <type str>, <type str>]]`

Default Value: `{}`

### **main.extra-mime-type-names**

This is a map from mime type to displayable name for that mime type; one entry should be added for each new mime type added with the `main.extra-mime-types` preference.

Internal Name: `main.extra-mime-type-names`

Data Specification: `[dict; keys: <type str>, values: <type str>]`

Default Value: `{}`

### **main.ignored-updates**

Used internally to keep track of updates the user is not interested in

Internal Name: `main.ignored-updates`

Data Specification: [list of: <type str>]

Default Value: []

### **main.perspective-auto-save**

Selects whether to auto-save perspectives when switching to another perspective. Can always auto-save, never auto-save, prompt each time a perspective is left, or auto-save as configured on a per-perspective basis.

Internal Name: `main.perspective-auto-save`

Data Specification: [tuple length 2 of: [always, never, prompt, choose], <type str>]

Default Value: `always`

### **main.perspective-shared-file**

Selects the file to use for storing and retrieving shared perspectives. By default (when value is None) the file 'perspectives' in the user settings directory is used.

Internal Name: `main.perspective-shared-file`

Data Specification: [one of: <type NoneType>, <type str>]

Default Value: `None`

### **main.perspective-style**

Controls what parts of the GUI are altered when switching perspectives. Tools and Layout only captures the set of visible tools and overall layout, Tool, Layout, and Editors stores also the set of open editors, and All Visual State captures all details of the visual state, including the state of the tools, the size and position of windows, and the caret position.

Internal Name: `main.perspective-style`

Data Specification: [tuple length 2 of: [tools-and-editors, all, tools-only], <type str>]

Default Value: `tools-only`

### **main.plugin-overrides**

Defines which plugins are enabled or disabled.

Internal Name: `main.plugin-overrides`

Data Specification: [dict; keys: <type str>, values: <boolean: 0 or 1>]

Default Value: {}

### **main.script-path**

Specifies the directories in which Wing will look for user-defined scripts that extend the functionality of the IDE itself. The directory names may contain environment variables in the \$(envname) form. Use \$(WING:PROJECT\_DIR) for the project directory. For each directory, Wing will load all found Python modules and packages, treating any function whose name starts with a letter (not \_ or \_\_) as a script-provided command. Extension scripts found in files within directories later in the list will override scripts of the same name found earlier, except that scripts can never override commands that are defined internally in Wing itself (these are documented in the Command Reference in the users manual). See the Scripting and Extending chapter of the manual for more information on writing and using extension scripts. Note that WINGHOME/scripts is always appended to the given path since it contains scripts that ship with Wing.

Internal Name: `main.script-path`

Data Specification: [list of: <type str>]

Default Value: [u'USER\_SETTINGS\_DIR/scripts']

## **User Interface Preferences**

### **gui.apple-keyboard**

Whether an Apple keyboard is in use. Use query x11 option to attempt to determine setting from X11 server each time Wing is run. This is an OS X only preference.

Internal Name: `gui.apple-keyboard`

Data Specification: [query-x11, yes, no]

Default Value: `query-x11`

### **gui.feedback-email**

Email address to use by default in the Feedback and Bug Report dialogs

Internal Name: `gui.feedback-email`

Data Specification: <type str>

Default Value: ""

**gui.fix-osx-tiger-keyboard-conflict**

Whether to fix the inability to use Mode\_switch on Tiger (OS X 10.4). If true, Wing will run xmodmap when it starts to remap the Mode\_switch keys (option, Alt Gr, and other composition keys on non-US keyboards) from mod1 to mod5. The xmodmap modifications will affect all X11 applications.

Internal Name: `gui.fix-osx-tiger-keyboard-conflict`

Data Specification: `<boolean: 0 or 1>`

Default Value: `True`

**gui.osx-key-for-alt**

Use key for alt key in all X11 applications on OS X -- typically used when using a non OS X keyboard layout on the Apple X11 server. The option key should be used only if it's not needed to enter individual characters. This will use xmodmap to set the global X11 key map to use the specified key as the alt key modifier. Turning this option off if it was on previously will reset the option key back to mode\_switch, which is the Apple default setting. Non-default options will override any externally set xmodmap setting so use with care if you've customized your xmodmap.

Internal Name: `gui.osx-key-for-alt`

Data Specification: `[default, command, option2, option]`

Default Value: `default`

**gui.last-feedback-shown**

Used internally to avoid showing the feedback dialog on exit over and over again.

Internal Name: `gui.last-feedback-shown`

Data Specification: `<type float>`

Default Value: `0.0`

**gui.more-controls-for-search-in-files**

Controls whether "Search in Files" dialog has an extra row of visible options as buttons.

Internal Name: `gui.more-controls-for-search-in-files`

Data Specification: `<boolean: 0 or 1>`

Default Value: `0`

**gui.prefered-symbol-order**

Control preferred order in source index displays such as the editor browse menus. Either sort in “file-order” or “alpha-order”.

Internal Name: `gui.prefered-symbol-order`

Data Specification: `[file-order, alpha-order]`

Default Value: `alpha-order`

**gui.reported-exceptions**

Used internally to remember which unexpected exceptions have already been reported so we only show error reporting dialog once for each. This is a dict from product version to dict of exception info.

Internal Name: `gui.reported-exceptions`

Data Specification: `[dict; keys: <type str>, values: [dict; keys: <type str>, values: <boolean: 0 or 1>]]`

Default Value: `{}`

**gui.scan-for-pythoncom-shell-extensions**

Scan for pythoncom shell extensions on Windows

Internal Name: `gui.scan-for-pythoncom-shell-extensions`

Data Specification: `<boolean: 0 or 1>`

Default Value: `True`

**gui.set-win32-foreground-lock-timeout**

Controls whether or not to set the foreground lock timeout on Windows 98/ME and 2K/XP. On these systems, normally Wing will be unable to bring source windows to front whenever the debug process has windows in the foreground. When this preference is true, the system-wide value that prevents background applications from raising windows is cleared whenever Wing is running. This means that other apps will also be able to raise windows without these restrictions while Wing is running. Set the preference to false to avoid this, but be prepared for windows to fail to raise in some instances. Note: If Wing is terminated abnormally or from the task manager, the changed value will persist until the user logs out (or reboot on 98/ME).

Internal Name: `gui.set-win32-foreground-lock-timeout`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **gui.show-feedback-dialog**

Whether feedback dialog is shown to user on quit.

Internal Name: `gui.show-feedback-dialog`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **gui.show-osx-keyboard-warning**

Used internally to show information about osx keyboard issues to new users. Once turned off, it is never turned on again

Internal Name: `gui.show-osx-keyboard-warning`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **gui.startup-show-wingtips**

Controls whether or not the Wing Tips tool is shown automatically at startup of the IDE.

Internal Name: `gui.startup-show-wingtips`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **guimgr.toolbar-groups**

Controls which groups of tools will be shown in the toolbar.

Internal Name: `guimgr.toolbar-groups`

Data Specification: `[tuple of: [search, indent, clip, proj, file, diff, debug, batch-search]]`

Default Value: `['file', 'clip', 'search', 'indent', 'diff', 'proj', 'debug']`



## Editor Preferences

### **consoles.auto-clear**

Automatically clear the OS Commands consoles each time the command is re-executed

Internal Name: `consoles.auto-clear`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **edit.auto-edit-close**

Enable to auto-close quotes, parenthesis, braces, comments, and so forth.

Internal Name: `edit.auto-edit-close`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

### **edit.auto-edit-colon**

Auto-enter newline and auto-indent after typing a colon that starts a new Python block and indent following line or block of lines when colon is pressed repeatedly.

Internal Name: `edit.auto-edit-colon`

Data Specification: `<boolean: 0 or 1>`

Default Value: `0`

### **edit.auto-edit-comment**

Enable commenting out a non-empty selection when a comment character is pressed.

Internal Name: `edit.auto-edit-comment`

Data Specification: `<boolean: 0 or 1>`

Default Value: `1`

### **edit.auto-edit-continue**

Automatically continue comments or strings in the form `(“”)` or `()` after a newline is typed within the comment or string text

Internal Name: `edit.auto-edit-continue`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

#### **edit.auto-edit-enabled**

Enable or disable all auto-editing operations. When enabled, individual operations must be selected in preferences.

Internal Name: `edit.auto-edit-enabled`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

#### **edit.auto-edit-fixups**

Automatically correct code when typing keys out of order. This handles cases such as `x(.) -> x().` and `x(:) -> x():` as well as auto-inserting `.` when missing

Internal Name: `edit.auto-edit-fixups`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

#### **edit.auto-edit-invoke**

Enable auto-entry of invocation arguments for a function or method call.

Internal Name: `edit.auto-edit-invoke`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

#### **edit.auto-edit-parens**

Enable surrounding non-empty selection when a parenthesis is pressed.

Internal Name: `edit.auto-edit-parens`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

#### **edit.auto-edit-quotes**

Enable placing quotes around a non-empty selection.

Internal Name: `edit.auto-edit-quotes`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **`edit.auto-edit-spaces`**

Enable auto-entering spaces around operators and punctuation.

Internal Name: `edit.auto-edit-spaces`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **`edit.auto-edit-spaces-enforce`**

When auto-entering spaces is enabled, enforce PEP8 style spacing by preventing redundant spaces.

Internal Name: `edit.auto-edit-spaces-enforce`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **`edit.auto-edit-spaces-kw`**

Enable auto-entering spaces after keywords.

Internal Name: `edit.auto-edit-spaces-kw`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **`edit.autocomplete-autoshow`**

Controls whether or not the completer is shown automatically during typing. When disabled, it can still be shown on demand with the Show Completer item in the Source menu.

Internal Name: `edit.autocomplete-autoshow`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **`edit.bookmark-color`**

Color to use on the source editor to indicate the location of user-defined bookmarks.

Internal Name: `edit.bookmark-color`

Data Specification: [None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]

Default Value: (16, 192, 16)

### **edit.bookmark-style**

Visual display style to use for user-defined bookmarks: Either an underline, a background color change, or no visible marker.

Internal Name: `edit.bookmark-style`

Data Specification: [None, underline, background]

Default Value: background

### **edit.smart-clipboard**

Controls whether or not to copy or cut the whole current line when there is no selection on the editor.

Internal Name: `edit.smart-clipboard`

Data Specification: [disabled, copy-cut, copy]

Default Value: copy

### **edit.context-menu-custom-items**

Extra menu items to add to the editor context menu.

Internal Name: `edit.context-menu-custom-items`

Data Specification: [tuple of: [tuple length 2 of: <type str>, <type str>]]

Default Value: ()

### **edit.context-menu-groups**

Controls which groups of menu items will be shown in the editor's context menu.

Internal Name: `edit.context-menu-groups`

Data Specification: [tuple of: [comment, indent, clip, script, vcs, nav, debug]]

Default Value: `['clip', 'nav', 'debug', 'comment', 'indent', 'vcs', 'script']`

### **edit.dataentry-undo-grouping**

Whether or not to group all edits made during data entry mode into a single undo action.

Internal Name: `edit.dataentry-undo-grouping`

Data Specification: `<boolean: 0 or 1>`

Default Value: `False`

### **edit.diff-color**

Color to use on the source editor for differences during a diff/merge session. The current mark is drawn in a lighter version of the same color. The within-difference change indicators are drawn transparently with the color set in the Text Selection Color preference.

Internal Name: `edit.diff-color`

Data Specification: `[None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]`

Default Value: `(129, 212, 129)`

### **edit.fold-mime-types**

Set to a list of mime types for which folding should be allowed when folding in general is enabled.

Internal Name: `edit.fold-mime-types`

Data Specification: `[list of: <type str>]`

Default Value: `['text/x-python', 'text/x-c-source', 'text/x-cpp-source', 'text/x-java-source', 'text/x-javascript', 'text/html', 'text/x-mako', 'text/x-django', 'text/xml', 'text/x-zope-pt', 'text/x-eiffel', 'text/x-lisp', 'text/x-ruby', 'text/x-cython']`

### **edit.merged-diff-color**

Color to use on the source editor for already merged differences during a diff/merge session. The current mark is drawn in a lighter version of the same color. The within-difference change indicators are drawn transparently with the color set in the Text Selection Color preference.

Internal Name: `edit.merged-diff-color`

Data Specification: [None or [tuple length 3 of: [from 0 to 255], [from 0 to 255], [from 0 to 255]]]

Default Value: (212, 177, 177)

### **consoles.wrap-long-lines**

Wrap long output lines in OS Commands tool to fit within available display area.

Internal Name: `consoles.wrap-long-lines`

Data Specification: <boolean: 0 or 1>

Default Value: `False`

### **consoles.encoding**

Default encoding of sub-process input/output when run in the OS Commands panel. This can be overridden on a per-command basis, in each command's properties.

Internal Name: `consoles.encoding`

Data Specification: [None or [Central and Eastern European iso8859-2, Japanese iso-2022-jp-2004, Hebrew cp856, Japanese euc-jp, Vietnamese cp1258, Greek cp1253, Baltic Languages cp1257, Korean johab, Western European cp1252, Baltic Languages cp775, Japanese iso-2022-jp-ext, Korean iso-2022-kr, Icelandic cp861, Hebrew cp424, Cyrillic Languages cp1251, Turkish iso8859-9, Unicode (UTF-16, little endian) utf-16-le, Western European cp500, Chinese (PRC) gb18030, Greek cp875, Arabic cp864, Icelandic mac-iceland, Chinese (PRC) gbk, Turkish mac-turkish, Greek iso8859-7, Baltic Languages iso8859-13, Cyrillic Languages mac-cyrillic, Greek cp869, Turkish cp1254, Japanese iso-2022-jp-1, Central and Eastern European cp852, Japanese iso-2022-jp-2, Chinese (ROC) big5, Urdu cp1006, Hebrew iso8859-8, Celtic Languages iso8859-14, Thai cp874, Cyrillic Languages cp855, Western European iso8859-15, Greek mac-greek, Ukrainian koi8-u, Hebrew cp1255, Danish, Norwegian cp865, Cyrillic Languages iso8859-5, Turkish cp1026, Western European mac-roman, Western European cp1140, Chinese (PRC) hz, Japanese shift-jisx0213, Portuguese cp860, Chinese (ROC) cp950, Unicode (UTF-16, big endian) utf-16-be, Japanese shift-jis-2004, Console default (UTF-8), Japanese iso-2022-jp-3, Hebrew cp862, Western European latin-1, Japanese euc-jisx0213, US, Canada, and Others cp037, Japanese euc-jis-2004, None, Central and Eastern European cp1250, Baltic Languages iso8859-4, English ascii, Japanese shift-jis, Arabic iso8859-6, Canadian English/French cp863, Russian koi8-r, Japanese iso-2022-jp, Unicode (UTF-8) utf-8, Greek cp737, Nordic Languages iso8859-10, Central and Eastern European

mac-latin2, Chinese (PRC) gb2312, Unicode (UTF-7) utf-7, Arabic cp1256, Chinese (PRC) big5hkscs, Western European cp850, Esperanto and Maltese iso8859-3, Turkish cp857, Korean cp949, US, Australia, New Zealand, S. Africa cp437, Unicode (UTF-16) utf-16, Japanese cp932]]

Default Value: None

### **consoles.python-prompt-after-execution**

Drop into Python shell after executing any Python file in the OS Commands tool

Internal Name: `consoles.python-prompt-after-execution`

Data Specification: <boolean: 0 or 1>

Default Value: False

### **edit.symbol-find-alpha-sort**

Controls whether to sort Find Symbol dialog alphabetically or in natural file order

Internal Name: `edit.symbol-find-alpha-sort`

Data Specification: <boolean: 0 or 1>

Default Value: True

### **edit.symbol-find-include-args**

Controls whether to include argument specs in the searchable text used in the Find Symbol dialog

Internal Name: `edit.symbol-find-include-args`

Data Specification: <boolean: 0 or 1>

Default Value: False

### **edit.use-default-foreground-when-printing**

Use default foreground color for all text when printing. It's to set this if foreground color are customized for display on a dark background. The background color when printing is assumed to be white.

Internal Name: `edit.use-default-foreground-when-printing`

Data Specification: <boolean: 0 or 1>

Default Value: False

## Project Manager Preferences

### **proj.context-menu-custom-items**

Extra menu items to add to the Project tool context menu.

Internal Name: `proj.context-menu-custom-items`

Data Specification: `[tuple of: [tuple length 2 of: <type str>, <type str>]]`

Default Value: `()`

### **proj.context-menu-groups**

Controls which groups of menu items will be shown in the Project tool's context menu.

Internal Name: `proj.context-menu-groups`

Data Specification: `[tuple of: [clip, script, vcs, nav, proj, file, debug]]`

Default Value: `['clip', 'nav', 'debug', 'vcs', 'proj', 'file', 'script']`

### **proj.follow-editor**

Controls whether or not the IDE will follow the current editor by expanding the project tree to show the file open in the editor.

Internal Name: `proj.follow-editor`

Data Specification: `<boolean: 0 or 1>`

Default Value: `0`

### **proj.follow-selection**

Controls whether or not the IDE will follow the current project manager selection by opening the corresponding source file in a non-sticky (auto-closing) editor. In either case, the project manager will always open a file in sticky mode when an item is double clicked or the Goto Source context menu item is used.

Internal Name: `proj.follow-selection`

Data Specification: `<boolean: 0 or 1>`

Default Value: `0`

### **proj.open-from-project-full-paths**



Match fragments to full path of the file name, rather than just the file name. Full path matching still occurs when the path separation character is included in the search pattern.

Internal Name: `proj.open-from-project-full-paths`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **proj.file-type**

Controls the type of project file that is written by default for new projects: “normal” for regular single-file format with extension `.wpr`, and “shared” for split format where the `.wpr` file contains shared project info that can be checked into a shared revision control repository and the `.wpu` file contains user-specific information such as location of breakpoints. This is useful to avoid revision control wars on a project with multiple developers.

Internal Name: `proj.file-type`

Data Specification: `[shared, normal]`

Default Value: `shared`

## **Debugger Preferences**

### **debug.auto-clear-debug-io**

Set to automatically clear the debug I/O text each time a new debug session is started

Internal Name: `debug.auto-clear-debug-io`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **debug.auto-show-debug-io**

Set to automatically show the debug I/O tool the first time it receives output in each debug session

Internal Name: `debug.auto-show-debug-io`

Data Specification: `[False, True, first]`

Default Value: 1

### **debug.default-python-exec**

Sets the default Python Executable to use for debugging and source code analysis. This can be overridden on a project by project basis in Project Properties.

Internal Name: `debug.default-python-exec`

Data Specification: [None or <type str>]

Default Value: None

### **debug.filter-shell-history**

Filter shell history traversal when something is entered prior to starting traversal. Going forward or back will find the next or previous item starting with the text between the start of the current item and the cursor.

Internal Name: `debug.filter-shell-history`

Data Specification: <boolean: 0 or 1>

Default Value: False

### **main.launch-shared-file**

Selects the file to use for storing and retrieving shared launch configurations. By default (when value is None) the file 'launch' in the user settings directory is used.

Internal Name: `main.launch-shared-file`

Data Specification: [one of: <type NoneType>, <type str>]

Default Value: None

### **debug.python-exec**

Set this to override the default Python executable used with the debug server. A None (default) value uses /usr/bin/env python on Linux and the configured default on NT. Otherwise, give the full path of the python executable, e.g. /usr/local/bin/python or C:devpython. This preference only affects programs that are launched from the IDE.

Internal Name: `debug.python-exec`

Data Specification: [None or <type str>]

Default Value: None

### **debug.shell-auto-restart-before-eval**

Auto-restart the Python Shell before a file is evaluated within it. When this is disabled, be aware that previously defined symbols will linger in the Python Shell environment.

Internal Name: `debug.shell-auto-restart-before-eval`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **debug.shell-eval-whole-lines**

Evaluate whole lines from editor rather than the exact selection, when a selection from the editor is sent to the Python Shell tool.

Internal Name: `debug.shell-eval-whole-lines`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **debug.shell-pasted-line-threshold**

The number of lines after which the Python Shell will just print a summary rather than the actual lines of code pasted, dragged, or other transferred to the shell.

Internal Name: `debug.shell-pasted-line-threshold`

Data Specification: `<type int>`

Default Value: 30

### **debug.show-exceptions-tip**

Used internally to show information about exception handling to new users. Once turned off, it is never turned on again

Internal Name: `debug.show-exceptions-tip`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **debug.stop-timeout**

Number of seconds to wait before the debugger will stop in its own code after a pause request is received and no other Python code is reached.

Internal Name: `debug.stop-timeout`

Data Specification: `<type int>, <type float>`

Default Value: 3.0

### **debug.use-members-attrib**

Set this to true to have the debug server use the `__members__` attribute to try to interpret otherwise opaque data values. This is a preference because some extension modules contain bugs that result in crashing if this attribute is accessed. Note that `__members__` has been deprecated since Python version 2.2.

Internal Name: `debug.use-members-attrib`

Data Specification: `<boolean: 0 or 1>`

Default Value: 1

### **debug.warn-stale-shell**

Set to display a dialog when the Python Shell state no longer matches the configured Python Executable and/or Python Path.

Internal Name: `debug.warn-stale-shell`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **debug.wrap-debug-io**

Set to true to turn on line wrapping in the integrated debug I/O panel.

Internal Name: `debug.wrap-debug-io`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **debug.wrap-debug-probe**

Set to true to turn on line wrapping in the integrated debug probe panel.

Internal Name: `debug.wrap-debug-probe`

Data Specification: `<boolean: 0 or 1>`

Default Value: 0

### **debug.wrap-python-shell**

Set to true to turn on line wrapping in the Python shell panel.

Internal Name: `debug.wrap-python-shell`

Data Specification: `<boolean: 0 or 1>`

Default Value: `0`

## Source Analysis Preferences

**`pysource.instance-attrib-scan-mode`**

How to scan for instance attributes.

Internal Name: `pysource.instance-attrib-scan-mode`

Data Specification: `[init-only, all-methods]`

Default Value: `all-methods`



# Command Reference

This chapter describes the entire top-level command set of Wing IDE. Use this reference to look up command names for use in modified **keyboard bindings**.

Commands that list arguments of type `<numeric modifier>` accept either a number or previously entered numeric modifier. This is used with key bindings that provide a way to enter a numeric modifier (such as `Esc 1 2 3` in the emacs personality or typing numerals in browse mode in the vi personality).

## 12.1. Top-level Commands

### Application Control Commands

These are the high level application control commands.

**abandon-changes** (confirm=True)

Abandon any changes in the current document and reload it from disk. Prompts for user to confirm the operation unless either there are no local changes being abandoned or confirm is set to False.

**about-application** ()

Show the application-wide about box

**begin-visited-document-cycle** (move\_back=True, back\_key=None, forward\_key=None)

Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released. *Key Binding: Ctrl-Tab invokes begin-visited-document-cycle(move\_back=True)*

**bookmarks-menu-items** ()

Returns list of menu items for selecting among defined bookmarks

### **check-for-updates** ()

Check for updates to Wing IDE and offer to install any that are available

**close** (ignore\_changes=False, close\_window=True, can\_quit=False)

Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true. *Key Bindings: Normal: Ctrl-F4; VI/VIM: Ctrl-F4; Emacs: Ctrl-F4; Brief: Ctrl-F4; Visual Studio: Ctrl-F4; Eclipse (Experimental): Ctrl-F4; OS X: Command-F4*

**close-all** (omit\_current=False, ignore\_changes=False, close\_window=False)

Close all documents in the current window, or in all windows if in one-window-per-editor windowing policy. Leave currently visible documents (or active window in one-window-per-editor-mode) if omit\_current is True. Abandons changes rather than saving them when ignore\_changes is True. Close empty window and quit if all document windows closed when close\_window is True. *Key Bindings: Eclipse (Experimental): Ctrl-Shift-F2*

### **close-window** ()

Close the current window and all documents and panels in it *Key Bindings: Normal: Alt-F4; VI/VIM: Alt-F4; Emacs: Alt-F4; Brief: Alt-F4; Visual Studio: Alt-F4; Eclipse (Experimental): Alt-F4; OS X: Option-F4*

**command-by-name** (command\_name)

Execute given command by name, collecting any args as needed *Key Bindings: Normal: Ctrl-F12; VI/VIM: Ctrl-F12; Emacs: Ctrl-F12; Brief: Ctrl-F12; Visual Studio: Ctrl-F12; Eclipse (Experimental): Ctrl-F12; OS X: Command-F12*

### **copy-tutorial** ()

Prompt user and copy the tutorial directory from the Wing IDE installation to the directory selected by the user

### **edit-file-sets** ()

Show the File Sets preference editor

### **edit-preferences-file** ()

Edit the preferences as a text file

**execute-file** (loc=None)



Execute the file at the given location or use the active view if loc is None. *Key Bindings: Eclipse (Experimental): Ctrl-U*

**execute-os-command** (title)

Execute one of the stored commands in the OS Commands tool, selecting it by its title

**execute-os-command-by-id** (id, raise\_panel=True)

Execute one of the stored commands in the OS Commands tool, selecting it by its internal ID

**execute-process** (cmd\_line)

Execute the given command line in the OS Commands tool using default run directory and environment as defined in project properties, or the values set in an existing command with the same command line in the OS Commands tool. *Key Bindings: Emacs: Alt-!*

**goto-bookmark** (mark)

Goto named bookmark *Key Bindings: Normal: Ctrl-Alt-G; Emacs: Ctrl-X R B; Visual Studio: Ctrl-Alt-G; Eclipse (Experimental): Ctrl-Alt-G; OS X: Command-Shift-D*

**goto-next-bookmark** (current\_file\_only=False)

Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when current\_file\_only is True. *Key Bindings: Normal: Ctrl-Alt-Right; VI/VIM: Ctrl-Alt-Right; Emacs: Ctrl-Alt-Right; Brief: Ctrl-Alt-Right; Visual Studio: Ctrl-Alt-Right; Eclipse (Experimental): Ctrl-Alt-Right*

**goto-previous-bookmark** (current\_file\_only=False)

Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when current\_file\_only is True. *Key Bindings: Normal: Ctrl-Alt-Left; VI/VIM: Ctrl-Alt-Left; Emacs: Ctrl-Alt-Left; Brief: Ctrl-Alt-Left; Visual Studio: Ctrl-Alt-Left; Eclipse (Experimental): Ctrl-Alt-Left*

**hide-line-numbers** ()

Hide line numbers in editors

**initiate-numeric-modifier** (digit)

VI style repeat/numeric modifier for following command *Key Bindings: VI/VIM: 1 invokes initiate-numeric-modifier(digit=1)*

**initiate-repeat** ()

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Ctrl-U*

### **initiate-repeat-0 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-0*

### **initiate-repeat-1 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-1*

### **initiate-repeat-2 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-2*

### **initiate-repeat-3 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-3*

### **initiate-repeat-4 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-4; Brief: Ctrl-R*

### **initiate-repeat-5 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-5*

### **initiate-repeat-6 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-6*

### **initiate-repeat-7 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-7*

### **initiate-repeat-8 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-8*

### **initiate-repeat-9 ()**

Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke. *Key Bindings: Emacs: Alt-9*

### **internal-profile-start** ()

Start internal profiling. Profile information is collected for Wing IDE's internals until `internal_profile_stop` is executed.

### **internal-profile-stop** ()

Stop internal profiling after earlier `internal_profile_start` command. The profile can be found in the error-log file or submitted to Wingware as part of the error log included with a bug report from the Help menu.

### **new-blank-file** (filename)

Create a new blank file on disk, open it in an editor, and add it to the current project.

### **new-directory** (filename)

Create a new directory on disk and add it to the current project.

### **new-document-window** ()

Create a new document window with same documents and panels as in the current document window (if any; otherwise empty with default panels) *Key Bindings: Emacs: Ctrl-X 5 2; OS X: Shift-F4*

### **new-file** (ext='.py')

Create a new file *Key Bindings: Normal: Ctrl-N; Visual Studio: Ctrl-N; Eclipse (Experimental): Ctrl-N; OS X: Command-N*

### **new-package** (filename)

Create a new Python package directory on disk, add it to the current project, and open the new `__init__.py` in the editor.

### **new-panel-window** (panel\_type=None)

Create a new panel window of given type

### **next-document** (repeat=<numeric modifier; default=1>)

Move to the next document alphabetically in the list of documents open in the current window *Key Bindings: Normal: Ctrl-Page\_Down; VI/VIM: Ctrl-Page\_Down; Emacs: Ctrl-Page\_Down; Brief: Ctrl-Page\_Down; Visual Studio: Ctrl-Page\_Down; Eclipse (Experimental): Ctrl-Page\_Down; OS X: Command-0*

**next-window ()**

Switch to the next window alphabetically by title *Key Bindings: Normal: Ctrl-Comma; Emacs: Ctrl-O; Visual Studio: Ctrl-Comma; Eclipse (Experimental): Ctrl-Comma*

**nth-document** (n=<numeric modifier; default=0>)

Move to the nth document alphabetically in the list of documents open in the current window *Key Bindings: VI/VIM: Ctrl-^*

**open** (filename)

Open a file from disk using keyboard-driven selection of the file

**open-from-keyboard** (filename)

Open a file from disk using keyboard-driven selection of the file *Key Bindings: Normal: Ctrl-K; Emacs: Ctrl-X Ctrl-F; Visual Studio: Ctrl-K Ctrl-O; Eclipse (Experimental): Ctrl-K*

**open-from-project** (fragment=", skip\_if\_unique=False)

Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If skip\_if\_unique is true, the file is opened without the dialog being displayed if only one filename matches the fragment. *Key Bindings: Normal: Ctrl-Shift-O; VI/VIM: Ctrl-Shift-O; Emacs: Ctrl-X Ctrl-O; Visual Studio: Ctrl-Shift-O; Eclipse (Experimental): Ctrl-Shift-O; OS X: Command-Shift-O*

**open-gui** (filename=None)

Open a file from disk, prompting with file selection dialog if necessary *Key Bindings: Normal: Ctrl-O; Brief: Alt-E; Visual Studio: Ctrl-O; Eclipse (Experimental): Ctrl-O; OS X: Command-O*

**perspective-disable-auto** ()

Disable auto-perspectives

**perspective-enable-auto** ()

Enable auto-perspectives

**perspective-manage** ()

Display the perspectives manager dialog

**perspective-restore** (name)

Restore the given named perspective.

**previous-document** (repeat=<numeric modifier; default=1>)

Move to the previous document alphabetically in the list of documents open in the current window *Key Bindings: Normal: Ctrl-Page\_Up; VI/VIM: Ctrl-Page\_Up; Emacs: Ctrl-Page\_Up; Brief: Ctrl-Page\_Up; Visual Studio: Ctrl-Page\_Up; Eclipse (Experimental): Ctrl-Page\_Up; OS X: Command-9*

**previous-window** ()

Switch to the previous window alphabetically by title

**query-end-session** ()

Process query-end-session message on win32

**quit** ()

Quit the application. *Key Bindings: Normal: Ctrl-Q; Emacs: Ctrl-X Ctrl-C; Brief: Alt-X; Visual Studio: Ctrl-Q; Eclipse (Experimental): Ctrl-Q; OS X: Command-Q*

**recent-document** ()

Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode. *Key Bindings: Normal: Ctrl-8; Emacs: Ctrl-X D; Visual Studio: Ctrl-8; Eclipse (Experimental): Ctrl-8; OS X: Command-8*

**reload-scripts** ()

Force reload of all scripts, from all configured script directories. This is usually only needed when adding a new script file. Existing scripts are automatically reloaded when they change on disk.

**remove-bookmark** (mark)

Remove the given named bookmark

**remove-bookmark-current** ()

Remove bookmark at current line, if any. This command is only available if there is a bookmark on the line.

**rename-current-file** (filename)

Rename current file, moving the file on disk if it exists.

**restart-wing** ()

Restart the application

**restore-default-tools** ()

Hide/remove all tools and restore to original default state

**save** (close=False, force=False)

Save active document. Also close it if close is True. *Key Bindings: Normal: Ctrl-S; VI/VIM: Ctrl-S; Emacs: Ctrl-X Ctrl-S; Brief: Alt-W; Visual Studio: Ctrl-S; Eclipse (Experimental): Ctrl-S; OS X: Command-S*

**save-all** (close\_window=False)

Save all unsaved items, prompting for names for any new items that don't have a filename already. *Key Bindings: Visual Studio: Ctrl-Shift-S; Eclipse (Experimental): Ctrl-Shift-S*

**save-as** ()

Save active document to a new file *Key Bindings: Normal: Ctrl-Shift-S; Eclipse (Experimental): Ctrl-Shift-S; OS X: Command-Shift-S*

**scratch-document** (title='Scratch', mime\_type='text/plain')

Create a new scratch buffer with given title and mime type. The buffer is never marked as changed but can be saved w/ save-as.

**set-bookmark** (mark)

Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark. *Key Bindings: Normal: Ctrl-Alt-M; Emacs: Ctrl-X R M; Brief: Alt-0 invokes set-bookmark(mark="0"); Visual Studio: Ctrl-Alt-M; Eclipse (Experimental): Ctrl-Alt-M; OS X: Command-D*

**set-bookmark-default** ()

Set a bookmark at current line, using a default bookmark name for that context. This command is only available if there is not already a bookmark on the line.

**show-bookmarks** ()

Show a list of all currently defined bookmarks *Key Bindings: Normal: Ctrl-Alt-K; Emacs: Ctrl-X R Return; Brief: Alt-J; Visual Studio: Ctrl-Alt-K; Eclipse (Experimental): Ctrl-Alt-K; OS X: Command-Shift-K*

**show-bug-report-dialog** ()

Show the bug reporting dialog

**show-document** (section='manual')

Show the given documentation section *Key Bindings: OS X: Command-?*

**show-feedback-dialog** ()

Show the feedback submission dialog

**show-file-in-os-file-manager** (filename=None)

Show the selected file in the Explorer, Finder, or other OS-provided file manager. Shows the given file, if any, or the current file selected in the GUI.

**show-howtos** ()

Show the How-Tos index

**show-html-document** (section='manual')

Show the given document section in HTML format.

**show-line-numbers** (show=1)

Show the line numbers in editors

**show-manual-html** ()

Show the HTML version of the Wing IDE users manual

**show-manual-pdf** ()

Show the PDF version of the Wing IDE users manual for either US Letter or A4, depending on user's print locale

**show-panel** (panel\_type, flash=True, grab\_focus=None)

Show most recently visited panel instance of given type. If no such panel exists, add one to the primary window and show it. Returns the panel view object or None if not shown. Focus is shifted to panel if grab\_focus is specified and is true; if grab\_focus is not specified, it defaults to the value of flash.

The valid panel types are:

project (\*) browser (\*\*) batch-search (\*) interactive-search source-assistant (\*\*) debug-data debug-stack debug-io debug-exceptions debug-breakpoints (\*\*) debug-probe (\*\*) debug-watch (\*\*) debug-modules (\*\*) python-shell messages (\*) help indent (\*\*) bookmarks (\*\*) testing (\*\*) open-files (\*) os-command (\*\*) snippets (\*\*) diff (\*\*) uses (\*\*) refactoring (\*\*) versioncontrol.svn (\*\*) versioncontrol.hg (\*\*) versioncontrol.git (\*\*) versioncontrol.bzr (\*\*) versioncontrol.cvs (\*\*) versioncontrol.perforce (\*\*)

(\*) Wing Personal and Pro only (\*\*) Wing Pro only *Key Bindings: Eclipse (Experimental): Ctrl-E invokes show-panel(panel\_type="open-files")*

**show-panel-batch-search** (flash=True, grab\_focus=None)

Not documented

**show-panel-debug-data** (flash=True, grab\_focus=None)

Not documented

**show-panel-debug-exceptions** (flash=True, grab\_focus=None)

Not documented

**show-panel-debug-io** (flash=True, grab\_focus=None)

Not documented

**show-panel-debug-stack** (flash=True, grab\_focus=None)

Not documented

**show-panel-help** (flash=True, grab\_focus=None)

Not documented

**show-panel-indent** (flash=True, grab\_focus=None)

Not documented

**show-panel-interactive-search** (flash=True, grab\_focus=None)

Not documented

**show-panel-messages** (flash=True, grab\_focus=None)

Not documented

**show-panel-open-files** (flash=True, grab\_focus=None)

Not documented

**show-panel-project** (flash=True, grab\_focus=None)

Not documented

**show-panel-python-shell** (flash=True, grab\_focus=None)

Not documented



**show-panel-source-assistant** (flash=True, grab\_focus=None)

Not documented

**show-pdf-document** (doc='manual')

Show the given document in PDF format. One of 'manual', 'intro', or 'howtos'.

**show-plugins-gui** ()

Show the plugins GUI for enabling and disabling plugins

**show-preferences-gui** (prefname=None)

Edit the preferences file using the preferences GUI, optionally opening to the section that contains the given preference by name *Key Bindings: OS X: Command-Comma*

**show-python-donate-html** ()

Show the Python donations web page

**show-python-for-beginners-html** ()

Show the Python for Beginners web page

**show-python-manual-html** ()

Show the Python users manual

**show-python-org-html** ()

Show the python.org site home page

**show-python-org-search-html** ()

Show the python.org site search page

**show-quickstart** ()

Show the quick start guide

**show-success-stories-html** ()

Show the Python Success Stories page

**show-support-html** ()

Show the Wing IDE support site home page

**show-text-registers** ()

Show the contents of all non-empty text registers in a temporary editor

**show-tutorial** ()

Show the tutorial

**show-wingtip** (section='/' )

Show the Wing Tips window

**show-wingware-website** ()

Show the Wingware home page

**show-wingware-wiki** ()

Show the Wingware wiki for sharing scripts, tips, and tricks

**switch-document** (document\_name)

Switches to named document. Name may either be the complete name or the last path component of a path name. *Key Bindings: Emacs: Ctrl-X B; Visual Studio: Ctrl-K Ctrl-S*

**toggle-bookmark** ()

Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default. *Key Bindings: Normal: Ctrl-Alt-T; Emacs: Ctrl-X R T; Visual Studio: Ctrl-Alt-T; Eclipse (Experimental): Ctrl-Alt-T; OS X: Command-H*

**toggle-bookmark-at-click** ()

Set or remove a bookmark at the position in the editor where the most recent mouse click occurred. When set, the name of the bookmark is set to an auto-generated default.

**toolbar-search** (text, next=False, set\_anchor=True, forward=True)

Search using given text and the toolbar search area. The search is always forward from the current cursor or selection position

**toolbar-search-focus** ()

Move focus to toolbar search entry. *Key Bindings: Normal: Ctrl-D; Visual Studio: Ctrl-D; Eclipse (Experimental): Ctrl-D*

**toolbar-search-next** (set\_anchor=True)

Move to next match of text already entered in the toolbar search area

**toolbar-search-prev** (set\_anchor=True)

Move to previous match of text already entered in the toolbar search area

**vi-delete-bookmark** (marks)

Remove one or more bookmarks (pass in space separated list of names)

**vi-goto-bookmark** ()

Goto bookmark using single character name defined by the next pressed key *Key Bindings: VI/VIM: Grave*

**vi-set-bookmark** ()

Set a bookmark at current location on the editor using the next key press as the name of the bookmark. *Key Bindings: VI/VIM: m*

**wing-tips** ()

Display interactive tip manager

**write-changed-file-and-close** (filename)

Write current document to given location only if it contains any changes and close it. Writes to current file name if given filename is None.

**write-file** (filename, start\_line=None, end\_line=None, follow=True)

Write current file to a new location, optionally omitting all but the lines in the given range. The editor is changed to point to the new location when follow is True. Note that the editor contents will be truncated to the given start/end lines when follow is True. *Key Bindings: Emacs: Ctrl-X Ctrl-W*

**write-file-and-close** (filename)

Write current document to given location and close it. Saves to current file name if the given filename is None. *Key Bindings: VI/VIM: Shift-Z Shift-Z invokes write-file-and-close(filename=None)*

## Dock Window Commands

Commands for windows that contain dockable tool areas. These are available for the currently active window, if any.

**enter-fullscreen** ()

Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen` *Key Binding: Shift-F2*

**exit-fullscreen** ()

Restore previous non-fullscreen state of all tools and tool bar *Key Binding: Shift-F2*

**hide-horizontal-tools** ()

Hide the horizontal tool area

**hide-toolbar** ()

Hide toolbars in all document windows

**hide-vertical-tools** ()

Hide the vertical tool area

**minimize-horizontal-tools** ()

Minimize the horizontal tool area *Key Binding: F1*

**minimize-vertical-tools** ()

Minimize the vertical tool area *Key Binding: F2*

**show-horizontal-tools** ()

Show the horizontal tool area *Key Binding: F1*

**show-toolbar** ()

Show toolbars in all document windows

**show-vertical-tools** ()

Show the vertical tool area *Key Binding: F2*

**toggle-horizontal-tools** ()

Show or minimize the horizontal tool area

**toggle-vertical-tools** ()

Show or minimize the vertical tool area

## Document Viewer Commands

Commands for the documentation viewer. These are available when the documentation viewer has the keyboard focus.

### **copy** ()

Copy any selected text. *Key Bindings: Normal: Ctrl-Insert; VI/VIM: Ctrl-Insert; Emacs: Ctrl-Insert; Brief: Ctrl-Insert; Visual Studio: Ctrl-Insert; Eclipse (Experimental): Ctrl-Insert; OS X: Command-C*

### **document-back** ()

Go back to prior page in the history of those that have been viewed

### **document-contents** ()

Go to the document contents page

### **document-forward** ()

Go forward to next page in the history of those that have been viewed

### **document-next** ()

Go to the next page in the current document

### **document-previous** ()

Go to the previous page in the current document

### **isearch-backward** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search backward from the cursor position, optionally entering the given search string. *Key Bindings: Normal: Ctrl-Shift-U; Emacs: Ctrl-R; Visual Studio: Ctrl-Shift-U; Eclipse (Experimental): Ctrl-Shift-U; OS X: Command-Shift-U*

### **isearch-backward-regex** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string. *Key Bindings: VI/VIM: ?; Emacs: Ctrl-Alt-R*

### **isearch-forward** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search forward from the cursor position, optionally entering

the given search string. *Key Bindings: Normal: Ctrl-U; Emacs: Ctrl-S; Visual Studio: Ctrl-I; Eclipse (Experimental): Ctrl-U; OS X: Command-U*

**isearch-forward-regex** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string. *Key Bindings: VI/VIM: /; Emacs: Ctrl-Alt-S*

**isearch-repeat** (reverse=False, repeat=<numeric modifier; default=1>)

Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True. *Key Bindings: VI/VIM: n*

**isearch-sel-backward** (persist=True, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately. *Key Bindings: Normal: Ctrl-Shift-B; VI/VIM: # invokes isearch-sel-backward(persist=0, whole\_word=1); Emacs: Ctrl-C R; Visual Studio: Ctrl-Shift-B; Eclipse (Experimental): Ctrl-Shift-B*

**isearch-sel-forward** (persist=True, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately. *Key Bindings: Normal: Ctrl-B; VI/VIM: \* invokes isearch-sel-forward(persist=0, whole\_word=1); Emacs: Ctrl-C S; Visual Studio: Ctrl-B; Eclipse (Experimental): Ctrl-B*

**repeat-search-char** (opposite=0, repeat=<numeric modifier; default=1>)

Repeat the last search\_char operation, optionally in the opposite direction. *Key Bindings: VI/VIM: ;*

**search-char** (dir=1, pos=0, repeat=<numeric modifier; default=1>, single\_line=0)

Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single\_line=1 to search only within the current line. *Key Bindings: VI/VIM: F invokes search-char(dir=1, single\_line=1)*

## Global Documentation Commands

Commands for the documentation viewer that are available regardless of where the focus is.

**document-search** (txt=None)

Search all documentation.

## Window Commands

Commands for windows in general. These are available for the currently active window, if any.

**focus-current-editor** ()

Move focus back to the current editor, out of any tool, if there is an active editor. *Key Bindings: Eclipse (Experimental): F12*

**move-editor-focus** (dir=1, wrap=True)

Move focus to next or previous editor split, optionally wrapping when the end is reached. *Key Bindings: VI/VIM: Ctrl-W j invokes move-editor-focus(wrap=False); Emacs: Ctrl-X O*

**move-editor-focus-first** ()

Move focus to first editor split *Key Bindings: VI/VIM: Ctrl-W t*

**move-editor-focus-last** ()

Move focus to last editor split *Key Bindings: VI/VIM: Ctrl-W b*

**move-editor-focus-previous** ()

Move focus to previous editor split *Key Bindings: VI/VIM: Ctrl-W p*

**move-focus** ()

Move the keyboard focus forward within the Window to the next editable area *Key Binding: Shift-F1*

## Wing Tips Commands

Commands for the Wing Tips tool. These are only available when the tool is visible and has focus

**wingtips-close** ()

Close the Wing Tips window

**wingtips-contents ()**

Go to the Wing Tips contents page

**wingtips-next ()**

Go to the next page in Wing Tips

**wingtips-next-unseen ()**

Go to a next unseen Wing Tips page

**wingtips-previous ()**

Go to the previous page in Wing Tips

## 12.2. Project Manager Commands

### Project Manager Commands

These commands act on the project manager or on the current project, regardless of whether the project list has the keyboard focus.

**add-current-file-to-project ()**

Add the frontmost currently open file to project *Key Bindings: Normal: Ctrl-Shift-I; VI/VIM: Ctrl-Shift-I; Emacs: Ctrl-Shift-I; Brief: Ctrl-Shift-I; Visual Studio: Ctrl-Shift-I; Eclipse (Experimental): Ctrl-Shift-I; OS X: Command-Shift-I*

**add-directory-to-project** (loc=None, recursive=True, filter='\*', include\_hidden=False, gui=True)

Add directory to project.

**add-file-to-project ()**

Add an existing file to the project.

**browse-selected-from-project ()**

Browse file currently selected in the project manager

**clear-project-main-debug-file ()**

Clear main debug entry point to nothing, so that debugging runs the file in the current editor by default



**close-project** ()

Close currently open project file

**debug-selected-from-project** ()

Start debugging the file currently selected in the project manager

**execute-selected-from-project** ()

Execute the file currently selected in the project manager

**new-project** ()

Create a new project.

**open-ext-selected-from-project** ()

Open file currently selected in the project manager

**open-project** (filename=None)

Open the given project file, or prompt the user to select a file if the filename is not given.

**open-selected-from-project** ()

Open files currently selected in the project manager

**remove-directory-from-project** (loc=None, gui=True)

Remove directory from project.

**remove-selection-from-project** ()

Remove currently selected file or package from the project

**rescan-project-directories** (dirs=None, recursive=True)

Scan project directories for changes. If list of directories is not specified, currently selected directories are used.

**save-project** ()

Save project file.

**save-project-as** (filename=None)

Save project file under the given name, or prompt user for a name if the filename is not given.

**set-current-as-main-debug-file ()**

Set current editor file as the main debug entry point for this project

**set-selected-as-main-debug-file ()**

Set selected file as the main debug file for this project

**show-analysis-stats ()**

Show the effective Python version and path for the current configuration. This command name will be deprecated in Wing 5 and removed in Wing 6. Use `show-python-environment` in any new code or key bindings.

**show-current-file-in-project-tool ()**

Show the currently selected file in the project view, if present. The selection may be the current editor, if it has focus, or files selected in other views.

**show-project-window ()**

Raise the project manager window

**show-python-environment ()**

Show the effective Python version and path for the current configuration

**use-normal-project ()**

Store project in normal format

**use-shared-project ()**

Store project in sharable format

**view-directory-properties (loc=None)**

Show the project manager's directory properties dialog

**view-file-properties (loc=None, page=None, highlighted\_attribs=None)**

View project properties for a particular file (current file if none is given) *Key Bindings: Eclipse (Experimental): Alt-Enter; OS X: Command-I*

**view-project-as-flat-tree ()**

View project as flattened directory tree from project file

**view-project-as-tree ()**

View project as directory tree from project file

**view-project-properties** (highlighted\_attrib=None)

View or change project-wide properties *Key Bindings: Visual Studio: Alt-F7*

## Project View Commands

Commands that are available only when the project view has the keyboard focus.

**browse-selected-from-project** ()

Browse file currently selected in the project manager

**debug-selected-from-project** ()

Start debugging the file currently selected in the project manager

**execute-selected-from-project** ()

Execute the file currently selected in the project manager

**move-files-selected-in-project-to-trash** ()

Move the files and/or directories currently selected in the project view to the trash or recycling bin

**open-ext-selected-from-project** ()

Open file currently selected in the project manager

**open-selected-from-project** ()

Open files currently selected in the project manager

**remove-selection-from-project** ()

Remove currently selected file or package from the project

**rename-selected-in-project** (new\_name)

Rename the currently selected file or directory in the project view

**search-in-selected-from-project** ()

Search in file or directory currently selected in the project manager

**set-selected-as-main-debug-file** ()

Set selected file as the main debug file for this project

**view-project-as-flat-tree** ()

View project as flattened directory tree from project file

**view-project-as-tree** ()

View project as directory tree from project file

## 12.3. Editor Commands

### Editor Browse Mode Commands

Commands available only when the editor is in browse mode (used for VI bindings and possibly others)

**enter-insert-mode** (pos='before')

Enter editor insert mode *Key Bindings: VI/VIM: A invokes enter-insert-mode(pos="after")*

**enter-replace-mode** ()

Enter editor replace mode *Key Bindings: VI/VIM: Shift-R*

**enter-visual-mode** (unit='char')

Enter editor visual mode. Unit should be one of 'char', 'line', or 'block'.

**previous-select** ()

Turn on auto-select using previous mode and selection *Key Bindings: VI/VIM: g v*

**start-select-block** ()

Turn on auto-select block mode *Key Bindings: Normal: Shift-Ctrl-F8; VI/VIM: Shift-Ctrl-F8; Emacs: Shift-Ctrl-F8; Brief: Shift-Ctrl-F8; Visual Studio: Shift-Ctrl-F8; Eclipse (Experimental): Shift-Ctrl-F8; OS X: Shift-Command-F8*

**start-select-char** ()

Turn on auto-select mode character by character *Key Binding: Shift-F8*

**start-select-line** ()

Turn on auto-select mode line by line *Key Bindings: Normal: Ctrl-F8; VI/VIM: Ctrl-F8; Emacs: Ctrl-F8; Brief: Ctrl-F8; Visual Studio: Ctrl-F8; Eclipse (Experimental): Ctrl-F8; OS X: Command-F8*

### **vi-command-by-name ()**

Execute a VI command (implements “:” commands from VI) *Key Bindings: VI/VIM: :*

### **vi-set (command)**

Perform vi’s :set action. The command is the portion after :set. Currently supports ic, noic, ai, noai, number or nu, nonumber or nonu, ro, noro, sm, and nosm. Multiple options can be specied in one call as for :set ic sm ai

## **Editor Insert Mode Commands**

Commands available only when editor is in insert mode (used for VI bindings and possibly others)

### **enter-browse-mode (provisional=False)**

Enter editor browse mode *Key Bindings: VI/VIM: Esc*

## **Editor Non Modal Commands**

Commands available only when the editor is in non-modal editing mode

### **exit-visual-mode ()**

Exit visual mode and return back to default mode *Key Binding: Esc*

### **start-select-block ()**

Turn on auto-select block mode *Key Bindings: Normal: Shift-Ctrl-F8; VI/VIM: Shift-Ctrl-F8; Emacs: Shift-Ctrl-F8; Brief: Shift-Ctrl-F8; Visual Studio: Shift-Ctrl-F8; Eclipse (Experimental): Shift-Ctrl-F8; OS X: Shift-Command-F8*

### **start-select-char ()**

Turn on auto-select mode character by character *Key Binding: Shift-F8*

### **start-select-line ()**

Turn on auto-select mode line by line *Key Bindings: Normal: Ctrl-F8; VI/VIM: Ctrl-*

*F8; Emacs: Ctrl-F8; Brief: Ctrl-F8; Visual Studio: Ctrl-F8; Eclipse (Experimental): Ctrl-F8; OS X: Command-F8*

## Editor Panel Commands

Commands that control splitting up an editor panel. These are available when one split in the editor panel has the keyboard focus.

**split-horizontally** (new=0)

Split current view horizontally. *Key Bindings: VI/VIM: Ctrl-W v; Emacs: Ctrl-X 3*

**split-horizontally-open-file** (filename)

Split current view horizontally and open selected file

**split-vertically** (new=0)

Split current view vertically. Create new editor in new view when new==1. *Key Bindings: VI/VIM: Ctrl-W s; Emacs: Ctrl-X 2; Brief: F3*

**split-vertically-open-file** (filename)

Split current view vertically and open selected file

**unsplit** (action='current')

Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  -- Change to recent buffer before unsplitting
recent-or-close -- Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future. *Key Bindings: VI/VIM: Ctrl-W q invokes unsplit(action="close"); Emacs: Ctrl-X 1; Brief: F4*

## Editor Replace Mode Commands

Commands available only when editor is in replace mode (used for VI bindings and possibly others)

**enter-browse-mode** (provisional=False)

Enter editor browse mode *Key Bindings: VI/VIM: Esc*

## Editor Split Commands

Commands for a particular editor split, available when the editor in that split has the keyboard focus. Additional commands affecting the editor's content are defined separately.

**activate-file-option-menu** ()

Activate the file menu for the editor. *Key Bindings: Normal: Ctrl-1; VI/VIM: Ctrl-1; Emacs: Ctrl-1; Brief: Ctrl-1; Visual Studio: Ctrl-1; Eclipse (Experimental): Ctrl-1; OS X: Command-1*

**grow-split-horizontally** ()

Increase width of this split

**grow-split-vertically** ()

Increase height of this split *Key Bindings: VI/VIM: Ctrl-W +*

**shrink-split-horizontally** ()

Decrease width of this split

**shrink-split-vertically** ()

Decrease height of this split *Key Bindings: VI/VIM: Ctrl-W -*

**visit-history-next** ()

Move forward in history to next visited editor position *Key Bindings: Normal: Alt-Right; VI/VIM: Alt-Right; Emacs: Alt-Right; Brief: Alt-Right; Visual Studio: Alt-Right; Eclipse (Experimental): Alt-Right; OS X: Ctrl-*

**visit-history-previous** ()

Move back in history to previous visited editor position *Key Bindings: Normal: Alt-Left;*

*VI/VIM: Alt-Left; Emacs: Alt-Left; Brief: Alt-Left; Visual Studio: Alt-Left; Eclipse (Experimental): Alt-Left; OS X: Ctrl-Comma*

## Editor Visual Mode Commands

Commands available only when the editor is in visual mode (used for VI bindings and some others)

### **enter-browse-mode** ()

Enter editor browse mode *Key Bindings: VI/VIM: Esc*

### **enter-insert-mode** (pos='delete-sel')

Enter editor insert mode *Key Bindings: VI/VIM: A invokes enter-insert-mode(pos="after")*

### **enter-visual-mode** (unit='char')

Alter type of editor visual mode or exit back to browse mode. Unit should be one of 'char', 'line', or 'block'.

### **exit-visual-mode** ()

Exit visual mode and return back to default mode *Key Binding: Esc*

### **vi-command-by-name** ()

Execute a VI command (implements ":" commands from VI) *Key Bindings: VI/VIM: :*

## Active Editor Commands

Commands that only apply to editors when they have the keyboard focus. These commands are also available for the Python Shell, Debug Probe, and Debug I/O tools, which subclass the source editor, although some of the commands are modified or disabled as appropriate in those contexts.

### **activate-symbol-option-menu-1** ()

Activate the 1st symbol menu for the editor. *Key Bindings: Normal: Ctrl-2; VI/VIM: Ctrl-2; Emacs: Ctrl-2; Brief: Ctrl-2; Visual Studio: Ctrl-2; Eclipse (Experimental): Ctrl-2; OS X: Command-2*

### **activate-symbol-option-menu-2** ()



Activate the 2nd symbol menu for the editor. *Key Bindings: Normal: Ctrl-3; VI/VIM: Ctrl-3; Emacs: Ctrl-3; Brief: Ctrl-3; Visual Studio: Ctrl-3; Eclipse (Experimental): Ctrl-3; OS X: Command-3*

### **activate-symbol-option-menu-3** ()

Activate the 3rd symbol menu for the editor. *Key Bindings: Normal: Ctrl-4; VI/VIM: Ctrl-4; Emacs: Ctrl-4; Brief: Ctrl-4; Visual Studio: Ctrl-4; Eclipse (Experimental): Ctrl-4; OS X: Command-4*

### **activate-symbol-option-menu-4** ()

Activate the 4th symbol menu for the editor. *Key Bindings: Normal: Ctrl-5; VI/VIM: Ctrl-5; Emacs: Ctrl-5; Brief: Ctrl-5; Visual Studio: Ctrl-5; Eclipse (Experimental): Ctrl-5; OS X: Command-5*

### **activate-symbol-option-menu-5** ()

Activate the 5th symbol menu for the editor. *Key Bindings: Normal: Ctrl-6; VI/VIM: Ctrl-6; Emacs: Ctrl-6; Brief: Ctrl-6; Visual Studio: Ctrl-6; Eclipse (Experimental): Ctrl-6; OS X: Command-6*

### **backward-char** (wrap=1, repeat=<numeric modifier; default=1>)

Move cursor backward one character *Key Bindings: Normal: Left; VI/VIM: Left; Emacs: Left; Brief: Left; Visual Studio: Left; Eclipse (Experimental): Left; OS X: Option-Up*

### **backward-char-extend** (wrap=1, repeat=<numeric modifier; default=1>)

Move cursor backward one character, adjusting the selection range to new position *Key Binding: Shift-Left*

### **backward-char-extend-rect** (wrap=1, repeat=<numeric modifier; default=1>)

Move cursor backward one character, adjusting the rectangular selection range to new position *Key Bindings: Normal: Shift-Alt-Left; VI/VIM: Shift-Alt-Left; Emacs: Shift-Alt-Left; Brief: Shift-Alt-Left; Visual Studio: Shift-Alt-Left; Eclipse (Experimental): Shift-Alt-Left; OS X: Ctrl-Option-Left*

### **backward-delete-char** (repeat=<numeric modifier; default=1>)

Delete one character behind the cursor, or the current selection if not empty. *Key Bindings: Normal: BackSpace; VI/VIM: BackSpace; Emacs: BackSpace; Brief: BackSpace; Visual Studio: BackSpace; Eclipse (Experimental): BackSpace; OS X: Backspace*

### **backward-delete-word** (delimiters=None, repeat=<numeric modifier; default=1>)

Delete one word behind of the cursor *Key Bindings: Normal: Ctrl-BackSpace; VI/VIM: Ctrl-BackSpace; Emacs: Ctrl-BackSpace; Brief: Ctrl-BackSpace; Visual Studio: Ctrl-BackSpace; Eclipse (Experimental): Ctrl-BackSpace; OS X: Ctrl-Option-Delete*

**backward-page** (repeat=<numeric modifier; default=1>)

Move cursor backward one page *Key Bindings: Normal: Prior; VI/VIM: Prior; Emacs: Prior; Brief: Prior; Visual Studio: Prior; Eclipse (Experimental): Prior; OS X: Ctrl-Up*

**backward-page-extend** (repeat=<numeric modifier; default=1>)

Move cursor backward one page, adjusting the selection range to new position *Key Bindings: Normal: Shift-Prior; VI/VIM: Shift-Prior; Emacs: Shift-Prior; Brief: Shift-Prior; Visual Studio: Shift-Prior; Eclipse (Experimental): Shift-Prior; OS X: Shift-Page\_Up*

**backward-paragraph** (repeat=<numeric modifier; default=1>)

Move cursor backward one paragraph (to next all-whitespace line). *Key Bindings: VI/VIM: {*

**backward-paragraph-extend** (repeat=<numeric modifier; default=1>)

Move cursor backward one paragraph (to next all-whitespace line), adjusting the selection range to new position.

**backward-tab** ()

Outdent line at current position *Key Binding: Shift-Tab*

**backward-word** (delimiters=None, gravity='start', repeat=<numeric modifier; default=1>)

Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word. *Key Bindings: Normal: Ctrl-Left; VI/VIM: Ctrl-Left; Emacs: Ctrl-Left; Brief: Ctrl-Left; Visual Studio: Ctrl-Left; Eclipse (Experimental): Ctrl-Left; OS X: Option-Left*

**backward-word-extend** (delimiters=None, gravity='start', repeat=<numeric modifier; default=1>)

Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word. *Key Bindings: Normal: Ctrl-Shift-Left; VI/VIM: Ctrl-Shift-Left; Emacs: Ctrl-Shift-Left; Brief: Ctrl-Shift-Left; Visual Studio: Ctrl-Shift-Left; Eclipse (Experimental): Ctrl-Shift-Left; OS X: Option-Shift-Left*

**beginning-of-line** (toggle=True)

Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa). *Key Bindings:* VI/VIM: *0* invokes *beginning-of-line(toggle=0)*; Emacs: *Home*; Brief: *Shift-Home*; OS X: *Command-Left*

**beginning-of-line-extend** (toggle=True)

Move to beginning of current line, adjusting the selection range to the new position. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa). *Key Bindings:* Emacs: *Shift-Home*; OS X: *Ctrl-Shift-Left*

**beginning-of-line-text** (toggle=True)

Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa). *Key Bindings:* Normal: *Home*; VI/VIM: *Home*; Emacs: *Home*; Brief: *Home*; Visual Studio: *Home*; Eclipse (Experimental): *Home*

**beginning-of-line-text-extend** (toggle=True)

Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa). *Key Bindings:* Normal: *Shift-Home*; VI/VIM: *Shift-Home*; Emacs: *Shift-Home*; Brief: *Shift-Home*; Visual Studio: *Shift-Home*; Eclipse (Experimental): *Shift-Home*

**beginning-of-screen-line** ()

Move to beginning of current wrapped line *Key Bindings:* VI/VIM: *g 0*

**beginning-of-screen-line-extend** ()

Move to beginning of current wrapped line, extending selection

**beginning-of-screen-line-text** ()

Move to first non-blank character at beginning of current wrapped line *Key Bindings:* VI/VIM: *g ^*

**beginning-of-screen-line-text-extend** ()

Move to first non-blank character at beginning of current wrapped line, extending selection

**brace-match** ()

Match brace at current cursor position, selecting all text between the two and highlighting the braces *Key Bindings: Normal: Ctrl-E; Emacs: Ctrl-M; Visual Studio: Ctrl-E; Eclipse (Experimental): Ctrl-E; OS X: Command-B*

**cancel** ()

Cancel current editor command

**cancel-autocompletion** ()

Cancel any active autocompletion.

**case-lower** (repeat=<numeric modifier; default=1>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to lower case *Key Bindings: Visual Studio: Ctrl-U*

**case-lower-next-move** (repeat=<numeric modifier; default=1>)

Change case of text spanned by next cursor movement to lower case *Key Bindings: VI/VIM: g u*

**case-swap** (repeat=<numeric modifier; default=1>)

Change case of the current selection, or character ahead of the cursor if there is no selection, so each letter is the opposite of its current case *Key Bindings: VI/VIM: ~*

**case-swap-next-move** (repeat=<numeric modifier; default=1>)

Change case of text spanned by next cursor movement so each letter is the opposite of its current case *Key Bindings: VI/VIM: g ~*

**case-title** (repeat=<numeric modifier; default=1>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to title case (first letter of each word capitalized)

**case-title-next-move** (repeat=<numeric modifier; default=1>)

Change case of text spanned by next cursor movement to title case (first letter of each word capitalized)

**case-upper** (repeat=<numeric modifier; default=1>)

Change case of the current selection, or character ahead of the cursor if there is no selection, to upper case *Key Bindings: Visual Studio: Ctrl-Shift-U*

**case-upper-next-move** (repeat=<numeric modifier; default=1>)

Change case of text spanned by next cursor movement to upper case *Key Bindings: VI/VIM: g Shift-U*

**center-cursor** ()

Scroll so cursor is centered on display *Key Bindings: VI/VIM: z .; Emacs: Ctrl-L; Brief: Ctrl-C*

**clear** ()

Clear selected text

**clear-move-command** ()

Clear any pending move command action, as for VI mode *Key Bindings: VI/VIM: Esc*

**complete-autocompletion** (append=)

Complete the current active autocompletion.

**copy** ()

Copy selected text *Key Bindings: Normal: Ctrl-Insert; VI/VIM: Ctrl-Insert; Emacs: Ctrl-Insert; Brief: Ctrl-Insert; Visual Studio: Ctrl-Insert; Eclipse (Experimental): Ctrl-Insert; OS X: Command-C*

**copy-line** ()

Copy the current lines(s) to clipboard *Key Bindings: Brief: KP\_Add*

**copy-range** (start\_line, end\_line, target\_line)

Copy the given range of lines to the given target line. Copies to current line if target\_line is '.'.

**copy-selection-or-line** ()

Copy the current selection or current line if there is no selection. The text is placed on the clipboard.

**cursor-move-to-bottom** (offset=<numeric modifier; default=0>)

Move cursor to bottom of display (without scrolling), optionally at an offset of given number of lines before bottom *Key Bindings: VI/VIM: Shift-L*

**cursor-move-to-center** ()

Move cursor to center of display (without scrolling) *Key Bindings: VI/VIM: Shift-M*

**cursor-move-to-top** (offset=<numeric modifier; default=0>)

Move cursor to top of display (without scrolling), optionally at an offset of given number of lines below top *Key Bindings: VI/VIM: Shift-H*

### **cursor-to-bottom** ()

Scroll so cursor is centered at bottom of display *Key Bindings: VI/VIM: z -*

### **cursor-to-top** ()

Scroll so cursor is centered at top of display *Key Bindings: VI/VIM: z Return*

### **cut** ()

Cut selected text *Key Bindings: Normal: Shift-Delete; VI/VIM: Shift-Delete; Emacs: Shift-Delete; Brief: Shift-Delete; Visual Studio: Shift-Delete; Eclipse (Experimental): Shift-Delete; OS X: Command-X*

### **cut-line** ()

Cut the current line(s) to clipboard. *Key Bindings: Brief: KP\_Subtract; Visual Studio: Ctrl-L*

### **cut-selection-or-line** ()

Cut the current selection or current line if there is no selection. The text is placed on the clipboard. *Key Bindings: Visual Studio: Shift-Delete*

### **delete-line** (repeat=<numeric modifier; default=1>)

Delete the current line or lines when the selection spans multiple lines or given repeat is > 1 *Key Bindings: Normal: Ctrl-Shift-C; Eclipse (Experimental): Ctrl-Shift-C*

### **delete-line-insert** (repeat=<numeric modifier; default=1>)

Delete the current line or lines when the selection spans multiple lines or given repeat is > 1. Enters insert mode (when working with modal key bindings). *Key Bindings: VI/VIM: Shift-S*

### **delete-next-move** (repeat=<numeric modifier; default=1>)

Delete the text covered by the next cursor move command. *Key Bindings: VI/VIM: d*

### **delete-next-move-insert** (repeat=<numeric modifier; default=1>)

Delete the text covered by the next cursor move command and then enter insert mode (when working in a modal editor key binding) *Key Bindings: VI/VIM: c*

### **delete-range** (start\_line, end\_line, register=None)

Delete given range of lines, copying them into given register (or currently selected default register if register is None)

**delete-to-end-of-line** (repeat=<numeric modifier; default=1>, post\_offset=0)

Delete everything between the cursor and end of line *Key Bindings: VI/VIM: Shift-D invokes delete-to-end-of-line(post\_offset=-1)*

**delete-to-end-of-line-insert** (repeat=<numeric modifier; default=1>)

Delete everything between the cursor and end of line and enter insert move (when working in a modal editor key binding) *Key Bindings: VI/VIM: Shift-C*

**delete-to-start-of-line** ()

Delete everything between the cursor and start of line *Key Bindings: VI/VIM: Ctrl-U*

**duplicate-line** (pos='below')

Duplicate the current line or lines. Places the duplicate on the line following the selection if pos is 'below' or before the selection if it is 'above'. *Key Bindings: Normal: Ctrl-Shift-V; Eclipse (Experimental): Ctrl-Shift-V*

**duplicate-line-above** ()

Duplicate the current line or lines above the selection. *Key Bindings: Normal: Ctrl-Shift-Y; Eclipse (Experimental): Ctrl-Shift-Y*

**enclose** (start='(', end=')')

Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text. *Key Bindings: Normal: Alt-( invokes enclose(start="(", end=")"); VI/VIM: Alt-( invokes enclose(start="(", end=")"); Emacs: Alt-( invokes enclose(start="(", end=")"); Brief: Alt-( invokes enclose(start="(", end=")"); Visual Studio: Alt-( invokes enclose(start="(", end=")"); Eclipse (Experimental): Alt-( invokes enclose(start="(", end=")"))*

**end-of-document** ()

Move cursor to end of document *Key Bindings: Normal: Ctrl-End; VI/VIM: Ctrl-End; Emacs: Ctrl-End; Brief: Ctrl-End; Visual Studio: Ctrl-End; Eclipse (Experimental): Ctrl-End; OS X: Command-Down*

**end-of-document-extend** ()

Move cursor to end of document, adjusting the selection range to new position *Key Bindings: Normal: Ctrl-Shift-End; VI/VIM: Ctrl-Shift-End; Emacs: Ctrl-Shift-End;*

*Brief: Ctrl-Shift-End; Visual Studio: Ctrl-Shift-End; Eclipse (Experimental): Ctrl-Shift-End; OS X: Command-Shift-Down*

**end-of-line** (count=<numeric modifier; default=1>)

Move to end of current line *Key Bindings: Normal: End; VI/VIM: End; Emacs: End; Brief: End; Visual Studio: End; Eclipse (Experimental): End; OS X: Ctrl-Right*

**end-of-line-extend** (count=<numeric modifier; default=1>)

Move to end of current line, adjusting the selection range to new position *Key Bindings: Normal: Shift-End; VI/VIM: Shift-End; Emacs: Shift-End; Brief: Shift-End; Visual Studio: Shift-End; Eclipse (Experimental): Shift-End; OS X: Ctrl-Shift-Right*

**end-of-screen-line** (count=<numeric modifier; default=1>)

Move to end of current wrapped line *Key Bindings: VI/VIM: g \$*

**end-of-screen-line-extend** (count=<numeric modifier; default=1>)

Move to end of current wrapped line, extending selection

**exchange-point-and-mark** ()

When currently marking text, this exchanges the current position and mark ends of the current selection *Key Bindings: VI/VIM: o; Emacs: Ctrl-X Ctrl-X*

**filter-next-move** (repeat=<numeric modifier; default=1>)

Filter the lines covered by the next cursor move command through an external command and replace the lines with the result *Key Bindings: VI/VIM: !*

**filter-range** (cmd, start\_line=0, end\_line=-1)

Filter a range of lines in the editor through an external command and replace the lines with the result. Filters the whole file by default. Filters nothing and opens up a scratch buffer with the output of the command if start\_line and end\_line are both -1.

**filter-selection** (cmd)

Filter the current selection through an external command and replace the lines with the result *Key Bindings: VI/VIM: !*

**form-feed** ()

Place a form feed character at the current cursor position

**forward-char** (wrap=1, repeat=<numeric modifier; default=1>)



Move cursor forward one character *Key Binding: Right*

**forward-char-extend** (wrap=1, repeat=<numeric modifier; default=1>)

Move cursor forward one character, adjusting the selection range to new position *Key Binding: Shift-Right*

**forward-char-extend-rect** (wrap=1, repeat=<numeric modifier; default=1>)

Move cursor forward one character, adjusting the rectangular selection range to new position *Key Bindings: Normal: Shift-Alt-Right; VI/VIM: Shift-Alt-Right; Emacs: Shift-Alt-Right; Brief: Shift-Alt-Right; Visual Studio: Shift-Alt-Right; Eclipse (Experimental): Shift-Alt-Right; OS X: Ctrl-Option-Right*

**forward-delete-char** (repeat=<numeric modifier; default=1>)

Delete one character in front of the cursor *Key Binding: Delete*

**forward-delete-char-insert** (repeat=<numeric modifier; default=1>)

Delete one char in front of the cursor and enter insert mode (when working in modal key bindings) *Key Bindings: VI/VIM: s*

**forward-delete-char-within-line** (repeat=<numeric modifier; default=1>)

Delete one character in front of the cursor unless at end of line, in which case delete backward. Do nothing if the line is empty. This is VI style 'x' in browser mode. *Key Bindings: VI/VIM: x*

**forward-delete-word** (delimiters=None, repeat=<numeric modifier; default=1>)

Delete one word in front of the cursor *Key Bindings: Normal: Ctrl-Delete; VI/VIM: Ctrl-Delete; Emacs: Ctrl-Delete; Brief: Ctrl-Delete; Visual Studio: Ctrl-Delete; Eclipse (Experimental): Ctrl-Delete; OS X: Option-Delete*

**forward-delete-word-insert** (delimiters=None, repeat=<numeric modifier; default=1>)

Delete one word in front of the cursor and enter insert mode (when working in modal key bindings)

**forward-page** (repeat=<numeric modifier; default=1>)

Move cursor forward one page *Key Bindings: Normal: Next; VI/VIM: Next; Emacs: Next; Brief: Next; Visual Studio: Next; Eclipse (Experimental): Next; OS X: Ctrl-Down*

**forward-page-extend** (repeat=<numeric modifier; default=1>)

Move cursor forward one page, adjusting the selection range to new position *Key Bind-*

*ings: Normal: Shift-Next; VI/VIM: Shift-Next; Emacs: Shift-Next; Brief: Shift-Next; Visual Studio: Shift-Next; Eclipse (Experimental): Shift-Next; OS X: Shift-Page\_Down*

**forward-paragraph** (repeat=<numeric modifier; default=1>)

Move cursor forward one paragraph (to next all-whitespace line). *Key Bindings: VI/VIM: }*

**forward-paragraph-extend** (repeat=<numeric modifier; default=1>)

Move cursor forward one paragraph (to next all-whitespace line), adjusting the selection range to new position.

**forward-tab** ()

Place a tab character at the current cursor position *Key Binding: Ctrl-T*

**forward-word** (delimiters=None, gravity='start', repeat=<numeric modifier; default=1>)

Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word. *Key Bindings: Normal: Ctrl-Right; VI/VIM: Ctrl-Right; Emacs: Ctrl-Right; Brief: Ctrl-Right; Visual Studio: Ctrl-Right; Eclipse (Experimental): Ctrl-Right; OS X: Option-Right*

**forward-word-extend** (delimiters=None, gravity='start', repeat=<numeric modifier; default=1>)

Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word. *Key Bindings: Normal: Ctrl-Shift-Right; VI/VIM: Ctrl-Shift-Right; Emacs: Ctrl-Shift-Right; Brief: Ctrl-Shift-Right; Visual Studio: Ctrl-Shift-Right; Eclipse (Experimental): Ctrl-Shift-Right; OS X: Option-Shift-Right*

**goto-overridden-method** ()

Goes to the method that is overridden by the current method

**hide-selection** ()

Turn off display of the current text selection

**indent-to-match** (toggle=False)

Indent the current line or selected region to match indentation of preceding non-blank

line. Set `toggle=True` to indent instead of one level higher if already at the matching position. *Key Binding: Ctrl=*

### **indent-to-next-indent-stop** ()

Indent to next indent stop from the current position. Acts like indent command if selection covers multiple lines.

### **isearch-backward** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search backward from the cursor position, optionally entering the given search string *Key Bindings: Normal: Ctrl-Shift-U; Emacs: Ctrl-R; Visual Studio: Ctrl-Shift-U; Eclipse (Experimental): Ctrl-Shift-U; OS X: Command-Shift-U*

### **isearch-backward-regex** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string *Key Bindings: VI/VIM: ?; Emacs: Ctrl-Alt-R*

### **isearch-forward** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search forward from the cursor position, optionally entering the given search string *Key Bindings: Normal: Ctrl-U; Emacs: Ctrl-S; Visual Studio: Ctrl-I; Eclipse (Experimental): Ctrl-U; OS X: Command-U*

### **isearch-forward-regex** (search\_string=None, repeat=<numeric modifier; default=1>)

Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string *Key Bindings: VI/VIM: /; Emacs: Ctrl-Alt-S*

### **isearch-repeat** (reverse=False, repeat=<numeric modifier; default=1>)

Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True. *Key Bindings: VI/VIM: n*

### **isearch-sel-backward** (persist=True, whole\_word=False, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately. *Key Bindings: Normal: Ctrl-Shift-B; VI/VIM: # invokes isearch-sel-backward(persist=0, whole\_word=1); Emacs: Ctrl-C R; Visual Studio: Ctrl-Shift-B; Eclipse (Experimental): Ctrl-Shift-B*

### **isearch-sel-forward** (persist=True, whole\_word=False, repeat=<numeric modifier; default=1>)

Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately. *Key Bindings: Normal: Ctrl-B; VI/VIM: \* invokes isearch-select-forward(persist=0, whole\_word=1); Emacs: Ctrl-C S; Visual Studio: Ctrl-B; Eclipse (Experimental): Ctrl-B*

### **kill-line ()**

Kill rest of line from cursor to end of line, and place it into the clipboard with any other contiguously removed lines. End-of-line is removed only if there is nothing between the cursor and the end of the line. *Key Bindings: Emacs: Ctrl-K; Brief: Alt-D; OS X: Ctrl-k*

### **middle-of-screen-line ()**

Move to middle of current wrapped line *Key Bindings: VI/VIM: g m*

### **middle-of-screen-line-extend ()**

Move to middle of current wrapped line, extending selection

### **move-line-down** (indent=True, repeat=<numeric modifier; default=1>)

Move the current line or lines up down line, optionally indenting to match the new position *Key Bindings: Normal: Ctrl-Shift-Down; Eclipse (Experimental): Ctrl-Shift-Down*

### **move-line-up** (indent=True, repeat=<numeric modifier; default=1>)

Move the current line or lines up one line, optionally indenting to match the new position *Key Bindings: Normal: Ctrl-Shift-Up; Eclipse (Experimental): Ctrl-Shift-Up*

### **move-range** (start\_line, end\_line, target\_line)

Move the given range of lines to the given target line. Moves to current line if target\_line is '.'.

### **move-to-register** (unit='char', cut=0, num=<numeric modifier; default=1>)

Cut or copy a specified number of characters or lines, or the current selection. Set `cut=1` to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection. *Key Bindings: VI/VIM: Shift-Y invokes move-to-register(unit="line")*

### **move-to-register-next-move** (cut=0, repeat=<numeric modifier; default=1>)

Move the text spanned by the next cursor motion to a register *Key Bindings: VI/VIM: y*

**new-line ()**

Place a new line at the current cursor position *Key Binding: Return*

**new-line-after ()**

Place a new line after the current line *Key Bindings: Normal: Ctrl-Return; VI/VIM: Ctrl-Return; Emacs: Ctrl-Return; Brief: Ctrl-Return; Visual Studio: Ctrl-Return; Eclipse (Experimental): Ctrl-Return*

**new-line-before ()**

Place a new line before the current line *Key Bindings: Normal: Shift-Return; VI/VIM: Shift-Return; Emacs: Shift-Return; Brief: Shift-Return; Visual Studio: Shift-Return; Eclipse (Experimental): Shift-Return*

**next-blank-line** (threshold=0, repeat=<numeric modifier; default=1>)

Move to the next blank line in the file, if any. If threshold>0 then a line is considered blank if it contains less than that many characters after leading and trailing whitespace are removed. *Key Bindings: Emacs: Alt-}* invokes *next-blank-line(threshold=1)*

**next-block** (count=1, ignore\_indented=True)

Select the next block. Will ignore indented blocks under the current block unless ignore\_indented is False. Specify a count of more than 1 to go forward multiple blocks.

**next-line** (cursor='same', repeat=<numeric modifier; default=1>)

Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char. *Key Binding: Down*

**next-line-extend** (cursor='same', repeat=<numeric modifier; default=1>)

Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char. *Key Binding: Shift-Down*

**next-line-extend-rect** (cursor='same', repeat=<numeric modifier; default=1>)

Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char. *Key Bindings: Normal: Shift-Alt-Down; VI/VIM: Shift-Alt-Down; Emacs: Shift-Alt-Down; Brief: Shift-Alt-Down; Visual Studio: Shift-Alt-Down; Eclipse (Experimental): Shift-Alt-Down; OS X: Ctrl-Option-Down*

**next-line-in-file** (cursor='start', repeat=<numeric modifier; default=1>)

Move to next line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fnb' for first non-blank char. *Key Bindings: VI/VIM: + invokes next-line-in-file(cursor="fnb")*

**next-scope** (count=1, sibling\_only=False)

Select the next scope. Specify a count of more than 1 to go forward multiple scopes. If sibling\_only is true, move only to other scopes of the same parent. *Key Bindings: Eclipse (Experimental): Ctrl-Shift-Down*

**next-statement** (count=1, ignore\_indented=True)

Select the next statement. Will ignore indented statements under the current statements unless ignore\_indented is False. Specify a count of more than 1 to go forward multiple statements. *Key Bindings: Eclipse (Experimental): Alt-Shift-Right*

**open-line** ()

Open the current line by inserting a newline after the caret *Key Bindings: Emacs: Ctrl-O*

**paste** ()

Paste text from clipboard *Key Bindings: Normal: Shift-Insert; VI/VIM: Shift-Insert; Emacs: Shift-Insert; Brief: Shift-Insert; Visual Studio: Shift-Insert; Eclipse (Experimental): Shift-Insert; OS X: Ctrl-y*

**paste-register** (pos=1, indent=0, cursor=-1)

Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes. *Key Bindings: VI/VIM: p*

**previous-blank-line** (threshold=0, repeat=<numeric modifier; default=1>)

Move to the previous blank line in the file, if any. If threshold>0 then a line is considered blank if it contains less than that many characters after leading and trailing whitespace are removed. *Key Bindings: Emacs: Alt-{ invokes previous-blank-line(threshold=1)*

**previous-block** (count=1, ignore\_indented=True)

Select the previous block. Will ignore indented blocks under the current block unless `ignore_indented` is `False`. Specify a count of more than 1 to go backward multiple blocks.

**previous-line** (`cursor='same'`, `repeat=<numeric modifier; default=1>`)

Move to previous screen line, optionally repositioning character within line: `'same'` to leave in same horizontal position, `'start'` at start, `'end'` at end, or `'fmb'` for first non-blank char. *Key Binding: Up*

**previous-line-extend** (`cursor='same'`, `repeat=<numeric modifier; default=1>`)

Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: `'same'` to leave in same horizontal position, `'start'` at start, `'end'` at end, or `'fmb'` for first non-blank char. *Key Binding: Shift-Up*

**previous-line-extend-rect** (`cursor='same'`, `repeat=<numeric modifier; default=1>`)

Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: `'same'` to leave in same horizontal position, `'start'` at start, `'end'` at end, or `'fmb'` for first non-blank char. *Key Bindings: Normal: Shift-Alt-Up; VI/VIM: Shift-Alt-Up; Emacs: Shift-Alt-Up; Brief: Shift-Alt-Up; Visual Studio: Shift-Alt-Up; Eclipse (Experimental): Shift-Alt-Up; OS X: Ctrl-Option-Up*

**previous-line-in-file** (`cursor='start'`, `repeat=<numeric modifier; default=1>`)

Move to previous line in file, repositioning character within line: `'start'` at start, `'end'` at end, or `'fmb'` for first non-blank char. *Key Bindings: VI/VIM: - invokes previous-line-in-file(cursor="fmb")*

**previous-scope** (`count=1`, `sibling_only=False`)

Select the previous scope. Specify a count of more than 1 to go backward multiple scopes. If `sibling_only` is `true`, move only to other scopes of the same parent. *Key Bindings: Eclipse (Experimental): Ctrl-Shift-Up*

**previous-statement** (`count=1`, `ignore_indented=True`)

Select the previous statement. Will ignore indented statements under the current statements unless `ignore_indented` is `False`. Specify a count of more than 1 to go back multiple statements. *Key Bindings: Eclipse (Experimental): Alt-Shift-Left*

**profile-editor-start** ()

Turn on profiling for the current source editor

**profile-editor-stop** ()

Stop profiling and print stats to stdout

**reanalyze-file** ()

Rescan file for code analysis.

**redo** ()

Redo last action *Key Bindings: Normal: Ctrl-Y; VI/VIM: Ctrl-R; Emacs: Ctrl-.; Brief: Ctrl-U; Visual Studio: Ctrl-Y; Eclipse (Experimental): Ctrl-Y; OS X: Command-Shift-Z*

**repeat-command** (repeat=<numeric modifier; default=1>)

Repeat the last editor command *Key Bindings: VI/VIM: .*

**repeat-search-char** (opposite=0, repeat=<numeric modifier; default=1>)

Repeat the last search\_char operation, optionally in the opposite direction. *Key Bindings: VI/VIM: ;*

**rstrip-each-line** ()

Strip trailing whitespace from each line.

**scroll-text-down** (repeat=<numeric modifier; default=1>, move\_cursor=True)

Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line. *Key Bindings: Normal: Ctrl-Down; VI/VIM: Ctrl-Down; Emacs: Ctrl-Down; Brief: Ctrl-Down; Visual Studio: Ctrl-Down; Eclipse (Experimental): Ctrl-Down*

**scroll-text-left** (repeat=<numeric modifier; default=1>)

Scroll text left a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen. *Key Bindings: VI/VIM: z l*

**scroll-text-page-down** (repeat=<numeric modifier; default=1>, move\_cursor=True)

Scroll text down a page w/o moving cursor's relative position on screen. Repeat is number of pages or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**scroll-text-page-up** (repeat=<numeric modifier; default=1>, move\_cursor=True)

Scroll text up a page w/o moving cursor's relative position on screen. Repeat is number of pages or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave



cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**scroll-text-right** (repeat=<numeric modifier; default=1>)

Scroll text right a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen. *Key Bindings: VI/VIM: z h*

**scroll-text-up** (repeat=<numeric modifier; default=1>, move\_cursor=True)

Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line. *Key Bindings: Normal: Ctrl-Up; VI/VIM: Ctrl-Up; Emacs: Ctrl-Up; Brief: Ctrl-Up; Visual Studio: Ctrl-Up; Eclipse (Experimental): Ctrl-Up*

**scroll-to-cursor** ()

Scroll to current cursor position, if not already visible

**search-char** (dir=1, pos=0, repeat=<numeric modifier; default=1>, single\_line=0)

Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single\_line=1 to search only within the current line. *Key Bindings: VI/VIM: F invokes search-char(dir=1, single\_line=1)*

**select-all** ()

Select all text in the editor *Key Bindings: Normal: Ctrl-A; Visual Studio: Ctrl-A; Eclipse (Experimental): Ctrl-A; OS X: Command-A*

**select-block** ()

Select the block the cursor is in.

**select-less** ()

Select less code; undoes the last select-more command *Key Bindings: Eclipse (Experimental): Alt-Shift-Down*

**select-lines** ()

Select the current line or lines

**select-more** ()

Select more code on either the current line or larger multi-line blocks. *Key Bindings: Eclipse (Experimental): Alt-Shift-Up*

### **select-scope ()**

Select the scope the cursor is in.

### **select-statement ()**

Select the statement the cursor is in.

### **set-mark-command (unit='char')**

Set start of text marking for selection at current cursor position. Subsequently, all cursor move operations will automatically extend the text selection until stop-mark-command is issued. Unit defines what is selected: can be one of char, line, or block (rectangle). *Key Bindings: Emacs: Ctrl-Space*

### **set-register ()**

Set the register to use for subsequent cut/copy/paste operations *Key Bindings: VI/VIM: “*

### **show-autocompleter ()**

Show the auto-completer for current cursor position *Key Bindings: Normal: Ctrl-space; Emacs: Alt-Tab; Visual Studio: Ctrl-J; Eclipse (Experimental): Ctrl-space; OS X: Ctrl-space*

### **show-selection ()**

Turn on display of the current text selection

### **smart-tab ()**

Implement smart handling of tab key. The behavior varies by context as follows:

Unexpected indentation.

- In Non-Python code, always indents to the next indent stop
- On a non-blank line when cursor is at end or before a comment, insert tab
- On a where indent does not match the computed indent level, move to the matching indent level
- Otherwise decrease indent one level (thus a non-blank line toggles between matching position and one block higher)

**start-of-document ()**

Move cursor to start of document *Key Bindings: Normal: Ctrl-Home; VI/VIM: Ctrl-Home; Emacs: Ctrl-Home; Brief: Ctrl-Home; Visual Studio: Ctrl-Home; Eclipse (Experimental): Ctrl-Home; OS X: Command-Up*

**start-of-document-extend ()**

Move cursor to start of document, adjusting the selection range to new position *Key Bindings: Normal: Ctrl-Shift-Home; VI/VIM: Ctrl-Shift-Home; Emacs: Ctrl-Shift-Home; Brief: Ctrl-Shift-Home; Visual Studio: Ctrl-Shift-Home; Eclipse (Experimental): Ctrl-Shift-Home; OS X: Command-Shift-Up*

**stop-mark-command (deselect=True)**

Stop text marking for selection at current cursor position, leaving the selection set as is. Subsequent cursor move operations will deselect the range and set selection to cursor position. Deselect immediately when deselect is True. *Key Bindings: Emacs: Ctrl-G*

**swap-lines (previous=False)**

Swap the line at start of current selection with the line that follows it, or the preceding line if previous is True. *Key Bindings: Normal: Ctrl-Shift-L; Emacs: Ctrl-X Ctrl-T invokes swap-lines(previous=True); Eclipse (Experimental): Ctrl-Shift-L*

**tab-key ()**

Implement the tab key, the action of which is configurable by preference *Key Bindings: Normal: KP\_Tab; VI/VIM: KP\_Tab; Emacs: KP\_Tab; Brief: KP\_Tab; Visual Studio: KP\_Tab; Eclipse (Experimental): KP\_Tab; OS X: Tab*

**undo ()**

Undo last action *Key Bindings: Normal: Ctrl-Z; VI/VIM: u; Emacs: Ctrl-/; Brief: Alt-U; Visual Studio: Ctrl-Z; Eclipse (Experimental): Ctrl-Z; OS X: Command-Z*

**yank-line ()**

Yank contents of kill buffer created with kill-line into the edit buffer *Key Bindings: Emacs: Ctrl-Y*

## General Editor Commands

Editor commands that act on the current (most recently active) source editor, whether or not it currently has the keyboard focus.

**check-indent-consistency** ()

Check whether indents consistently use spaces or tabs throughout the file.

**comment-out-region** (style=None)

Comment out the selected region. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting. *Key Bindings: Normal: Ctrl-/; Emacs: Ctrl-C C; Visual Studio: Ctrl-K Ctrl-C; Eclipse (Experimental): Ctrl-/; OS X: Command-'*

**comment-out-toggle** (style=None)

Comment out the selected lines. This command is not available if they lines are already commented out. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**comment-toggle** (style=None)

Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. *Key Bindings: Normal: Ctrl-.; Emacs: Ctrl-C #; Visual Studio: Ctrl-K Ctrl-T; Eclipse (Experimental): Ctrl-.; OS X: Command-;*

**convert-indents-to-mixed** (indent\_size)

Convert all lines with leading spaces to mixed tabs and spaces.

**convert-indents-to-spaces-only** (indent\_size)

Convert all lines containing leading tabs to spaces only.

**convert-indents-to-tabs-only** ()

Convert all indentation to use tab characters only and no spaces

**evaluate-file-in-shell** (restart\_shell=None)

Run the contents of the editor within the Python Shell *Key Bindings: Normal: Ctrl-Alt-V; Eclipse (Experimental): Ctrl-Alt-V*

**evaluate-sel-in-debug-probe** (whole\_lines=None)

Evaluate the current selection from the editor within the Debug Probe tool. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead. *Key Bindings: Normal: Ctrl-Alt-D; Eclipse (Experimental): Ctrl-Alt-D*

**evaluate-sel-in-shell** (`restart_shell=False`, `whole_lines=None`)

Evaluate the current selection from the editor within the Python Shell tool, optionally restarting the shell first. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead. *Key Bindings: Normal: Ctrl-Alt-E; Emacs: Ctrl-C |; Eclipse (Experimental): Ctrl-Alt-E*

**execute-kbd-macro** (`register='a'`, `repeat=<numeric modifier; default=1>`)

Execute most recently recorded keyboard macro. If `register` is `None` then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default. *Key Bindings: Normal: Ctrl-M; VI/VIM: @ invokes execute-kbd-macro(register=None); Emacs: Ctrl-X E; Brief: F8; Visual Studio: Ctrl-M; Eclipse (Experimental): Ctrl-M; OS X: Command-M*

**fill-paragraph** ()

Attempt to auto-justify the paragraph around the current start of selection *Key Bindings: Normal: Ctrl-J; VI/VIM: g q q; Emacs: Ctrl-J; Visual Studio: Ctrl-K Ctrl-F; Eclipse (Experimental): Ctrl-J; OS X: Command-J*

**find-symbol** ()

Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name *Key Bindings: Normal: Ctrl-Shift-T; VI/VIM: Ctrl-Shift-T; Emacs: Ctrl-X Ctrl-G; Visual Studio: Ctrl-Shift-T; Eclipse (Experimental): Ctrl-Shift-T; OS X: Command-Shift-T*

**fold-collapse-all** ()

Collapse all fold points in the current file *Key Bindings: Normal: Alt-Home; VI/VIM: Alt-Home; Emacs: Alt-Home; Brief: Alt-Home; Visual Studio: Alt-Home; Eclipse (Experimental): Alt-Home; OS X: Command-Ctrl-KP\_Subtract*

**fold-collapse-all-clicked** ()

Collapse the clicked fold point completely

**fold-collapse-all-current** ()

Collapse the current fold point completely *Key Bindings: Normal: Alt-Page\_Up;*

*VI/VIM: Alt-Page\_Up; Emacs: Alt-Page\_Up; Brief: Alt-Page\_Up; Visual Studio: Alt-Page\_Up; Eclipse (Experimental): Alt-Page\_Up; OS X: Command-KP\_Subtract*

### **fold-collapse-current ()**

Collapse the current fold point *Key Bindings: VI/VIM: z c; Eclipse (Experimental): Ctrl-KP\_Subtract*

### **fold-collapse-more-clicked ()**

Collapse the clicked fold point one more level

### **fold-collapse-more-current ()**

Collapse the current fold point one more level *Key Bindings: Normal: Alt-Up; VI/VIM: Alt-Up; Emacs: Alt-Up; Brief: Alt-Up; Visual Studio: Alt-Up; Eclipse (Experimental): Alt-Up; OS X: Command-Shift-KP\_Subtract*

### **fold-expand-all ()**

Expand all fold points in the current file *Key Bindings: Normal: Alt-End; VI/VIM: Alt-End; Emacs: Alt-End; Brief: Alt-End; Visual Studio: Alt-End; Eclipse (Experimental): Alt-End; OS X: Command-Ctrl-KP\_Multiply*

### **fold-expand-all-clicked ()**

Expand the clicked fold point completely

### **fold-expand-all-current ()**

Expand the current fold point completely *Key Bindings: Normal: Alt-Page\_Down; VI/VIM: Alt-Page\_Down; Emacs: Alt-Page\_Down; Brief: Alt-Page\_Down; Visual Studio: Alt-Page\_Down; Eclipse (Experimental): Alt-Page\_Down; OS X: Command-KP\_Multiply*

### **fold-expand-current ()**

Expand the current fold point *Key Bindings: VI/VIM: z o; Eclipse (Experimental): Ctrl-KP\_Add*

### **fold-expand-more-clicked ()**

Expand the clicked fold point one more level

### **fold-expand-more-current ()**

Expand the current fold point one more level *Key Bindings: Normal: Alt-Down; VI/VIM: Alt-Down; Emacs: Alt-Down; Brief: Alt-Down; Visual Studio: Alt-Down; Eclipse (Experimental): Alt-Down; OS X: Command-KP\_Add*

**fold-toggle ()**

Toggle the current fold point *Key Bindings: Normal: Alt-/; VI/VIM: Alt-/; Emacs: Alt-/; Brief: Alt-/; Visual Studio: Alt-/; Eclipse (Experimental): Alt-/; OS X: Command-KP\_Divide*

**fold-toggle-clicked ()**

Toggle the clicked fold point

**force-indent-style-to-match-file ()**

Force the indent style of the editor to match the indent style found in the majority of the file

**force-indent-style-to-mixed ()**

Force the indent style of the editor to mixed use of tabs and spaces, regardless of the file contents

**force-indent-style-to-spaces-only ()**

Force the indent style of the editor to use spaces only, regardless of file contents

**force-indent-style-to-tabs-only ()**

Force the indent style of the editor to use tabs only, regardless of file contents

**goto-column** (column=<numeric modifier; default=0>)

Move cursor to given column *Key Bindings: VI/VIM: |*

**goto-line** (lineno=<numeric modifier>)

Position cursor at start of given line number *Key Bindings: Normal: Ctrl-L; Emacs: Alt-G; Brief: Alt-G; Visual Studio: Ctrl-G; Eclipse (Experimental): Ctrl-L; OS X: Command-L*

**goto-line-select** (lineno=<numeric modifier>)

Scroll to and select the given line number

**goto-nth-line** (lineno=<numeric modifier; default=1>, cursor='start')

Position cursor at start of given line number (1=first, -1 = last). This differs from goto-line in that it never prompts for a line number but instead uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fmb' for first non-blank character. *Key Bindings: VI/VIM: g g invokes goto-nth-line(cursor="fmb")*

**goto-nth-line-default-end** (lineno=<numeric modifier; default=0>, cursor='start')

Same as goto\_nth\_line but defaults to end of file if no lineno is given *Key Bindings: VI/VIM: Shift-G invokes goto-nth-line-default-end(cursor="fmb")*

**goto-percent-line** (percent=<numeric modifier; default=0>, cursor='start')

Position cursor at start of line at given percent in file. This uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fmb' for first non-blank character, or in VI mode it will do brace matching operation to reflect how VI overrides this command. *Key Bindings: VI/VIM: % invokes goto-percent-line(cursor="fmb")*

**hide-all-whitespace** ()

Turn off all special marks for displaying white space and end-of-line

**hide-eol** ()

Turn off special marks for displaying end-of-line chars

**hide-indent-guides** ()

Turn off special marks for displaying indent level

**hide-whitespace** ()

Turn off special marks for displaying white space

**indent-lines** (lines=None, levels=<numeric modifier; default=1>)

Indent selected number of lines from cursor position. Set lines to None to indent all the lines in current selection. Set levels to indent more than one level at a time. *Key Bindings: VI/VIM: >; Eclipse (Experimental): Ctrl-| invokes indent-lines(lines=1)*

**indent-next-move** (num=<numeric modifier; default=1>)

Indent lines spanned by next cursor move *Key Bindings: VI/VIM: >*

**indent-region** (sel=None)

Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent. *Key Bindings: Normal: Ctrl->; VI/VIM: Ctrl-T; Emacs: Ctrl-C >; Visual Studio: Ctrl->; Eclipse (Experimental): Ctrl->; OS X: Command-]*

**indent-to-match-next-move** (num=<numeric modifier; default=1>)



Indent lines spanned by next cursor move to match, based on the preceding line *Key Bindings: VI/VIM: =*

### **insert-command** (cmd)

Insert the output for the given command at current cursor position. Some special characters in the command line (if not escaped with `\`) will be replaced as follows:

```
% -- Current file's full path name
# -- Previous file's full path name
```

### **insert-file** (filename)

Insert a file at current cursor position, prompting user for file selection *Key Bindings: Emacs: Ctrl-X I; Brief: Alt-R*

### **join-lines** (delim=' ', num=<numeric modifier; default=2>)

Join together specified number of lines after current line (replace newlines with the given delimiter (single space by default) *Key Bindings: VI/VIM: Shift-J*

### **join-selection** (delim=' ')

Join together all lines in given selection (replace newlines with the given delimiter (single space by default) *Key Bindings: VI/VIM: Shift-J*

### **kill-buffer** ()

Close the current text file *Key Bindings: Emacs: Ctrl-X K; Brief: Ctrl--*

### **outdent-lines** (lines=None, levels=<numeric modifier; default=1>)

Outdent selected number of lines from cursor position. Set lines to None to indent all the lines in current selection. Set levels to outdent more than one level at a time. *Key Bindings: VI/VIM: <*

### **outdent-next-move** (num=<numeric modifier; default=1>)

Outdent lines spanned by next cursor move *Key Bindings: VI/VIM: <*

### **outdent-region** (sel=None)

Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent. *Key Bindings: Normal: Ctrl-<; VI/VIM: Ctrl-D; Emacs: Ctrl-C <; Visual Studio: Ctrl-<; Eclipse (Experimental): Ctrl-<; OS X: Command-[*

**page-setup** ()

Show printing page setup dialog

**print-view** ()

Print active editor document *Key Bindings: Normal: Ctrl-P; Visual Studio: Ctrl-P; Eclipse (Experimental): Ctrl-P; OS X: Command-P*

**query-replace** (search\_string, replace\_string)

Initiate incremental mini-search query/replace from the cursor position. *Key Bindings: Normal: Alt-comma; Emacs: Alt-%; Visual Studio: Alt-comma; Eclipse (Experimental): Alt-comma; OS X: Ctrl-R*

**query-replace-regex** (search\_string, replace\_string)

Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression. *Key Bindings: Normal: Ctrl-Alt-Comma; Emacs: Ctrl-Alt-%; Visual Studio: Ctrl-Alt-Comma; Eclipse (Experimental): Ctrl-Alt-Comma*

**range-replace** (search\_string, replace\_string, confirm, range\_limit, match\_limit, regex)

Initiate incremental mini-search query/replace within the given selection. This is similar to query\_replace but allows some additional options:

```
confirm -- True to confirm each replace
range_limit -- None to replace between current selec-
tion start and end of document,
    1 to limit operation to current selection or to cur-
rent line if selection is empty,
    (start, end) to limit operation to within given selec-
tion range, or "first|last"
    to limit operating withing given range of lines (1=first).
match_limit -- None to replace any num-
ber of matches, or limit of number of replaces.
    When set to "1" plus a number, limits to that num-
ber of matches per line,
    rather than as a whole.
regex -- Treat search string as a regular expression
```

**repeat-replace** (repeat=<numeric modifier; default=1>)

Repeat the last query replace or range replace operation on the current line. The first match is replaced without confirmation. *Key Bindings: VI/VIM: ℘*

**replace-char** (line\_mode='multiline', num=<numeric modifier; default=1>)

Replace num characters with given character. Set line\_mode to multiline to allow replacing across lines, extend to replace on current line and then extend the line length, and restrict to replace only if enough characters exist on current line after cursor position. *Key Bindings: VI/VIM: r invokes replace-char(line\_mode="restrict")*

**replace-string** (search\_string, replace\_string)

Replace all occurrences of a string from the cursor position to end of file. *Key Bindings: Normal: Alt-.; Emacs: Alt-@; Visual Studio: Alt-.; Eclipse (Experimental): Alt-.*

**replace-string-regex** (search\_string, replace\_string)

Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression. *Key Bindings: Normal: Ctrl-Alt-.; Emacs: Ctrl-Alt-@; Visual Studio: Ctrl-Alt-.; Eclipse (Experimental): Ctrl-Alt-.*

**save-buffer** ()

Save the current text file to disk

**set-readonly** ()

Set editor to be readonly. This cannot be done if the editor contains any unsaved edits.

**set-visit-history-anchor** ()

Set anchor in the visit history to go back to

**set-writable** ()

Set editor to be writable. This can be used to override the read-only state used initially for editors displaying files that are read-only on disk.

**show-all-whitespace** ()

Turn on all special marks for displaying white space and end-of-line

**show-eol** ()

Turn on special marks for displaying end-of-line chars

**show-indent-guides** ()

Turn on special marks for displaying indent level

**show-indent-manager** ()

Display the indentation manager for this editor file

**show-whitespace ()**

Turn on special marks for displaying white space

**start-kbd-macro** (register='a')

Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default. *Key Bindings: Normal: Ctrl-(; VI/VIM: q invokes start-kbd-macro(register=None); Emacs: Ctrl-X (; Brief: F7; Visual Studio: Ctrl-(; Eclipse (Experimental): Ctrl-(; OS X: Command-(*

**stop-kbd-macro ()**

Stop definition of a keyboard macro *Key Bindings: Normal: Ctrl-); VI/VIM: q; Emacs: Ctrl-X ); Brief: Shift-F7; Visual Studio: Ctrl-); Eclipse (Experimental): Ctrl-); OS X: Command-)*

**toggle-auto-editing ()**

Toggle the global auto-editing switch. When enabled, the editor performs the auto-edits that have been selected in the Editor > Auto-Editing preferences group.

**toggle-line-wrapping ()**

Toggles line wrapping preference for all editors

**toggle-overtyping ()**

Toggle status of overtyping mode *Key Bindings: Normal: Insert; VI/VIM: Insert; Emacs: Insert; Brief: Insert; Visual Studio: Insert; Eclipse (Experimental): Insert*

**uncomment-out-region** (one\_level=True)

Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting. *Key Bindings: Normal: Ctrl-?; Emacs: Ctrl-C U; Visual Studio: Ctrl-K Ctrl-U; Eclipse (Experimental): Ctrl-?; OS X: Command-"*

**uncomment-out-toggle** (style=None)

Remove commenting from the selected lines, if any. This command is not available if the lines are not commented out.

**use-lexer-ada ()**

Force syntax highlighting Ada source

**use-lexer-apache-conf ()**

Force syntax highlighting for Apache configuration file format

**use-lexer-asm** ()

Force syntax highlighting for Masm assembly language

**use-lexer-ave** ()

Force syntax highlighting for Avenue GIS language

**use-lexer-baan** ()

Force syntax highlighting for Baan

**use-lexer-bash** ()

Force syntax highlighting for bash scripts

**use-lexer-bullant** ()

Force syntax highlighting for Bullant

**use-lexer-by-doctype** ()

Use syntax highlighting appropriate to the file type

**use-lexer-cpp** ()

Force syntax highlighting for C/C++ source *Key Bindings: Normal: Ctrl-7 C; Emacs: Ctrl-X L C; Visual Studio: Ctrl-7 C; Eclipse (Experimental): Ctrl-7 C; OS X: Command-7 C*

**use-lexer-css2** ()

Force syntax highlighting for CSS2

**use-lexer-cython** ()

Force syntax highlighting for Cython source

**use-lexer-diff** ()

Force syntax highlighting for diff/cdiff files

**use-lexer-django** ()

Force syntax highlighting for Django template file

**use-lexer-dos-batch** ()

Force syntax highlighting for DOS batch files

**use-lexer-eiffel** ()

Force syntax highlighting for Eiffel source

**use-lexer-errlist** ()

Force syntax highlighting for error list format

**use-lexer-escript** ()

Force syntax highlighting for EScript

**use-lexer-fortran** ()

Force syntax highlighting for Fortran

**use-lexer-html** ()

Force syntax highlighting for HTML *Key Bindings: Normal: Ctrl-7 H; Emacs: Ctrl-X L H; Visual Studio: Ctrl-7 H; Eclipse (Experimental): Ctrl-7 H; OS X: Command-7 H*

**use-lexer-idl** ()

Force syntax highlighting for XP IDL

**use-lexer-java** ()

Force syntax highlighting for Java source

**use-lexer-javascript** ()

Force syntax highlighting for Javascript

**use-lexer-latex** ()

Force syntax highlighting for LaTeX

**use-lexer-lisp** ()

Force syntax highlighting for Lisp source

**use-lexer-lout** ()

Force syntax highlighting for LOUT typesetting language

**use-lexer-lua** ()

Force syntax highlighting for Lua

**use-lexer-makefile ()**

Force syntax highlighting for make files *Key Bindings: Normal: Ctrl-7 M; Emacs: Ctrl-X L M; Visual Studio: Ctrl-7 M; Eclipse (Experimental): Ctrl-7 M; OS X: Command-7 M*

**use-lexer-mako ()**

Force syntax highlighting for Mako template file

**use-lexer-matlab ()**

Force syntax highlighting for Matlab

**use-lexer-mmixal ()**

Force syntax highlighting for MMIX assembly language

**use-lexer-msidl ()**

Force syntax highlighting for MS IDL

**use-lexer-nncrontab ()**

Force syntax highlighting for NNCrontab files

**use-lexer-none ()**

Use no syntax highlighting *Key Bindings: Normal: Ctrl-7 N; Emacs: Ctrl-X L N; Visual Studio: Ctrl-7 N; Eclipse (Experimental): Ctrl-7 N; OS X: Command-7 N*

**use-lexer-nsis ()**

Force syntax highlighting for NSIS

**use-lexer-pascal ()**

Force syntax highlighting for Pascal source

**use-lexer-perl ()**

Force syntax highlighting for Perl source

**use-lexer-php ()**

Force syntax highlighting for PHP source

**use-lexer-plsql ()**

Force syntax highlighting for PL/SQL files

**use-lexer-pov ()**

Force syntax highlighting for POV ray tracer scene description language

**use-lexer-properties ()**

Force syntax highlighting for properties files

**use-lexer-ps ()**

Force syntax highlighting for Postscript

**use-lexer-python ()**

Force syntax highlighting for Python source *Key Bindings: Normal: Ctrl-7 P; Emacs: Ctrl-X L P; Visual Studio: Ctrl-7 P; Eclipse (Experimental): Ctrl-7 P; OS X: Command-7 P*

**use-lexer-qss ()**

Force syntax highlighting for QSS (Qt Style sheets)

**use-lexer-r ()**

Force syntax highlighting for R source file

**use-lexer-rc ()**

Force syntax highlighting for RC file format

**use-lexer-ruby ()**

Force syntax highlighting for Ruby source

**use-lexer-scriptol ()**

Force syntax highlighting for Scriptol

**use-lexer-sql ()**

Force syntax highlighting for SQL *Key Bindings: Normal: Ctrl-7 S; Emacs: Ctrl-X L S; Visual Studio: Ctrl-7 S; Eclipse (Experimental): Ctrl-7 S; OS X: Command-7 S*

**use-lexer-tcl ()**

Force syntax highlighting for TCL

**use-lexer-vb ()**

Force syntax highlighting for Visual Basic



**use-lexer-vxml ()**

Force syntax highlighting for VXML

**use-lexer-xcode ()**

Force syntax highlighting for XCode files

**use-lexer-xml ()**

Force syntax highlighting for XML files *Key Bindings: Normal: Ctrl-7 X; Visual Studio: Ctrl-7 X; Eclipse (Experimental): Ctrl-7 X; OS X: Command-7 X*

**use-lexer-yaml ()**

Force syntax highlighting for YAML

**zoom-in ()**

Zoom in, increasing the text display size temporarily by one font size *Key Binding: Ctrl-KP\_Add*

**zoom-out ()**

Zoom out, increasing the text display size temporarily by one font size *Key Binding: Ctrl-KP\_Subtract*

## Shell Or Editor Commands

Commands available when working either in the shell or editor

**goto-clicked-symbol-defn ()**

Goto the definition of the source symbol that was last clicked on *Key Bindings: Normal: Ctrl-Left\_Click; VI/VIM: Ctrl-Left\_Click; Emacs: Ctrl-Left\_Click; Brief: Ctrl-Left\_Click; Visual Studio: Ctrl-Left\_Click; Eclipse (Experimental): Ctrl-Left\_Click; OS X: Command-Left\_Click*

**goto-selected-symbol-defn (other\_split=False)**

Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True. *Key Binding: F4*

## 12.4. Search Manager Commands

### Toolbar Search Commands

Commands available when the tool bar search entry area has the keyboard focus.

#### **backward-char ()**

Move backward one character *Key Bindings: Normal: Left; VI/VIM: Left; Emacs: Left; Brief: Left; Visual Studio: Left; Eclipse (Experimental): Left; OS X: Option-Up*

#### **backward-char-extend ()**

Move backward one character, extending the selection *Key Binding: Shift-Left*

#### **backward-delete-char ()**

Delete character behind the cursor *Key Bindings: Normal: BackSpace; VI/VIM: BackSpace; Emacs: BackSpace; Brief: BackSpace; Visual Studio: BackSpace; Eclipse (Experimental): BackSpace; OS X: Backspace*

#### **backward-delete-word ()**

Delete word behind the cursor *Key Bindings: Normal: Ctrl-BackSpace; VI/VIM: Ctrl-BackSpace; Emacs: Ctrl-BackSpace; Brief: Ctrl-BackSpace; Visual Studio: Ctrl-BackSpace; Eclipse (Experimental): Ctrl-BackSpace; OS X: Ctrl-Option-Delete*

#### **backward-word ()**

Move backward one word *Key Bindings: Normal: Ctrl-Left; VI/VIM: Ctrl-Left; Emacs: Ctrl-Left; Brief: Ctrl-Left; Visual Studio: Ctrl-Left; Eclipse (Experimental): Ctrl-Left; OS X: Option-Left*

#### **backward-word-extend ()**

Move backward one word, extending the selection *Key Bindings: Normal: Ctrl-Shift-Left; VI/VIM: Ctrl-Shift-Left; Emacs: Ctrl-Shift-Left; Brief: Ctrl-Shift-Left; Visual Studio: Ctrl-Shift-Left; Eclipse (Experimental): Ctrl-Shift-Left; OS X: Option-Shift-Left*

#### **beginning-of-line ()**

Move to the beginning of the toolbar search entry *Key Bindings: VI/VIM: 0 invokes beginning-of-line(toggle=0); Emacs: Home; Brief: Shift-Home; OS X: Command-Left*

#### **beginning-of-line-extend ()**

Move to the beginning of the toolbar search entry, extending the selection *Key Bindings:* *Emacs: Shift-Home; OS X: Ctrl-Shift-Left*

### **copy ()**

Cut selection *Key Bindings: Normal: Ctrl-Insert; VI/VIM: Ctrl-Insert; Emacs: Ctrl-Insert; Brief: Ctrl-Insert; Visual Studio: Ctrl-Insert; Eclipse (Experimental): Ctrl-Insert; OS X: Command-C*

### **cut ()**

Cut selection *Key Bindings: Normal: Shift-Delete; VI/VIM: Shift-Delete; Emacs: Shift-Delete; Brief: Shift-Delete; Visual Studio: Shift-Delete; Eclipse (Experimental): Shift-Delete; OS X: Command-X*

### **end-of-line ()**

Move to the end of the toolbar search entry *Key Bindings: Normal: End; VI/VIM: End; Emacs: End; Brief: End; Visual Studio: End; Eclipse (Experimental): End; OS X: Ctrl-Right*

### **end-of-line-extend ()**

Move to the end of the toolbar search entry, extending the selection *Key Bindings: Normal: Shift-End; VI/VIM: Shift-End; Emacs: Shift-End; Brief: Shift-End; Visual Studio: Shift-End; Eclipse (Experimental): Shift-End; OS X: Ctrl-Shift-Right*

### **forward-char ()**

Move forward one character *Key Binding: Right*

### **forward-char-extend ()**

Move forward one character, extending the selection *Key Binding: Shift-Right*

### **forward-delete-char ()**

Delete character in front of the cursor *Key Binding: Delete*

### **forward-delete-word ()**

Delete word in front of the cursor *Key Bindings: Normal: Ctrl-Delete; VI/VIM: Ctrl-Delete; Emacs: Ctrl-Delete; Brief: Ctrl-Delete; Visual Studio: Ctrl-Delete; Eclipse (Experimental): Ctrl-Delete; OS X: Option-Delete*

### **forward-word ()**

Move forward one word *Key Bindings: Normal: Ctrl-Right; VI/VIM: Ctrl-Right; Emacs:*

*Ctrl-Right; Brief: Ctrl-Right; Visual Studio: Ctrl-Right; Eclipse (Experimental): Ctrl-Right; OS X: Option-Right*

### **forward-word-extend ()**

Move forward one word, extending the selection *Key Bindings: Normal: Ctrl-Shift-Right; VI/VIM: Ctrl-Shift-Right; Emacs: Ctrl-Shift-Right; Brief: Ctrl-Shift-Right; Visual Studio: Ctrl-Shift-Right; Eclipse (Experimental): Ctrl-Shift-Right; OS X: Option-Shift-Right*

### **paste ()**

Paste from clipboard *Key Bindings: Normal: Shift-Insert; VI/VIM: Shift-Insert; Emacs: Shift-Insert; Brief: Shift-Insert; Visual Studio: Shift-Insert; Eclipse (Experimental): Shift-Insert; OS X: Ctrl-y*

## **Search Manager Commands**

Globally available commands defined for the search manager. These commands are available regardless of whether a search manager is visible or has keyboard focus.

### **batch-replace** (look\_in=None, use\_selection=True)

Display search and replace in files tool. *Key Bindings: Normal: Ctrl-Shift-R; VI/VIM: Ctrl-Shift-G; Emacs: Ctrl-; Visual Studio: Ctrl-Shift-R; Eclipse (Experimental): Ctrl-Shift-R; OS X: Command-Shift-R*

### **batch-search** (look\_in=None, use\_selection=True, search\_text=None)

Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided *Key Bindings: Normal: Ctrl-Shift-F; VI/VIM: Ctrl-Shift-F; Emacs: Ctrl-; Visual Studio: Ctrl-Shift-F; Eclipse (Experimental): Ctrl-Shift-F; OS X: Command-Shift-F*

### **batch-search-backward** ()

Move to the previous found match in the Search in Files tool.

### **batch-search-forward** ()

Move to the next found match in the Search in Files tool.

### **batch-search-pause** ()

Pause the currently running batch search, if any

## **replace ()**

Bring up the search manager in replace mode. *Key Bindings: Normal: Ctrl-R; Emacs: Ctrl-0; Brief: F6; Visual Studio: Ctrl-R; Eclipse (Experimental): Ctrl-R; OS X: Command-R*

## **replace-again ()**

Replace current selection with the search manager.

## **replace-and-search ()**

Replace current selection and search again. *Key Bindings: Normal: Ctrl-I; Emacs: Alt-.; Brief: Shift-F6; Visual Studio: Ctrl-I; Eclipse (Experimental): Ctrl-I; OS X: Command-Ctrl-R*

## **search ()**

Bring up the search manager in search mode. *Key Bindings: Normal: Alt-F3; VI/VIM: Alt-F3; Emacs: Alt-F3; Brief: Alt-F3; Visual Studio: Alt-F3; Eclipse (Experimental): Alt-F3; OS X: Option-F3*

## **search-again** (search\_string="", direction=1)

Search again using the search manager's current settings.

## **search-backward** (search\_string=None)

Search again using the search manager's current settings in backward direction *Key Binding: Shift-F3*

## **search-forward** (search\_string="")

Search again using the search manager's current settings in forward direction *Key Binding: F3*

## **search-sel ()**

Search forward using current selection

## **search-sel-backward ()**

Search backward using current selection *Key Bindings: Normal: Ctrl-Shift-F3; VI/VIM: Ctrl-Shift-F3; Emacs: Ctrl-Shift-F3; Brief: Ctrl-Shift-F3; Visual Studio: Ctrl-Shift-F3; Eclipse (Experimental): Ctrl-Shift-F3; OS X: Command-Shift-F3*

## **search-sel-forward ()**

Search forward using current selection *Key Bindings: Normal: Ctrl-F3; VI/VIM: Ctrl-F3; Emacs: Ctrl-F3; Brief: Ctrl-F3; Visual Studio: Ctrl-F3; Eclipse (Experimental): Ctrl-F3; OS X: Command-F3*

## Search Manager Instance Commands

Commands for a particular search manager instance. These are only available when the search manager has they keyboard focus.

**clear** ()

Clear selected text

**copy** ()

Copy selected text *Key Bindings: Normal: Ctrl-Insert; VI/VIM: Ctrl-Insert; Emacs: Ctrl-Insert; Brief: Ctrl-Insert; Visual Studio: Ctrl-Insert; Eclipse (Experimental): Ctrl-Insert; OS X: Command-C*

**cut** ()

Cut selected text *Key Bindings: Normal: Shift-Delete; VI/VIM: Shift-Delete; Emacs: Shift-Delete; Brief: Shift-Delete; Visual Studio: Shift-Delete; Eclipse (Experimental): Shift-Delete; OS X: Command-X*

**forward-tab** ()

Place a forward tab at the current cursor position in search or replace string *Key Binding: Ctrl-T*

**paste** ()

Paste text from clipboard *Key Bindings: Normal: Shift-Insert; VI/VIM: Shift-Insert; Emacs: Shift-Insert; Brief: Shift-Insert; Visual Studio: Shift-Insert; Eclipse (Experimental): Shift-Insert; OS X: Ctrl-y*

## 12.5. Debugger Commands

### Debugger Commands

Commands that control the debugger and current debug process, if any.

**break-clear** ()

Clear the breakpoint on the current line *Key Binding: F9*

**break-clear-all** ()

Clear all breakpoints *Key Bindings: Normal: Ctrl-F9; VI/VIM: Ctrl-F9; Emacs: Ctrl-F9; Brief: Ctrl-F9; Visual Studio: Ctrl-F9; Eclipse (Experimental): Ctrl-F9; OS X: Command-F9*

**break-clear-clicked** ()

Clear the breakpoint at current click location

**break-disable** ()

Disable the breakpoint on current line *Key Binding: Shift-F9*

**break-disable-all** ()

Disable all breakpoints *Key Bindings: Normal: Ctrl-Shift-F9; VI/VIM: Ctrl-Shift-F9; Emacs: Ctrl-Shift-F9; Brief: Ctrl-Shift-F9; Visual Studio: Ctrl-Shift-F9; Eclipse (Experimental): Ctrl-Shift-F9*

**break-disable-clicked** ()

Disable the breakpoint at current click location

**break-edit-cond** ()

Edit condition for the breakpoint on current line

**break-edit-cond-clicked** ()

Edit condition for the breakpoint at the current mouse click location

**break-enable** ()

Enable the breakpoint on the current line *Key Binding: Shift-F9*

**break-enable-all** ()

Enable all breakpoints *Key Bindings: Normal: Ctrl-Shift-F9; VI/VIM: Ctrl-Shift-F9; Emacs: Ctrl-Shift-F9; Brief: Ctrl-Shift-F9; Visual Studio: Ctrl-Shift-F9; Eclipse (Experimental): Ctrl-Shift-F9*

**break-enable-clicked** ()

Enable the breakpoint at current click location

**break-enable-toggle** ()

Toggle whether breakpoint on current line is enabled or disabled

**break-ignore** ()

Ignore the breakpoint on current line for N iterations

**break-ignore-clicked** ()

Ignore the breakpoint at the current mouse click location for N iterations

**break-set** ()

Set a new regular breakpoint on current line *Key Binding: F9*

**break-set-clicked** ()

Set a new regular breakpoint at the current mouse click location

**break-set-cond** ()

Set a new conditional breakpoint on current line

**break-set-cond-clicked** ()

Set a new conditional breakpoint at the current mouse click location

**break-set-temp** ()

Set a new temporary breakpoint on current line

**break-set-temp-clicked** ()

Set a new temporary breakpoint at the current mouse click location

**break-toggle** ()

Toggle breakpoint at current line (creates new regular bp when one is created)

**clear-exception-ignores-list** ()

Clear list of exceptions being ignored during debugging

**clear-var-errors** ()

Clear stored variable errors so they get refetched

**collapse-tree-more** ()

Collapse whole selected variables display subtree one more level

**debug-attach** ()



Attach to an already-running debug process

**debug-continue** (show\_dialog=None)

Continue (or start) running, to next breakpoint *Key Binding: F5*

**debug-detach** ()

Detach from the debug process and let it run

**debug-file** (show\_dialog=None)

Start debugging the current file (rather than the main entry point) *Key Binding: Shift-F5*

**debug-kill** ()

Stop debugging *Key Bindings: Normal: Ctrl-F5; VI/VIM: Ctrl-F5; Emacs: Ctrl-F5; Brief: Ctrl-F5; Visual Studio: Ctrl-F5; Eclipse (Experimental): Ctrl-F5; OS X: Command-F5*

**debug-probe-clear** ()

Clear debug probe.

**debug-restart** ()

Stop and restart the current debug process

**debug-stop** ()

Pause free-running execution at current program counter *Key Bindings: Normal: Ctrl-Shift-F5; VI/VIM: Ctrl-Shift-F5; Emacs: Ctrl-Shift-F5; Brief: Ctrl-Shift-F5; Visual Studio: Ctrl-Shift-F5; Eclipse (Experimental): Ctrl-Shift-F5; OS X: Command-Shift-F5*

**debug-to-clicked** ()

Debug to the line at the current mouse click location

**exception-always-stop** ()

Always stop on exceptions, even if they are handled by the code

**exception-never-stop** ()

Never stop on exceptions, even if they are unhandled in the code

**exception-stop-when-printed** ()

Stop only on exceptions when they are about to be printed

**exception-unhandled-stop ()**

Stop only on exceptions that are not handled by the code

**expand-tree-more ()**

Expand whole selected variables display subtree deeper

**force-var-reload ()**

Force refetch of a value from server

**frame-down ()**

Move down the current debug stack *Key Binding: F12*

**frame-show ()**

Show the position (thread and stack frame) where the debugger originally stopped *Key Bindings: Normal: Shift-F11; VI/VIM: Shift-F11; Emacs: Shift-F11; Brief: Shift-F11; Visual Studio: Shift-F11; Eclipse (Experimental): Shift-F11*

**frame-up ()**

Move up the current debug stack *Key Binding: F11*

**hide-detail ()**

Show the textual value detail area

**internal-extra-debugger-logging-start ()**

Turn on additional logging for diagnosing problems with the debugger

**internal-extra-debugger-logging-stop ()**

Turn off additional logging for diagnosing problems with the debugger

**python-shell-clear ()**

Clear python shell.

**python-shell-kill ()**

Kill python shell process.

**python-shell-restart ()**

Restart python shell.

**run-build-command** ()

Execute the build command defined in the project, if any

**run-to-cursor** ()

Run to current cursor position *Key Bindings: Normal: Alt-F5; VI/VIM: Alt-F5; Emacs: Alt-F5; Brief: Alt-F5; Visual Studio: Alt-F5; Eclipse (Experimental): Alt-F5*

**shell-copy-with-prompts** (shell=None)

Copy text from shell, including all prompts

**shell-ctrl-down** ()

Not documented

**shell-ctrl-return** ()

Not documented

**shell-ctrl-up** ()

Not documented

**show-detail** ()

Show the textual value detail area

**step-into** (show\_dialog=None)

Step into current execution point, or start debugging at first line *Key Binding: F7*

**step-out** ()

Step out of the current function or method *Key Binding: F8*

**step-over** ()

Step over current execution point *Key Binding: F6*

**watch** (style='ref')

Watch selected variable using a direct object reference to track it

**watch-expression** (expr=None)

Add a new expression to the watch list

**watch-module-ref** ()

Watch selected value relative to a module looked up by name in sys.modules

**watch-parent-ref** ()

Watch selected variable using a reference to the value's parent and the key slot for the value

**watch-ref** ()

Watch selected variable using a direct object reference to track it

**watch-symbolic** ()

Watch selected value using the symbolic path to it

## Debugger Watch Commands

Commands for the debugger's Watch tool (Wing IDE Professional only). These are available only when the watch tool has key board focus.

**watch-clear-all** ()

Clear all entries from the watch list

**watch-clear-selected** ()

Clear selected entry from the watch list

## Call Stack View Commands

Commands available on a specific instance of the call stack tool

**callstack-copy-to-clipboard** ()

Copy the call stack to the clipboard, as text

**callstack-set-codeline-mode** (mode)

Set the code line display mode for this call stack

**callstack-show-docs** ()

Show documentation for the call stack manager

## Exceptions Commands

Commands available when the debugger's Exceptions tool has the keyboard focus.

**clear** ()

Clear the exception currently shown on the display

**copy** ()

Copy the exception traceback to the clipboard *Key Bindings: Normal: Ctrl-Insert; VI/VIM: Ctrl-Insert; Emacs: Ctrl-Insert; Brief: Ctrl-Insert; Visual Studio: Ctrl-Insert; Eclipse (Experimental): Ctrl-Insert; OS X: Command-C*



# Key Binding Reference

This chapter documents all the default key bindings found in the keyboard personalities provided by Wing, set by the **Personality** preference. Key bindings are listed alphabetically. In some cases commands of the same name are provided by different implementations that are selected according to keyboard focus.

When multiple commands are defined for a single key binding, the first available command in the list is invoked. In this way a single binding can, for example, show or hide a tool panel.

Additional key bindings can be added as described in **keyboard bindings**. All available commands are documented in the **Command Reference**.

## 13.1. Normal Personality

This section documents all the default key bindings for the **Normal** keyboard personality, set by the **Personality** preference.

**Alt-1: fold-python-methods**

**Alt-2: fold-python-classes**

**Alt-3: fold-python-classes-and-defs**

**Alt-Apostrophe: `enclose(start="'", end="')`** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-BackSpace: `backward-delete-word`** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Braceleft: `enclose(start="{", end="}")`** - Enclose the selection or the rest of

the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft:** `enclose(start="[", end="]")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Delete:** `backward-delete-word` - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down:** `fold-expand-more-current` - Expand the current fold point one more level

**Alt-End:** `fold-expand-all` - Expand all fold points in the current file

**Alt-F11:** `prev-points-of-use-match`

**Alt-F12:** `next-points-of-use-match`

**Alt-F3:** `search` - Bring up the search manager in search mode.

**Alt-F4:** `close-window` - Close the current window and all documents and panels in it

**Alt-F5:** `run-to-cursor` - Run to current cursor position

**Alt-F6:** `run-failed-tests`

**Alt-F7:** `run-last-tests`

**Alt-Home:** `fold-collapse-all` - Collapse all fold points in the current file

**Alt-Left:** `visit-history-previous` - Move back in history to previous visited editor position

**Alt-Less:** `enclose(start="<", end=">")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Page\_Down:** `fold-expand-all-current` - Expand the current fold point completely

**Alt-Page\_Up:** `fold-collapse-all-current` - Collapse the current fold point completely

**Alt-Parenleft:** `enclose(start="(", end=")")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl:** `enclose(start="\"", end="\"")` - Enclose the selection or the rest of the



current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Return: new-line** - Place a new line at the current cursor position

**Alt-Right: visit-history-next** - Move forward in history to next visited editor position

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**Alt-comma: query-replace** - Initiate incremental mini-search query/replace from the cursor position.

**Alt-period: replace-string** - Replace all occurrences of a string from the cursor position to end of file.

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Ctrl-0: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-7 C: use-lexer-cpp** - Force syntax highlighting for C/C++ source

**Ctrl-7 H: use-lexer-html** - Force syntax highlighting for HTML

**Ctrl-7 M: use-lexer-makefile** - Force syntax highlighting for make files

**Ctrl-7 N: use-lexer-none** - Use no syntax highlighting

**Ctrl-7 P: use-lexer-python** - Force syntax highlighting for Python source

**Ctrl-7 S: use-lexer-sql** - Force syntax highlighting for SQL

**Ctrl-7 X: use-lexer-xml** - Force syntax highlighting for XML files

**Ctrl-8: recent-document** - Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode.

**Ctrl-9: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set `toggle=True` to indent instead of one level higher if already at the matching position.

**Ctrl-A: select-all** - Select all text in the editor

**Ctrl-Alt-B: search-sel-backward** - Search backward using current selection

**Ctrl-Alt-Comma: query-replace-regex** - Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression.

**Ctrl-Alt-D: evaluate-sel-in-debug-probe** - Evaluate the current selection from the editor within the Debug Probe tool. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-E: evaluate-sel-in-shell** - Evaluate the current selection from the editor within the Python Shell tool, optionally restarting the shell first. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead.

**Ctrl-Alt-F: search-sel-forward** - Search forward using current selection

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-G: goto-bookmark** - Goto named bookmark

**Ctrl-Alt-K: show-bookmarks** - Show a list of all currently defined bookmarks

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-M: set-bookmark** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-T: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Ctrl-Alt-Up: goto-previous-bookmark(current\_file\_only=True)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-V: evaluate-file-in-shell** - Run the contents of the editor within the Python Shell

**Ctrl-Alt-period: replace-string-regex** - Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression.

**Ctrl-B: isearch-sel-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-C: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Comma: next-window** - Switch to the next window alphabetically by title

**Ctrl-D: toolbar-search-focus** - Move focus to toolbar search entry.

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-E: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F: search** - Bring up the search manager in search mode.

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-G: search-forward** - Search again using the search manager's current settings in forward direction

**Ctrl-H: replace** - Bring up the search manager in replace mode.

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-I: replace-and-search** - Replace current selection and search again.

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-J: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Ctrl-K: open-from-keyboard** - Open a file from disk using keyboard-driven selection of the file

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-L: goto-line** - Position cursor at start of given line number

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-M: execute-kbd-macro** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Ctrl-N: new-file** - Create a new file

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-O: open-gui** - Open a file from disk, prompting with file selection dialog if necessary

**Ctrl-P: print-view** - Print active editor document

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Period: comment-toggle** - Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Q: quit** - Quit the application.

**Ctrl-Quoteleft: begin-visited-document-cycle(move\_back=True, back\_key="Ctrl-Quoteleft", forward\_key="Ctrl-AsciiTilde")** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-R: replace** - Bring up the search manager in replace mode.

**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Com-*

*mands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-S: save** - Save active document. Also close it if close is True.

**Ctrl-Shift-B: isearch-sel-backward** - Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**Ctrl-Shift-C: delete-line** - Delete the current line or lines when the selection spans multiple lines or given repeat is > 1

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: move-line-down** - Move the current line or lines up down line, optionally indenting to match the new position

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-F: batch-search** - Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-Shift-F3: search-sel-backward** - Search backward using current selection

**Ctrl-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: debug-all-tests**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-G: search-backward** - Search again using the search manager's current settings in backward direction

**Ctrl-Shift-H: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor



is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-L: swap-lines** - Swap the line at start of current selection with the line that follows it, or the preceding line if previous is True.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If skip\_if\_unique is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-R: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-S: save-as** - Save active document to a new file

**Ctrl-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-U: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Shift-Up: move-line-up** - Move the current line or lines up one line, optionally indenting to match the new position

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-V: duplicate-line** - Duplicate the current line or lines. Places the duplicate on the line following the selection if pos is 'below' or before the selection if it is 'above'.

**Ctrl-Shift-Y: duplicate-line-above** - Duplicate the current line or lines above the selection.

**Ctrl-Slash: comment-out-region** - Comment out the selected region. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting.

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-U: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position

on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-V: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Ctrl-W: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-X: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Y: redo** - Redo last action

**Ctrl-Z: undo** - Undo last action

**Ctrl-]: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-greater: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or “never-select” to unselect after indent, “always-select” to always select after indent, or “retain-select” to retain current selection after indent.

**Ctrl-less: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or “never-select” to unselect after indent, “always-select” to always select after indent, or “retain-select” to retain current selection after indent.

**Ctrl-parenleft: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**Ctrl-parenright: stop-kbd-macro** - Stop definition of a keyboard macro

**Ctrl-question: uncomment-out-region** - Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting.

**Ctrl-space: show-autocompleter** - Show the auto-completer for current cursor position

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Com-*

*mands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**F1**: Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F11: frame-up** - Move up the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2**: Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is True.

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F6: step-over** - Step over current execution point

**F7: step-into** - Step into current execution point, or start debugging at first line

**F8: step-out** - Step out of the current function or method

**F9**: Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: toggle-overtyping** - Toggle status of overtyping mode

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyping** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning charac-

ter within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N: diff-next**

**Shift-Alt-P: diff-previous**

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F2:** Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line

- **break-disable** - Disable the breakpoint on current line

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position



**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Up: previous-line** - Move to previous screen line, optionally repositioning character

within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

## 13.2. Emacs Personality

This section documents all the default key bindings for the **Emacs** keyboard personality, set by the **Personality** preference.

**Alt-!:** **execute-process** - Execute the given command line in the OS Commands tool using default run directory and environment as defined in project properties, or the values set in an existing command with the same command line in the OS Commands tool.

**Alt-0:** **initiate-repeat-0** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-1:** **fold-python-methods**

**Alt-1:** **initiate-repeat-1** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-2:** **fold-python-classes**

**Alt-2:** **initiate-repeat-2** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-3:** **fold-python-classes-and-defs**

**Alt-3:** **initiate-repeat-3** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-4:** **initiate-repeat-4** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-5:** **initiate-repeat-5** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-6:** **initiate-repeat-6** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-7:** **initiate-repeat-7** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-8: initiate-repeat-8** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-9: initiate-repeat-9** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Alt-@: replace-string** - Replace all occurrences of a string from the cursor position to end of file.

**Alt-Apostrophe: enclose(start="'", end="'")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-B: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Alt-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Backslash: fold-toggle** - Toggle the current fold point

**Alt-Braceleft: enclose(start="{", end="}")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft: enclose(start="[", end="]")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-C: title-case**

**Alt-D: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Alt-Delete: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down: fold-expand-more-current** - Expand the current fold point one more level

**Alt-End: fold-expand-all** - Expand all fold points in the current file

**Alt-F: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Alt-F11: prev-points-of-use-match**

**Alt-F12: next-points-of-use-match**

**Alt-F3: search** - Bring up the search manager in search mode.

**Alt-F4: close-window** - Close the current window and all documents and panels in it

**Alt-F5: run-to-cursor** - Run to current cursor position

**Alt-F6: run-failed-tests**

**Alt-F7: run-last-tests**

**Alt-G: goto-line** - Position cursor at start of given line number

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-KP\_Enter: new-line** - Place a new line at the current cursor position

**Alt-L: goto-line** - Position cursor at start of given line number

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Less: enclose(start=“<”, end=“>”)** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Parenleft: enclose(start=“(", end=“”)”)** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Period: replace-and-search** - Replace current selection and search again.

**Alt-Q: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Alt-Quotedbl: `enclose(start="“, end=”)`** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Return: `new-line`** - Place a new line at the current cursor position

**Alt-Right: `visit-history-next`** - Move forward in history to next visited editor position

**Alt-Slash: `fold-toggle`** - Toggle the current fold point

**Alt-Slash: `show-autocompleter`** - Show the auto-completer for current cursor position

**Alt-Tab: `show-autocompleter`** - Show the auto-completer for current cursor position

**Alt-Up: `fold-collapse-more-current`** - Collapse the current fold point one more level

**Alt-V: `backward-page`** - Move cursor backward one page

**Alt-W: `copy`** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Alt-X: `command-by-name`** - Execute given command by name, collecting any args as needed

**Alt-g: `goto-line`** - Position cursor at start of given line number

**Alt-greater: `end-of-document`** - Move cursor to end of document

**Alt-less: `start-of-document`** - Move cursor to start of document

**Alt-percent: `query-replace`** - Initiate incremental mini-search query/replace from the cursor position.

**Alt-percent: `query-replace`** - Initiate incremental mini-search query/replace from the cursor position.

**Alt-{: `previous-blank-line(threshold=1)`** - Move to the previous blank line in the file, if any. If `threshold>0` then a line is considered blank if it contains less than that many characters after leading and trailing whitespace are removed.

**Alt-}: `next-blank-line(threshold=1)`** - Move to the next blank line in the file, if any. If `threshold>0` then a line is considered blank if it contains less than that many characters after leading and trailing whitespace are removed.

**BackSpace: `backward-delete-char`** - Action varies according to focus: *Active Editor*

*Commands*: Delete one character behind the cursor, or the current selection if not empty.  
*; Toolbar Search Commands*: Delete character behind the cursor

**Ctrl-0: replace** - Bring up the search manager in replace mode.

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-9: search** - Bring up the search manager in search mode.

**Ctrl=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set toggle=True to indent instead of one level higher if already at the matching position.

**Ctrl-@: set-mark-command** - Set start of text marking for selection at current cursor position. Subsequently, all cursor move operations will automatically extend the text selection until stop-mark-command is issued. Unit defines what is selected: can be one of char, line, or block (rectangle).

**Ctrl-A: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Ctrl-Alt-@: replace-string-regex** - Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression.

**Ctrl-Alt-B: search-sel-backward** - Search backward using current selection

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when current\_file\_only is True.

**Ctrl-Alt-F: search-sel-forward** - Search forward using current selection

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-R: isearch-backward-regex** - Action varies according to focus: *Active Editor Commands*: Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-S: isearch-forward-regex** - Action varies according to focus: *Active Editor Commands*: Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-Alt-Up: goto-previous-bookmark(current\_file\_only=True)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-percent: query-replace-regex** - Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression.

**Ctrl-B: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-C Bar: evaluate-sel-in-shell** - Evaluate the current selection from the editor within the Python Shell tool, optionally restarting the shell first. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead.

**Ctrl-C C: comment-out-region** - Comment out the selected region. The style of commenting can be controlled with the style argument: 'indented' uses the default com-

ment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting.

**Ctrl-C Ctrl-C: debug-continue** - Continue (or start) running, to next breakpoint

**Ctrl-C Ctrl-K: debug-kill** - Stop debugging

**Ctrl-C Ctrl-S: debug-stop** - Pause free-running execution at current program counter

**Ctrl-C M: isearch-sel**

**Ctrl-C R: isearch-sel-backward** - Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Ctrl-C S: isearch-sel-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Ctrl-C U: uncomment-out-region** - Uncomment out the selected region if commented out. If `one_level` is True then each call removes only one level of commenting.

**Ctrl-C greater: indent-region** - Indent the selected region one level of indentation. Set `sel` to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-C less: outdent-region** - Outdent the selected region one level of indentation. Set `sel` to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-C numbersign: comment-toggle** - Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**Ctrl-D: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor*



*Commands:* Delete one word in front of the cursor ; *Toolbar Search Commands:* Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-E: end-of-line** - Action varies according to focus: *Active Editor Commands:* Move to end of current line; *Toolbar Search Commands:* Move to the end of the toolbar search entry

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F: forward-char** - Action varies according to focus: *Active Editor Commands:* Move cursor forward one character ; *Toolbar Search Commands:* Move forward one character

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-G: stop-mark-command** - Stop text marking for selection at current cursor position, leaving the selection set as is. Subsequent cursor move operations will deselect the range and set selection to cursor position. Deselect immediately when deselect is True.

**Ctrl-H: backward-delete-char** - Action varies according to focus: *Active Editor Commands:* Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands:* Delete character behind the cursor

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands:* Copy selected text ; *Document Viewer Commands:* Copy any selected text. ; *Exceptions Commands:* Copy the exception traceback to the clipboard ; *Search Manager Instance Commands:* Copy selected text ; *Toolbar Search Commands:* Cut selection

**Ctrl-J: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Ctrl-K: kill-line** - Kill rest of line from cursor to end of line, and place it into the clipboard with any other contiguously removed lines. End-of-line is removed only if there is nothing between the cursor and the end of the line.

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Enter: new-line** - Place a new line at the current cursor position

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-L: center-cursor** - Scroll so cursor is centered on display

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-M: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-N: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-O: next-window** - Switch to the next window alphabetically by title

**Ctrl-O: open-line** - Open the current line by inserting a newline after the caret

**Ctrl-P: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Period: redo** - Redo last action

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Quoteleft:** `begin-visited-document-cycle(move_back=True, back_key="Ctrl-Quoteleft", forward_key="Ctrl-AsciiTilde")` - Start moving

between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-R: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-Right: forward-word(gravity=“end”)** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-S: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same’ to leave in same horizontal position, ’start’ at start, ’end’ at end, or ’fmb’ for first non-blank char.

**Ctrl-Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-F3: search-sel-backward** - Search backward using current selection

**Ctrl-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: debug-all-tests**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which

characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same’ to leave in same horizontal position, ‘start’ at start, ‘end’ at end, or ‘fmb’ for first non-blank char.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be “start” or “end” to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same’ to leave in same horizontal position, ‘start’ at start, ‘end’ at end, or ‘fmb’ for first non-blank char.

**Ctrl-Slash: undo** - Undo last action

**Ctrl-Space: set-mark-command** - Set start of text marking for selection at current

cursor position. Subsequently, all cursor move operations will automatically extend the text selection until stop-mark-command is issued. Unit defines what is selected: can be one of char, line, or block (rectangle).

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-U: initiate-repeat** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-V: forward-page** - Move cursor forward one page

**Ctrl-W: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-X 1: unsplit** - Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  -- Change to recent buffer before unsplitting
recent-or-close -- Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future.

**Ctrl-X 2: split-vertically** - Split current view vertically. Create new editor in new view when new==1.

**Ctrl-X 3: split-horizontally** - Split current view horizontally.

**Ctrl-X 4 A: add-change-log-entry**

**Ctrl-X 5 0: close-window** - Close the current window and all documents and panels in it

**Ctrl-X 5 2: new-document-window** - Create a new document window with same documents and panels as in the current document window (if any; otherwise empty with default panels)

**Ctrl-X 5 3: new-document-window** - Create a new document window with same documents and panels as in the current document window (if any; otherwise empty with default panels)

**Ctrl-X 5 O: next-window** - Switch to the next window alphabetically by title

**Ctrl-X B: switch-document** - Switches to named document. Name may either be the complete name or the last path component of a path name.

**Ctrl-X Bracketleft: start-of-document** - Move cursor to start of document

**Ctrl-X Bracketright: end-of-document** - Move cursor to end of document

**Ctrl-X Ctrl-C: quit** - Quit the application.

**Ctrl-X Ctrl-F: open-from-keyboard** - Open a file from disk using keyboard-driven selection of the file

**Ctrl-X Ctrl-G: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-X Ctrl-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If `skip_if_unique` is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Ctrl-X Ctrl-S: save** - Save active document. Also close it if `close` is True.

**Ctrl-X Ctrl-T: swap-lines(previous=True)** - Swap the line at start of current selection with the line that follows it, or the preceding line if `previous` is True.

**Ctrl-X Ctrl-W: write-file** - Write current file to a new location, optionally omitting all but the lines in the given range. The editor is changed to point to the new location when `follow` is True. Note that the editor contents will be truncated to the given start/end lines when `follow` is True.

**Ctrl-X Ctrl-X: exchange-point-and-mark** - When currently marking text, this exchanges the current position and mark ends of the current selection

**Ctrl-X D: recent-document** - Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode.

**Ctrl-X E: execute-kbd-macro** - Execute most recently recorded keyboard macro. If



register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Ctrl-X I: insert-file** - Insert a file at current cursor position, prompting user for file selection

**Ctrl-X K: kill-buffer** - Close the current text file

**Ctrl-X L C: use-lexer-cpp** - Force syntax highlighting for C/C++ source

**Ctrl-X L H: use-lexer-html** - Force syntax highlighting for HTML

**Ctrl-X L M: use-lexer-makefile** - Force syntax highlighting for make files

**Ctrl-X L N: use-lexer-none** - Use no syntax highlighting

**Ctrl-X L P: use-lexer-python** - Force syntax highlighting for Python source

**Ctrl-X L S: use-lexer-sql** - Force syntax highlighting for SQL

**Ctrl-X L X: use-lexer-Xml**

**Ctrl-X N: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-X O: move-editor-focus** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Ctrl-X P: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-X R B: goto-bookmark** - Goto named bookmark

**Ctrl-X R M: set-bookmark** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Ctrl-X R Return: show-bookmarks** - Show a list of all currently defined bookmarks

**Ctrl-X R T: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Ctrl-X Space:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Ctrl-X U: undo** - Undo last action

**Ctrl-X parenleft: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**Ctrl-X parenright: stop-kbd-macro** - Stop definition of a keyboard macro

**Ctrl-Y:** Multiple commands (first available is executed):

- **yank-line** - Yank contents of kill buffer created with kill-line into the edit buffer
- **paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Ctrl-parenleft: batch-search** - Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-parenright: batch-replace** - Display search and replace in files tool.

**Ctrl-underscore: undo** - Undo last action

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**Esc X: command-by-name** - Execute given command by name, collecting any args as needed

**F1:** Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F11: frame-up** - Move up the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is True.

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F6: step-over** - Step over current execution point

**F7: step-into** - Step into current execution point, or start debugging at first line

**F8: step-out** - Step out of the current function or method

**F9:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When `toggle` is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If `toggle` is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: toggle-overtyping** - Toggle status of overtyping mode

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N: diff-next**

**Shift-Alt-P: diff-previous**

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F2:** Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-Home: beginning-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line, adjusting the selection range to the new position. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry, extending the selection

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position.

If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Enter: new-line** - Place a new line at the current cursor position

**Shift-KP\_Home: beginning-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line, adjusting the selection range to the new position. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry, extending the selection

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference



**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

### 13.3. VI/VIM Personality

This section documents all the default key bindings for the **VI/VIM** keyboard personality, set by the **Personality** preference.

**Alt-1: fold-python-methods**

**Alt-2: fold-python-classes**

**Alt-3: fold-python-classes-and-defs**

**Alt-Apostrophe: enclose(start="'", end="'")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Braceleft: enclose(start="{", end="}")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft: enclose(start="[", end="]")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Delete: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down: fold-expand-more-current** - Expand the current fold point one more level

**Alt-End: fold-expand-all** - Expand all fold points in the current file

**Alt-F11: prev-points-of-use-match**

**Alt-F12: next-points-of-use-match**

**Alt-F3: search** - Bring up the search manager in search mode.

**Alt-F4: close-window** - Close the current window and all documents and panels in it

**Alt-F5: run-to-cursor** - Run to current cursor position

**Alt-F6: run-failed-tests**

**Alt-F7: run-last-tests**

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Less: enclose(start=<“, end=>”)** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Parenleft: enclose(start=(“, end=”)”)** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl: enclose(start=““, end=”)”)** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Return: new-line** - Place a new line at the current cursor position

**Alt-Right: visit-history-next** - Move forward in history to next visited editor position

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Browse-!:** **filter-next-move** - Filter the lines covered by the next cursor move command through an external command and replace the lines with the result

**Browse-“:** **set-register** - Set the register to use for subsequent cut/copy/paste operations

**Browse-#:** `isearch-sel-backward(persist=0, whole_word=1)` - Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Browse-\$:** `end-of-line` - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**Browse-%:** `goto-percent-line(cursor="fnb")` - Position cursor at start of line at given percent in file. This uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fnb' for first non-blank character, or in VI mode it will do brace matching operation to reflect how VI overrides this command.

**Browse-&:** `repeat-replace` - Repeat the last query replace or range replace operation on the current line. The first match is replaced without confirmation.

**Browse-+:** `next-line-in-file(cursor="fnb")` - Move to next line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Browse-,::** `repeat-search-char(opposite=1)` - Repeat the last `search_char` operation, optionally in the opposite direction.

**Browse-.::** `repeat-command` - Repeat the last editor command

**Browse-/:** `isearch-forward-regex` - Action varies according to focus: *Active Editor Commands*: Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental regular expression mini-search forward from the cursor position, optionally entering the given search string.

**Browse-0:** `beginning-of-line(toggle=0)` - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When `toggle` is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Browse-1:** `initiate-numeric-modifier(digit=1)` - VI style repeat/numeric modifier for following command

**Browse-2:** `initiate-numeric-modifier(digit=2)` - VI style repeat/numeric modifier for following command

**Browse-3:** `initiate-numeric-modifier(digit=3)` - VI style repeat/numeric modifier for following command

**Browse-4: initiate-numeric-modifier(digit=4)** - VI style repeat/numeric modifier for following command

**Browse-5: initiate-numeric-modifier(digit=5)** - VI style repeat/numeric modifier for following command

**Browse-6: initiate-numeric-modifier(digit=6)** - VI style repeat/numeric modifier for following command

**Browse-7: initiate-numeric-modifier(digit=7)** - VI style repeat/numeric modifier for following command

**Browse-8: initiate-numeric-modifier(digit=8)** - VI style repeat/numeric modifier for following command

**Browse-9: initiate-numeric-modifier(digit=9)** - VI style repeat/numeric modifier for following command

**Browse-;: repeat-search-char** - Repeat the last search\_char operation, optionally in the opposite direction.

**Browse-<: outdent-next-move** - Outdent lines spanned by next cursor move

**Browse-=: indent-to-match-next-move** - Indent lines spanned by next cursor move to match, based on the preceding line

**Browse->: indent-next-move** - Indent lines spanned by next cursor move

**Browse-?: isearch-backward-regex** - Action varies according to focus: *Active Editor Commands*: Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental regular expression mini-search backward from the cursor position, optionally entering the given search string.

**Browse-@: execute-kbd-macro(register=None)** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Browse-A: enter-insert-mode(pos="after")** - Enter editor insert mode

**Browse-Apostrophe: vi-goto-bookmark** - Goto bookmark using single character name defined by the next pressed key

**Browse-BackSpace: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Browse-Ctrl-B: backward-page** - Move cursor backward one page

**Browse-Ctrl-C: vi-ctrl-c**

**Browse-Ctrl-D: scroll-text-down(repeat=0.5)** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Browse-Ctrl-E: scroll-text-down(move\_cursor=False)** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Browse-Ctrl-F: forward-page** - Move cursor forward one page

**Browse-Ctrl-I: visit-history-next** - Move forward in history to next visited editor position

**Browse-Ctrl-J: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-Ctrl-M: next-line-in-file(cursor="fmb")** - Move to next line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-Ctrl-N: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-Ctrl-O: visit-history-previous** - Move back in history to previous visited editor position

**Browse-Ctrl-P: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-Ctrl-Q: start-select-block** - Turn on auto-select block mode

**Browse-Ctrl-R: redo** - Redo last action

**Browse-Ctrl-Shift-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If skip\_if\_unique is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Browse-Ctrl-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Browse-Ctrl-U: scroll-text-up(repeat=0.5)** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Browse-Ctrl-V: vi-ctrl-v**

**Browse-Ctrl-W Browse-+: grow-split-vertically** - Increase height of this split

**Browse-Ctrl-W Browse-Ctrl-W: move-editor-focus** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-Ctrl-^: vi-split-edit-alternate**

**Browse-Ctrl-W Browse-Down: move-editor-focus(wrap=False)** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-Minus: shrink-split-vertically** - Decrease height of this split

**Browse-Ctrl-W Browse-Up: move-editor-focus(dir=-1, wrap=False)** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-W: move-editor-focus(dir=-1)** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-b: move-editor-focus-last** - Move focus to last editor split

**Browse-Ctrl-W Browse-c: unsplit(action="recent-or-close")** - Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  -- Change to recent buffer before unsplitting
recent-or-close -- Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future.

**Browse-Ctrl-W Browse-j: move-editor-focus(wrap=False)** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-k: `move-editor-focus(dir=-1, wrap=False)`** - Move focus to next or previous editor split, optionally wrapping when the end is reached.

**Browse-Ctrl-W Browse-n: `split-vertically(new=1)`** - Split current view vertically. Create new editor in new view when `new==1`.

**Browse-Ctrl-W Browse-o: `unsplit`** - Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  -- Change to recent buffer before unsplitting
recent-or-close -- Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future.

**Browse-Ctrl-W Browse-p: `move-editor-focus-previous`** - Move focus to previous editor split

**Browse-Ctrl-W Browse-q:** Multiple commands (first available is executed):

- **`unsplit(action="close")`** - Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  --
Change to recent buffer before unsplitting
recent-or-close --
Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future.

- **`close(close_window=1)`** - Close active document. Abandon any changes when `ignore_changes` is `True`. Close empty windows when `close_window` is `true` and quit if all document windows closed when `can_quit` is `true`.

**Browse-Ctrl-W Browse-s: `split-vertically`** - Split current view vertically. Create new editor in new view when `new==1`.

**Browse-Ctrl-W Browse-t: move-editor-focus-first** - Move focus to first editor split

**Browse-Ctrl-W Browse-v: split-horizontally** - Split current view horizontally.

**Browse-Ctrl-X: vi-ctrl-x**

**Browse-Ctrl-Y: scroll-text-up(move\_cursor=False)** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Browse-Ctrl-^: nth-document** - Move to the nth document alphabetically in the list of documents open in the current window

**Browse-Ctrl-h: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Browse-Esc: clear-move-command** - Clear any pending move command action, as for VI mode

**Browse-F: search-char(dir=1, single\_line=1)** - Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single\_line=1 to search only within the current line.

**Browse-G Browse-Shift-I: enter-insert-mode(pos="sol")** - Enter editor insert mode

**Browse-Grave: vi-goto-bookmark** - Goto bookmark using single character name defined by the next pressed key

**Browse-I: enter-insert-mode(pos="before")** - Enter editor insert mode

**Browse-Insert: enter-insert-mode(pos="before")** - Enter editor insert mode

**Browse-Minus: previous-line-in-file(cursor="fnb")** - Move to previous line in file, repositioning character within line: 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Browse-O: enter-insert-mode(pos="new-below")** - Enter editor insert mode

**Browse-Return: next-line(cursor="start")** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Browse-Shift-A: enter-insert-mode(pos="eol")** - Enter editor insert mode



**Browse-Shift-B: backward-word(delimiters=" tnr")** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Browse-Shift-C: delete-to-end-of-line-insert** - Delete everything between the cursor and end of line and enter insert move (when working in a modal editor key binding)

**Browse-Shift-D: delete-to-end-of-line(post\_offset=-1)** - Delete everything between the cursor and end of line

**Browse-Shift-E: forward-word(delimiters=" tnr", gravity="endml")** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Browse-Shift-F: search-char(dir=-1, single\_line=1)** - Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single\_line=1 to search only within the current line.

**Browse-Shift-G: goto-nth-line-default-end(cursor="fnb")** - Same as goto\_nth\_line but defaults to end of file if no lineno is given

**Browse-Shift-H: cursor-move-to-top** - Move cursor to top of display (without scrolling), optionally at an offset of given number of lines below top

**Browse-Shift-I: enter-insert-mode(pos="fnb")** - Enter editor insert mode

**Browse-Shift-J: join-lines** - Join together specified number of lines after current line (replace newlines with the given delimiter (single space by default))

**Browse-Shift-L: cursor-move-to-bottom** - Move cursor to bottom of display (without scrolling), optionally at an offset of given number of lines before bottom

**Browse-Shift-M: cursor-move-to-center** - Move cursor to center of display (without scrolling)

**Browse-Shift-N: isearch-repeat(reverse=1)** - Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True.

**Browse-Shift-O: enter-insert-mode(pos="new-above")** - Enter editor insert mode

**Browse-Shift-P: paste-register(pos=-1)** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before

or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set `pos = 1` to paste after, or `-1` to paste before. Set `indent=1` to indent the pasted text to match current line. Set `cursor=-1` to place cursor before lines or `cursor=1` to place it after lines after paste completes.

**Browse-Shift-R: enter-replace-mode** - Enter editor replace mode

**Browse-Shift-S: delete-line-insert** - Delete the current line or lines when the selection spans multiple lines or given repeat is  $> 1$ . Enters insert mode (when working with modal key bindings).

**Browse-Shift-T: search-char(dir=-1, pos=1, single\_line=1)** - Search for the given character. Searches to right if `dir > 0` and to left if `dir < 0`. Optionally place cursor `pos` characters to left or right of the target (e.g., use `-1` to place one to left). If repeat  $> 1$ , the Nth match is found. Set `single_line=1` to search only within the current line.

**Browse-Shift-V: start-select-line** - Turn on auto-select mode line by line

**Browse-Shift-W: forward-word(delimiters=" tnr")** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Browse-Shift-Y: move-to-register(unit="line")** - Cut or copy a specified number of characters or lines, or the current selection. Set `cut=1` to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

**Browse-Shift-Z Browse-Shift-Q: close(ignore\_changes=1, close\_window=1)** - Close active document. Abandon any changes when `ignore_changes` is True. Close empty windows when `close_window` is true and quit if all document windows closed when `can_quit` is true.

**Browse-Shift-Z Browse-Shift-Z: write-file-and-close(filename=None)** - Write current document to given location and close it. Saves to current file name if the given filename is None.

**Browse-Shift-x: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Browse-Space: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Browse-T: search-char(dir=1, pos=1, single\_line=1)** - Search for the given character. Searches to right if dir > 0 and to left if dir < 0. Optionally place cursor pos characters to left or right of the target (e.g., use -1 to place one to left). If repeat > 1, the Nth match is found. Set single\_line=1 to search only within the current line.

**Browse-Underscore: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Browse-[ Browse-p: paste-register(pos=-1, indent=1)** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes.

**Browse-\*: isearch-sel-forward(persist=0, whole\_word=1)** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**Browse-] Browse-p: paste-register(indent=1)** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes.

**Browse-^: beginning-of-line-text(toggle=0)** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Browse-b: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Browse-c: delete-next-move-insert** - Delete the text covered by the next cursor move command and then enter insert mode (when working in a modal editor key binding)

**Browse-colon: vi-command-by-name** - Execute a VI command (implements ":" commands from VI)

**Browse-d: delete-next-move** - Delete the text covered by the next cursor move command.

**Browse-e: forward-word(gravity="endm1")** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Browse-g Browse-\$: end-of-screen-line** - Move to end of current wrapped line

**Browse-g Browse-0: beginning-of-screen-line** - Move to beginning of current wrapped line

**Browse-g Browse-Shift-D: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**Browse-g Browse-Shift-E: backward-word(delimiters=" tnr", gravity="endm1")** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Browse-g Browse-Shift-J: join-lines(delim=" ")** - Join together specified number of lines after current line (replace newlines with the given delimiter (single space by default))

**Browse-g Browse-Shift-P: paste-register(pos=-1, cursor=1)** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes.

**Browse-g Browse-Shift-T: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Browse-g Browse-Shift-U: case-upper-next-move** - Change case of text spanned by next cursor movement to upper case

**Browse-g Browse-T: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Browse-g Browse-^: beginning-of-screen-line-text** - Move to first non-blank character at beginning of current wrapped line

**Browse-g Browse-d: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is `True`.

**Browse-g Browse-e: backward-word(gravity="endm1")** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Browse-g Browse-g: goto-nth-line(cursor="fnb")** - Position cursor at start of given line number (1=first, -1 = last). This differs from `goto-line` in that it never prompts for a line number but instead uses the previously entered numeric modifier or defaults to going to line one. The cursor can be positioned at 'start', 'end', or 'fnb' for first non-blank character.

**Browse-g Browse-j: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Browse-g Browse-k: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fnb' for first non-blank char.

**Browse-g Browse-m: middle-of-screen-line** - Move to middle of current wrapped line

**Browse-g Browse-p: paste-register(cursor=1)** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set `pos = 1` to paste after, or `-1` to paste before. Set `indent=1` to indent the pasted text to match current line. Set `cursor=-1` to place cursor before lines or `cursor=1` to place it after lines after paste completes.

**Browse-g Browse-q Browse-q: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Browse-g Browse-r: replace-char(line\_mode="extend")** - Replace `num` characters with given character. Set `line_mode` to `multiline` to allow replacing across lines, `extend` to replace on current line and then extend the line length, and `restrict` to replace only if enough characters exist on current line after cursor position.

**Browse-g Browse-u: case-lower-next-move** - Change case of text spanned by next cursor movement to lower case

**Browse-g Browse-v: previous-select** - Turn on auto-select using previous mode and selection

**Browse-g Browse-~: case-swap-next-move** - Change case of text spanned by next cursor movement so each letter is the opposite of its current case

**Browse-h: backward-char(wrap=0)** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Browse-j: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-k: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Browse-l: forward-char(wrap=0)** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Browse-m: vi-set-bookmark** - Set a bookmark at current location on the editor using the next key press as the name of the bookmark.

**Browse-n: isearch-repeat** - Repeat the most recent isearch, using same string and regex/text. Reverse direction when reverse is True.

**Browse-p: paste-register** - Paste text from register as before or after the current position. If the register contains only lines, then the lines are pasted before or after current line (rather than at cursor). If the register contains fragments of lines, the text is pasted over the current selection or either before or after the cursor. Set pos = 1 to paste after, or -1 to paste before. Set indent=1 to indent the pasted text to match current line. Set cursor=-1 to place cursor before lines or cursor=1 to place it after lines after paste completes.

**Browse-q:** Multiple commands (first available is executed):

- **start-kbd-macro(register=None)** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.
- **stop-kbd-macro** - Stop definition of a keyboard macro

**Browse-r: replace-char(line\_mode="restrict")** - Replace num characters with given character. Set line\_mode to multiline to allow replacing across lines, extend to replace on current line and then extend the line length, and restrict to replace only if enough characters exist on current line after cursor position.

**Browse-s: forward-delete-char-insert** - Delete one char in front of the cursor and enter insert mode (when working in modal key bindings)

**Browse-u: undo** - Undo last action

**Browse-v: start-select-char** - Turn on auto-select mode character by character

**Browse-w: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Browse-x: forward-delete-char-within-line** - Delete one character in front of the cursor unless at end of line, in which case delete backward. Do nothing if the line is empty. This is VI style 'x' in browser mode.

**Browse-y: move-to-register-next-move** - Move the text spanned by the next cursor motion to a register

**Browse-z Browse-.: center-cursor** - Scroll so cursor is centered on display

**Browse-z Browse-Minus: cursor-to-bottom** - Scroll so cursor is centered at bottom of display

**Browse-z Browse-Plus: cursor-to-top** - Scroll so cursor is centered at top of display

**Browse-z Browse-Return: cursor-to-top** - Scroll so cursor is centered at top of display

**Browse-z Browse-Shift-H: scroll-text-right(repeat=0.5)** - Scroll text right a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**Browse-z Browse-Shift-L: scroll-text-left(repeat=0.5)** - Scroll text left a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**Browse-z Browse-Shift-M: fold-collapse-all** - Collapse all fold points in the current file

**Browse-z Browse-Shift-O: fold-expand-all-current** - Expand the current fold point completely

**Browse-z Browse-Shift-R: fold-expand-all** - Expand all fold points in the current file

**Browse-z Browse-b: cursor-to-bottom** - Scroll so cursor is centered at bottom of display

**Browse-z Browse-c: fold-collapse-current** - Collapse the current fold point

**Browse-z Browse-h: scroll-text-right** - Scroll text right a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**Browse-z Browse-l: scroll-text-left** - Scroll text left a column w/o moving cursor's relative position on screen. Repeat is number of columns or if >0 and <1.0 then percent of screen.

**Browse-z Browse-m: vi-fold-less**

**Browse-z Browse-o: fold-expand-current** - Expand the current fold point

**Browse-z Browse-r: vi-fold-more**

**Browse-z Browse-t: cursor-to-top** - Scroll so cursor is centered at top of display

**Browse-z Browse-z: center-cursor** - Scroll so cursor is centered on display

**Browse-{: backward-paragraph** - Move cursor backward one paragraph (to next all-whitespace line).

**Browse-|: goto-column** - Move cursor to given column

**Browse-}: forward-paragraph** - Move cursor forward one paragraph (to next all-whitespace line).

**Browse-~: case-swap** - Change case of the current selection, or character ahead of the cursor if there is no selection, so each letter is the opposite of its current case

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.



**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set `toggle=True` to indent instead of one level higher if already at the matching position.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Up: goto-previous-bookmark(current\_file\_only=True)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if `>0` and `<1.0` then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when `ignore_changes` is True. Close empty windows when `close_window` is true and quit if all document windows closed when `can_quit` is true.

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Quoteleft: begin-visited-document-cycle(move\_back=True, back\_key="Ctrl-Quoteleft", forward\_key="Ctrl-AsciiTilde")** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or

”end“ to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-S: save** - Save active document. Also close it if close is True.

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-F: batch-search** - Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-Shift-F3: search-sel-backward** - Search backward using current selection

**Ctrl-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: debug-all-tests**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-G: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to

leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within

line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**F1:** Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F11: frame-up** - Move up the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F6: step-over** - Step over current execution point

**F7: step-into** - Step into current execution point, or start debugging at first line

**F8: step-out** - Step out of the current function or method

**F9:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: toggle-overtyping** - Toggle status of overtyping mode

**Insert-Ctrl-C: vi-ctrl-c**

**Insert-Ctrl-D: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Insert-Ctrl-H: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Insert-Ctrl-J: new-line** - Place a new line at the current cursor position

**Insert-Ctrl-M: new-line** - Place a new line at the current cursor position

**Insert-Ctrl-N: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Insert-Ctrl-O: enter-browse-mode(provisional=True)** - Enter editor browse mode

**Insert-Ctrl-P: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Insert-Ctrl-Q: start-select-block** - Turn on auto-select block mode

**Insert-Ctrl-T: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Insert-Ctrl-U: delete-to-start-of-line** - Delete everything between the cursor and start of line

**Insert-Ctrl-V: vi-ctrl-v**

**Insert-Ctrl-W: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Insert-Ctrl-X: vi-ctrl-x**

**Insert-Ctrl-[: enter-browse-mode** - Enter editor browse mode



**Insert-Esc: enter-browse-mode** - Enter editor browse mode

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Left: backward-char(wrap=0)** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Replace-Ctrl-C: enter-browse-mode** - Enter editor browse mode

**Replace-Ctrl-D: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Replace-Ctrl-H: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Replace-Ctrl-J: new-line** - Place a new line at the current cursor position

**Replace-Ctrl-M: new-line** - Place a new line at the current cursor position

**Replace-Ctrl-T: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Replace-Ctrl-U: delete-to-start-of-line** - Delete everything between the cursor and start of line

**Replace-Ctrl-W: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Replace-Ctrl-[: enter-browse-mode** - Enter editor browse mode

**Replace-Esc: enter-browse-mode** - Enter editor browse mode

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*:

Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Right: forward-char(wrap=0)** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N: diff-next**

**Shift-Alt-P: diff-previous**

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Down: forward-page** - Move cursor forward one page

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Com-*

*mands*: Move to end of current line, adjusting the selection range to new position ;  
*Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F2**: Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9**: Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to

leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Up: backward-page** - Move cursor backward one page

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-!: filter-selection** - Filter the current selection through an external command and replace the lines with the result

**Visual-1: initiate-numeric-modifier(digit=1)** - VI style repeat/numeric modifier for following command

**Visual-2: initiate-numeric-modifier(digit=2)** - VI style repeat/numeric modifier for following command

**Visual-3: initiate-numeric-modifier(digit=3)** - VI style repeat/numeric modifier for following command

**Visual-4: initiate-numeric-modifier(digit=4)** - VI style repeat/numeric modifier for following command

**Visual-5: initiate-numeric-modifier(digit=5)** - VI style repeat/numeric modifier for following command

**Visual-6: initiate-numeric-modifier(digit=6)** - VI style repeat/numeric modifier for following command

**Visual-7: initiate-numeric-modifier(digit=7)** - VI style repeat/numeric modifier for following command

**Visual-8: initiate-numeric-modifier(digit=8)** - VI style repeat/numeric modifier for following command

**Visual-9: initiate-numeric-modifier(digit=9)** - VI style repeat/numeric modifier for following command

**Visual-<: outdent-lines** - Outdent selected number of lines from cursor position. Set lines to None to indent all the lines in current selection. Set levels to outdent more than one level at a time.

**Visual->: indent-lines** - Indent selected number of lines from cursor position. Set lines to None to indent all the lines in current selection. Set levels to indent more than one level at a time.

**Visual-Ctrl-V: enter-browse-mode** - Enter editor browse mode

**Visual-Ctrl-[: exit-visual-mode** - Exit visual mode and return back to default mode

**Visual-Esc:** `exit-visual-mode` - Exit visual mode and return back to default mode

**Visual-Esc:** `exit-visual-mode` - Exit visual mode and return back to default mode

**Visual-Shift-A:** `enter-insert-mode(pos="after")` - Enter editor insert mode

**Visual-Shift-I:** `enter-insert-mode(pos="before")` - Enter editor insert mode

**Visual-Shift-J:** `join-selection` - Join together all lines in given selection (replace newlines with the given delimiter (single space by default))

**Visual-Shift-O:** `exchange-point-and-mark` - When currently marking text, this exchanges the current position and mark ends of the current selection

**Visual-Shift-R:** `enter-insert-mode(pos="delete-lines")` - Enter editor insert mode

**Visual-Shift-V:** `enter-browse-mode` - Enter editor browse mode

**Visual-Shift-Y:** `move-to-register(unit="line")` - Cut or copy a specified number of characters or lines, or the current selection. Set `cut=1` to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

**Visual-c:** `enter-insert-mode(pos="delete-sel")` - Enter editor insert mode

**Visual-colon:** `vi-command-by-name` - Execute a VI command (implements ":" commands from VI)

**Visual-d:** `move-to-register(unit="sel", cut=1)` - Cut or copy a specified number of characters or lines, or the current selection. Set `cut=1` to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

**Visual-g Visual-Shift-J:** `join-selection(delim=" ")` - Join together all lines in given selection (replace newlines with the given delimiter (single space by default))

**Visual-g Visual-q:** `fill-paragraph` - Attempt to auto-justify the paragraph around the current start of selection

**Visual-o:** `exchange-point-and-mark` - When currently marking text, this exchanges the current position and mark ends of the current selection

**Visual-r:** `replace-char` - Replace num characters with given character. Set `line_mode` to multiline to allow replacing across lines, extend to replace on current line and then extend the line length, and restrict to replace only if enough characters exist on current line after cursor position.

**Visual-s:** `enter-insert-mode(pos="delete-sel")` - Enter editor insert mode



**Visual-v: enter-browse-mode** - Enter editor browse mode

**Visual-x: move-to-register(unit="sel", cut=1)** - Cut or copy a specified number of characters or lines, or the current selection. Set cut=1 to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

**Visual-y: move-to-register(unit="sel")** - Cut or copy a specified number of characters or lines, or the current selection. Set cut=1 to remove the range of text from the editor after moving to register (otherwise it is just copied). Unit should be one of 'char' or 'line' or 'sel' for current selection.

## 13.4. Visual Studio Personality

This section documents all the default key bindings for the **Visual Studio** keyboard personality, set by the **Personality** preference.

**Alt-1: fold-python-methods**

**Alt-2: fold-python-classes**

**Alt-3: fold-python-classes-and-defs**

**Alt-Apostrophe: enclose(start="""", end="""")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Braceleft: enclose(start="{", end="}")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft: enclose(start="[", end="]")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Delete: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down: fold-expand-more-current** - Expand the current fold point one more level

**Alt-End: fold-expand-all** - Expand all fold points in the current file

**Alt-F11: prev-points-of-use-match**

**Alt-F12: next-points-of-use-match**

**Alt-F3: search** - Bring up the search manager in search mode.

**Alt-F4: close-window** - Close the current window and all documents and panels in it

**Alt-F5: run-to-cursor** - Run to current cursor position

**Alt-F6: run-failed-tests**

**Alt-F7: run-last-tests**

**Alt-F7: view-project-properties** - View or change project-wide properties

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Less: enclose(start="<", end=">")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Parenleft: enclose(start="(", end=")")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl: enclose(start="\"", end="\"")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Return: new-line** - Place a new line at the current cursor position

**Alt-Right: visit-history-next** - Move forward in history to next visited editor position

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**Alt-comma: query-replace** - Initiate incremental mini-search query/replace from the cursor position.

**Alt-period: replace-string** - Replace all occurrences of a string from the cursor position to end of file.

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Ctrl-0: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-7 C: use-lexer-cpp** - Force syntax highlighting for C/C++ source

**Ctrl-7 H: use-lexer-html** - Force syntax highlighting for HTML

**Ctrl-7 M: use-lexer-makefile** - Force syntax highlighting for make files

**Ctrl-7 N: use-lexer-none** - Use no syntax highlighting

**Ctrl-7 P: use-lexer-python** - Force syntax highlighting for Python source

**Ctrl-7 S: use-lexer-sql** - Force syntax highlighting for SQL

**Ctrl-7 X: use-lexer-xml** - Force syntax highlighting for XML files

**Ctrl-8: recent-document** - Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode.

**Ctrl-9: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set `toggle=True` to indent instead of one level higher if already at the matching position.

**Ctrl-A: select-all** - Select all text in the editor

**Ctrl-Alt-B: search-sel-backward** - Search backward using current selection

**Ctrl-Alt-Comma: query-replace-regex** - Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-F: search-sel-forward** - Search forward using current selection

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-G: goto-bookmark** - Goto named bookmark

**Ctrl-Alt-K: show-bookmarks** - Show a list of all currently defined bookmarks

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-M: set-bookmark** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-T: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Ctrl-Alt-Up: goto-previous-bookmark(current\_file\_only=True)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-period: replace-string-regex** - Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression.

**Ctrl-B: isearch-sel-forward** - Action varies according to focus: *Active Editor Com-*

*mands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-C: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Comma: next-window** - Switch to the next window alphabetically by title

**Ctrl-D: toolbar-search-focus** - Move focus to toolbar search entry.

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-E: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F: search** - Bring up the search manager in search mode.

**Ctrl-F10: debug-to-cursor**

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when `ignore_changes` is `True`. Close empty windows when `close_window` is `true` and quit if all document windows closed when `can_quit` is `true`.

**Ctrl-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-G: goto-line** - Position cursor at start of given line number

**Ctrl-H: replace** - Bring up the search manager in replace mode.

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-I: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-I: replace-and-search** - Replace current selection and search again.

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-J: show-autocompleter** - Show the auto-completer for current cursor position

**Ctrl-K Ctrl-C: comment-out-region** - Comment out the selected region. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting.

**Ctrl-K Ctrl-F: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Ctrl-K Ctrl-K: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Ctrl-K Ctrl-N: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is True.

**Ctrl-K Ctrl-O: open-from-keyboard** - Open a file from disk using keyboard-driven selection of the file

**Ctrl-K Ctrl-P: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is True.

**Ctrl-K Ctrl-S: switch-document** - Switches to named document. Name may either be the complete name or the last path component of a path name.

**Ctrl-K Ctrl-T: comment-toggle** - Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**Ctrl-K Ctrl-U: uncomment-out-region** - Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting.

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Multiply: visit-history-previous** - Move back in history to previous visited editor position

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-L: cut-line** - Cut the current line(s) to clipboard.

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-M: execute-kbd-macro** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Ctrl-Minus: visit-history-previous** - Move back in history to previous visited editor position

**Ctrl-N: new-file** - Create a new file

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-O: open-gui** - Open a file from disk, prompting with file selection dialog if necessary

**Ctrl-P: print-view** - Print active editor document

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page



**Ctrl-Q: quit** - Quit the application.

**Ctrl-Quoteleft:** `begin-visited-document-cycle(move_back=True, back_key="Ctrl-Quoteleft", forward_key="Ctrl-AsciiTilde")` - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-R: replace** - Bring up the search manager in replace mode.

**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-S: save** - Save active document. Also close it if close is True.

**Ctrl-Shift-B: isearch-sel-backward** - Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-F: batch-search** - Search on current selection using the Search in Files tool. The `look_in` argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either `use_selection` is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-Shift-F3: search-sel-backward** - Search backward using current selection

**Ctrl-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: debug-all-tests**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-G: search-backward** - Search again using the search manager's current settings in backward direction

**Ctrl-Shift-H: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If skip\_if\_unique is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-R: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-S: save-all** - Save all unsaved items, prompting for names for any new items that don't have a filename already.

**Ctrl-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-U: case-upper** - Change case of the current selection, or character ahead of the cursor if there is no selection, to upper case

**Ctrl-Shift-U: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Slash: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-U: case-lower** - Change case of the current selection, or character ahead of the cursor if there is no selection, to lower case

**Ctrl-Underscore: visit-history-next** - Move forward in history to next visited editor position

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-V: paste** - Action varies according to focus: *Active Editor Commands*: Paste

text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Ctrl-W: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-X: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Y: redo** - Redo last action

**Ctrl-Z: undo** - Undo last action

**Ctrl-]: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-greater: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-less: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-parenleft: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**Ctrl-parenright: stop-kbd-macro** - Stop definition of a keyboard macro

**Ctrl-space: show-autocompleter** - Show the auto-completer for current cursor position

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move

to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**F1**: Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F10**: **step-over** - Step over current execution point

**F11**: **frame-up** - Move up the current debug stack

**F11**: **step-into** - Step into current execution point, or start debugging at first line

**F12**: **frame-down** - Move down the current debug stack

**F2**: Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3**: **search-forward** - Search again using the search manager's current settings in forward direction

**F4**: **goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is `True`.

**F5**: **debug-continue** - Continue (or start) running, to next breakpoint

**F6**: **step-over** - Step over current execution point

**F7**: **step-into** - Step into current execution point, or start debugging at first line

**F8**: **step-out** - Step out of the current function or method

**F9**: Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home**: **beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If `toggle` is `True`, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*:

Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next:** **forward-page** - Move cursor forward one page

**Page\_Down:** **forward-page** - Move cursor forward one page

**Page\_Up:** **backward-page** - Move cursor backward one page

**Prior:** **backward-page** - Move cursor backward one page

**Return:** **new-line** - Place a new line at the current cursor position

**Right:** **forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A:** **diff-merge-a-b**

**Shift-Alt-B:** **diff-merge-b-a**

**Shift-Alt-Down:** **next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Left:** **backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N:** **diff-next**

**Shift-Alt-P:** **diff-previous**

**Shift-Alt-Right:** **forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up:** **previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-BackSpace:** **backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8:** **start-select-block** - Turn on auto-select block mode

**Shift-Delete:** **cut** - Action varies according to focus: *Active Editor Commands*: Cut



selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Delete: cut-selection-or-line** - Cut the current selection or current line if there is no selection. The text is placed on the clipboard.

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F11: step-out** - Step out of the current function or method

**Shift-F2**: Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F5: debug-kill** - Stop debugging

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-Home:** **beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert:** **paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete:** **cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down:** **next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End:** **end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Home:** **beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert:** **paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left:** **backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next:** **forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down:** **forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

## 13.5. OS X Personality

This section documents all the default key bindings for the OS X keyboard personality, set by the **Personality** preference.

**Backspace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Command-0: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Command-1: activate-file-option-menu** - Activate the file menu for the editor.

**Command-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Command-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Command-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Command-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Command-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Command-7 C: use-lexer-cpp** - Force syntax highlighting for C/C++ source

**Command-7 H: use-lexer-html** - Force syntax highlighting for HTML

**Command-7 M: use-lexer-makefile** - Force syntax highlighting for make files

**Command-7 N: use-lexer-none** - Use no syntax highlighting

**Command-7 P: use-lexer-python** - Force syntax highlighting for Python source

**Command-7 S: use-lexer-sql** - Force syntax highlighting for SQL

**Command-7 X: use-lexer-xml** - Force syntax highlighting for XML files

**Command-8: recent-document** - Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode.

**Command-9: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Command-A: select-all** - Select all text in the editor

**Command-Apostrophe: comment-out-region** - Comment out the selected region. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting.

**Command-B: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Command-Backslash: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set toggle=True to indent instead of one level higher if already at the matching position.

**Command-Bracketleft: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Command-Bracketright: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Command-C: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Command-Comma: show-preferences-gui** - Edit the preferences file using the preferences GUI, optionally opening to the section that contains the given preference by name

**Command-Ctrl-KP\_Divide: fold-python-classes**

**Command-Ctrl-KP\_Multiply: fold-expand-all** - Expand all fold points in the current file

**Command-Ctrl-KP\_Subtract: fold-collapse-all** - Collapse all fold points in the current file

**Command-Ctrl-KP\_Subtract: fold-python-methods**

**Command-Ctrl-R: replace-and-search** - Replace current selection and search again.

**Command-D: set-bookmark** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Command-Down: end-of-document** - Move cursor to end of document

**Command-E: search-sel-forward** - Search forward using current selection

**Command-F: search** - Bring up the search manager in search mode.

**Command-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Command-F3: search-sel-forward** - Search forward using current selection

**Command-F4: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Command-F5: debug-kill** - Stop debugging

**Command-F8: start-select-line** - Turn on auto-select mode line by line

**Command-F9: break-clear-all** - Clear all breakpoints

**Command-G: search-forward** - Search again using the search manager's current settings in forward direction

**Command-H: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Command-I: view-file-properties** - View project properties for a particular file (current file if none is given)

**Command-J: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Command-KP\_Add: fold-expand-more-current** - Expand the current fold point one more level

**Command-KP\_Divide: fold-toggle** - Toggle the current fold point

**Command-KP\_Enter: new-line** - Place a new line at the current cursor position

**Command-KP\_Equal:** **fold-python-classes-and-defs**

**Command-KP\_Multiply:** **fold-expand-all-current** - Expand the current fold point completely

**Command-KP\_Subtract:** **fold-collapse-all-current** - Collapse the current fold point completely

**Command-L:** **goto-line** - Position cursor at start of given line number

**Command-Left:** **beginning-of-line** - Action varies according to focus: *Active Editor Commands:* Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands:* Move to the beginning of the toolbar search entry

**Command-M:** **execute-kbd-macro** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Command-N:** **new-file** - Create a new file

**Command-O:** **open-gui** - Open a file from disk, prompting with file selection dialog if necessary

**Command-Option-F6:** **debug-failed-tests**

**Command-Option-F7:** **debug-last-tests**

**Command-P:** **print-view** - Print active editor document

**Command-Pointer\_Button1:** **goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Command-Q:** **quit** - Quit the application.

**Command-Question:** **show-document** - Show the given documentation section

**Command-Quotedbl:** **uncomment-out-region** - Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting.

**Command-R:** **replace** - Bring up the search manager in replace mode.

**Command-Return:** **new-line** - Place a new line at the current cursor position

**Command-Right:** **end-of-line** - Action varies according to focus: *Active Editor Commands:* Move to end of current line; *Toolbar Search Commands:* Move to the end of the toolbar search entry

**Command-S: save** - Save active document. Also close it if close is True.

**Command-Semicolon: comment-toggle** - Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**Command-Shift-D: goto-bookmark** - Goto named bookmark

**Command-Shift-Down: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Command-Shift-F: batch-search** - Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided

**Command-Shift-F3: search-sel-backward** - Search backward using current selection

**Command-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Command-Shift-F6: debug-all-tests**

**Command-Shift-F7: debug-current-tests**

**Command-Shift-G: search-backward** - Search again using the search manager's current settings in backward direction

**Command-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Command-Shift-K: show-bookmarks** - Show a list of all currently defined bookmarks

**Command-Shift-KP\_Subtract: fold-collapse-more-current** - Collapse the current fold point one more level

**Command-Shift-Left: beginning-of-line-extend** - Action varies according to focus:  
*Active Editor Commands:* Move to beginning of current line, adjusting the selection range to the new position. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands:* Move to the beginning of the toolbar search entry, extending the selection

**Command-Shift-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and



if it is None, the selected text or the symbol under the cursor is used. If `skip_if_unique` is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Command-Shift-R: batch-replace** - Display search and replace in files tool.

**Command-Shift-Right: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Command-Shift-S: save-as** - Save active document to a new file

**Command-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Command-Shift-U: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Command-Shift-Up: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Command-Shift-W: close** - Close active document. Abandon any changes when `ignore_changes` is True. Close empty windows when `close_window` is true and quit if all document windows closed when `can_quit` is true.

**Command-Shift-Z: redo** - Redo last action

**Command-T: search** - Bring up the search manager in search mode.

**Command-U: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Command-Up: start-of-document** - Move cursor to start of document

**Command-V: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Command-W: close** - Close active document. Abandon any changes when `ignore_changes` is True. Close empty windows when `close_window` is true and quit if all document windows closed when `can_quit` is true.

**Command-X: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Command-Z: undo** - Undo last action

**Command-parenleft: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**Command-parenright: stop-kbd-macro** - Stop definition of a keyboard macro

**Ctrl=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set toggle=True to indent instead of one level higher if already at the matching position.

**Ctrl-Comma: visit-history-previous** - Move back in history to previous visited editor position

**Ctrl-Down: forward-page** - Move cursor forward one page

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Enter: new-line** - Place a new line at the current cursor position

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-Left: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Ctrl-Option-Delete: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-Option-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within

line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Option-KP\_Delete: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-Option-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Ctrl-Option-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Ctrl-Option-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Period: visit-history-next** - Move forward in history to next visited editor position

**Ctrl-R: query-replace** - Initiate incremental mini-search query/replace from the cursor position.

**Ctrl-Return: new-line** - Place a new line at the current cursor position

**Ctrl-Right: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**Ctrl-Shift-Left: beginning-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line, adjusting the selection range to the new position. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry, extending the selection

**Ctrl-Shift-Right: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving be-

tween documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Up: backward-page** - Move cursor backward one page

**Ctrl-a: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Ctrl-b: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Ctrl-d: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Ctrl-e: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**Ctrl-f: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Ctrl-h: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Ctrl-k: kill-line** - Kill rest of line from cursor to end of line, and place it into the clipboard with any other contiguously removed lines. End-of-line is removed only if there is nothing between the cursor and the end of the line.

**Ctrl-n: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-p: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-space: show-autocompleter** - Show the auto-completer for current cursor position

**Ctrl-v: forward-page** - Move cursor forward one page

**Ctrl-y: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-document** - Move cursor to end of document

**F1:** Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F11: frame-up** - Move up the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F6: step-over** - Step over current execution point

**F7: step-into** - Step into current execution point, or start debugging at first line

**F8: step-out** - Step out of the current function or method

**F9:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line

- **break-clear** - Clear the breakpoint on the current line

**Home: start-of-document** - Move cursor to start of document

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_End: end-of-document** - Move cursor to end of document

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: start-of-document** - Move cursor to start of document

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Option-Backspace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Option-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Option-F3: search** - Bring up the search manager in search mode.

**Option-F4: close-window** - Close the current window and all documents and panels in it

**Option-F6: run-failed-tests**

**Option-F7: run-last-tests**

**Option-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Option-KP\_Enter: new-line** - Place a new line at the current cursor position

**Option-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Option-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Option-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Option-Page\_Down: forward-page** - Move cursor forward one page

**Option-Page\_Up: backward-page** - Move cursor backward one page

**Option-Return: new-line** - Place a new line at the current cursor position

**Option-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Option-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Option-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Option-Up: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Backspace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Command-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F2:** Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: new-document-window** - Create a new document window with same documents and panels as in the current document window (if any; otherwise empty with default panels)

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line



- **break-disable** - Disable the breakpoint on current line

**Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Shift-KP\_Enter: new-line** - Place a new line at the current cursor position

**Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line** - Place a new line at the current cursor position

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

## 13.6. Eclipse (Experimental) Personality

This section documents all the default key bindings for the **Eclipse (Experimental)** keyboard personality, set by the **Personality** preference.

**Alt-1: fold-python-methods**

**Alt-1: fold-python-methods**

**Alt-2: fold-python-classes**

**Alt-2: fold-python-classes**

**Alt-3: fold-python-classes-and-defs**

**Alt-3: fold-python-classes-and-defs**

**Alt-Apostrophe: enclose(start="'", end="'")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Apostrophe: enclose(start="'", end="'")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Braceleft: enclose(start="{", end="}")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Braceleft: enclose(start="{", end="}")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft: enclose(start="[", end="]")** - Enclose the selection or the rest of

the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft: `enclose(start="“, end="”“)`** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Delete: `backward-delete-word`** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Delete: `backward-delete-word`** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down: `fold-expand-more-current`** - Expand the current fold point one more level

**Alt-Down: `fold-expand-more-current`** - Expand the current fold point one more level

**Alt-Down: `move-line-down(indent=True)`** - Move the current line or lines up down line, optionally indenting to match the new position

**Alt-End: `fold-expand-all`** - Expand all fold points in the current file

**Alt-End: `fold-expand-all`** - Expand all fold points in the current file

**Alt-Enter: `view-file-properties`** - View project properties for a particular file (current file if none is given)

**Alt-F11: `prev-points-of-use-match`**

**Alt-F11: `prev-points-of-use-match`**

**Alt-F12: `next-points-of-use-match`**

**Alt-F12: `next-points-of-use-match`**

**Alt-F3: `search`** - Bring up the search manager in search mode.

**Alt-F3: `search`** - Bring up the search manager in search mode.

**Alt-F4: `close-window`** - Close the current window and all documents and panels in it

**Alt-F4: `close-window`** - Close the current window and all documents and panels in it

**Alt-F5: `run-to-cursor`** - Run to current cursor position

**Alt-F5: run-to-cursor** - Run to current cursor position

**Alt-F6: run-failed-tests**

**Alt-F6: run-failed-tests**

**Alt-F7: run-last-tests**

**Alt-F7: run-last-tests**

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Less: enclose(start="<", end=">")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Less: enclose(start="<", end=">")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Parenleft: enclose(start="(", end=")")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Parenleft: enclose(start="(", end=")")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl: `enclose(start="", end="")`** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl: `enclose(start="", end="")`** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Return: `new-line`** - Place a new line at the current cursor position

**Alt-Return: `new-line`** - Place a new line at the current cursor position

**Alt-Right: `visit-history-next`** - Move forward in history to next visited editor position

**Alt-Right: `visit-history-next`** - Move forward in history to next visited editor position

**Alt-Right: `visit-history-next`** - Move forward in history to next visited editor position

**Alt-Shift-Down: `select-less`** - Select less code; undoes the last select-more command

**Alt-Shift-L: `introduce-variable`**

**Alt-Shift-Left: `previous-statement`** - Select the previous statement. Will ignore indented statements under the current statements unless `ignore_indented` is False. Specify a count of more than 1 to go back multiple statements.

**Alt-Shift-M: `extract-def`**

**Alt-Shift-O: `show_preferences_gui(prefname="edit.highlight-occurrences")`**

**Alt-Shift-R: `rename-symbol`**

**Alt-Shift-Right: `next-statement`** - Select the next statement. Will ignore indented statements under the current statements unless `ignore_indented` is False. Specify a count of more than 1 to go forward multiple statements.

**Alt-Shift-T: `show-panel(panel_type="refactoring")`** - Show most recently visited panel instance of given type. If no such panel exists, add one to the primary window and show it. Returns the panel view object or None if not shown. Focus is shifted to panel if `grab_focus` is specified and is true; if `grab_focus` is not specified, it defaults to the value of `flash`.

The valid panel types are:

project (\*) browser (\*\*) batch-search (\*) interactive-search source-assistant (\*\*) debug-data debug-stack debug-io debug-exceptions debug-breakpoints (\*\*) debug-probe (\*\*) debug-watch (\*\*) debug-modules (\*\*) python-shell messages (\*) help indent (\*\*) bookmarks (\*\*) testing (\*\*) open-files (\*) os-command (\*\*) snippets (\*\*) diff (\*\*) uses (\*\*)

refactoring (\*\*) versioncontrol.svn (\*\*) versioncontrol.hg (\*\*) versioncontrol.git (\*\*) versioncontrol.bzr (\*\*) versioncontrol.cvs (\*\*) versioncontrol.perforce (\*\*)

(\*) Wing Personal and Pro only (\*\*) Wing Pro only

Alt-Shift-U: `show_preferences_gui(prefname="edit.highlight-occurrences")`

**Alt-Shift-Up: select-more** - Select more code on either the current line or larger multi-line blocks.

**Alt-Shift-V: move-symbol**

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**Alt-Up: move-line-up(indent=True)** - Move the current line or lines up one line, optionally indenting to match the new position

**Alt-comma: query-replace** - Initiate incremental mini-search query/replace from the cursor position.

**Alt-period: replace-string** - Replace all occurrences of a string from the cursor position to end of file.

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor*  
*Commands*: Delete one character behind the cursor, or the current selection if not empty.  
; *Toolbar Search Commands*: Delete character behind the cursor

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor*  
*Commands*: Delete one character behind the cursor, or the current selection if not empty.  
; *Toolbar Search Commands*: Delete character behind the cursor

Ctrl-/: comment-block-toggle

**Ctrl-0: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl-7 C: use-lexer-cpp** - Force syntax highlighting for C/C++ source

**Ctrl-7 H: use-lexer-html** - Force syntax highlighting for HTML

**Ctrl-7 M: use-lexer-makefile** - Force syntax highlighting for make files

**Ctrl-7 N: use-lexer-none** - Use no syntax highlighting

**Ctrl-7 P: use-lexer-python** - Force syntax highlighting for Python source

**Ctrl-7 S: use-lexer-sql** - Force syntax highlighting for SQL

**Ctrl-7 X: use-lexer-xml** - Force syntax highlighting for XML files

**Ctrl-8: recent-document** - Switches to previous document most recently visited in the current window or window set if in one-window-per-editor windowing mode.

**Ctrl-9: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-=: indent-to-match** - Indent the current line or selected region to match inden-

tation of preceding non-blank line. Set `toggle=True` to indent instead of one level higher if already at the matching position.

**Ctrl-=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set `toggle=True` to indent instead of one level higher if already at the matching position.

**Ctrl-A: select-all** - Select all text in the editor

**Ctrl-Alt-B: search-sel-backward** - Search backward using current selection

**Ctrl-Alt-Comma: query-replace-regex** - Initiate incremental mini-search query/replace from the cursor position. The search string is treated as a regular expression.

**Ctrl-Alt-D: evaluate-sel-in-debug-probe** - Evaluate the current selection from the editor within the Debug Probe tool. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead.

**Ctrl-Alt-Down: duplicate-line** - Duplicate the current line or lines. Places the duplicate on the line following the selection if `pos` is 'below' or before the selection if it is 'above'.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-E: evaluate-sel-in-shell** - Evaluate the current selection from the editor within the Python Shell tool, optionally restarting the shell first. When `whole_lines` is set, the selection is rounded to whole lines before evaluation. When unspecified (set to `None`), the setting from the Shell's Option menu is used instead.

**Ctrl-Alt-F: search-sel-forward** - Search forward using current selection

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-G: goto-bookmark** - Goto named bookmark



**Ctrl-Alt-K: show-bookmarks** - Show a list of all currently defined bookmarks

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-M: set-bookmark** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-T: toggle-bookmark** - Set or remove a bookmark at current location on the editor. When set, the name of the bookmark is set to an auto-generated default.

**Ctrl-Alt-Up: duplicate-line-above** - Duplicate the current line or lines above the selection.

**Ctrl-Alt-Up: goto-previous-bookmark(`current_file_only=True`)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Up: goto-previous-bookmark(`current_file_only=True`)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-V: evaluate-file-in-shell** - Run the contents of the editor within the Python Shell

**Ctrl-Alt-period: replace-string-regex** - Replace all occurrences of a string from the cursor position to end of file. The search string is treated as a regular expression.

**Ctrl-B: isearch-sel-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, using current selection as the search string. Set `persist=False` to do the search but end the interactive search session immediately.

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-C: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Comma: next-window** - Switch to the next window alphabetically by title

**Ctrl-D: delete-line** - Delete the current line or lines when the selection spans multiple lines or given repeat is  $> 1$

**Ctrl-D: toolbar-search-focus** - Move focus to toolbar search entry.

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-E: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-E: show-panel(panel\_type="open-files")** - Show most recently visited panel instance of given type. If no such panel exists, add one to the primary window and show it. Returns the panel view object or None if not shown. Focus is shifted to panel if `grab_focus` is specified and is true; if `grab_focus` is not specified, it defaults to the value of `flash`.

The valid panel types are:

project (\*) browser (\*\*) batch-search (\*) interactive-search source-assistant (\*\*) debug-data debug-stack debug-io debug-exceptions debug-breakpoints (\*\*) debug-probe (\*\*) debug-watch (\*\*) debug-modules (\*\*) python-shell messages (\*) help indent (\*\*) bookmarks (\*\*) testing (\*\*) open-files (\*) os-command (\*\*) snippets (\*\*) diff (\*\*) uses (\*\*) refactoring (\*\*) versioncontrol.svn (\*\*) versioncontrol.hg (\*\*) versioncontrol.git (\*\*) versioncontrol.bzr (\*\*) versioncontrol.cvs (\*\*) versioncontrol.perforce (\*\*)

(\*) Wing Personal and Pro only (\*\*) Wing Pro only

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F: search** - Bring up the search manager in search mode.

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-F4: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F5: run-to-cursor** - Run to current cursor position

**Ctrl-F6: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-G: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is `True`.

**Ctrl-G: search-forward** - Search again using the search manager's current settings in forward direction

**Ctrl-H: batch-search** - Search on current selection using the Search in Files tool. The `look_in` argument gets entered in the look in field if not `None` or `''`. The current selection is put into the search field if it doesn't span multiple lines and either `use_selection` is `true` or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-H: replace** - Bring up the search manager in replace mode.

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-I: replace-and-search** - Replace current selection and search again.

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-J: fill-paragraph** - Attempt to auto-justify the paragraph around the current start of selection

**Ctrl-J: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-K: open-from-keyboard** - Open a file from disk using keyboard-driven selection of the file

**Ctrl-K: search-forward** - Search again using the search manager's current settings in forward direction

**Ctrl-KP\_Add: fold-expand-current** - Expand the current fold point

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Divide: fold-toggle** - Toggle the current fold point

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Multiply: fold-expand-all** - Expand all fold points in the current file

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: fold-collapse-current** - Collapse the current fold point

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-L: goto-line** - Position cursor at start of given line number

**Ctrl-L: goto-line** - Position cursor at start of given line number

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-M**: Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit.fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Ctrl-M: execute-kbd-macro** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**Ctrl-N: new-file** - Create a new file

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-O: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-O: open-gui** - Open a file from disk, prompting with file selection dialog if necessary

**Ctrl-P: print-view** - Print active editor document

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Period: comment-toggle** - Toggle commenting out of the selected lines. The style of commenting can be controlled with the style argument: 'indented' uses the default comment style indented at end of leading white space and 'block' uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used.

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Q: quit** - Quit the application.

**Ctrl-Q: visit-history-previous** - Move back in history to previous visited editor position

**Ctrl-Quoteleft:** `begin-visited-document-cycle(move_back=True, back_key="Ctrl-Quoteleft", forward_key="Ctrl-AsciiTilde")` - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Quoteleft:** `begin-visited-document-cycle(move_back=True, back_key="Ctrl-Quoteleft", forward_key="Ctrl-AsciiTilde")` - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-R: replace** - Bring up the search manager in replace mode.

**Ctrl-R: run-to-cursor** - Run to current cursor position

**Ctrl-Return: new-line-after** - Place a new line after the current line



**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-S: save** - Save active document. Also close it if close is True.

**Ctrl-Shift-B**: Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Ctrl-Shift-B: isearch-sel-backward** - Initiate incremental mini-search backward from the cursor position, using current selection as the search string. Set persist=False to do the search but end the interactive search session immediately.

**Ctrl-Shift-C: comment-block-toggle**

**Ctrl-Shift-C: delete-line** - Delete the current line or lines when the selection spans multiple lines or given repeat is > 1

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: move-line-down** - Move the current line or lines up down line, optionally indenting to match the new position

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Down: next-scope** - Select the next scope. Specify a count of more than

1 to go forward multiple scopes. If `sibling_only` is true, move only to other scopes of the same parent.

**Ctrl-Shift-E: `focus-current-editor`** - Move focus back to the current editor, out of any tool, if there is an active editor.

**Ctrl-Shift-End: `end-of-document-extend`** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-End: `end-of-document-extend`** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-Enter: `new-line-before`** - Place a new line before the current line

**Ctrl-Shift-F: `batch-search`** - Search on current selection using the Search in Files tool. The `look_in` argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either `use_selection` is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-Shift-F: `fill-paragraph`** - Attempt to auto-justify the paragraph around the current start of selection

**Ctrl-Shift-F2: `close-all`** - Close all documents in the current window, or in all windows if in one-window-per-editor windowing policy. Leave currently visible documents (or active window in one-window-per-editor-mode) if `omit_current` is True. Abandons changes rather than saving them when `ignore_changes` is True. Close empty window and quit if all document windows closed when `close_window` is True.

**Ctrl-Shift-F3: `search-sel-backward`** - Search backward using current selection

**Ctrl-Shift-F3: `search-sel-backward`** - Search backward using current selection

**Ctrl-Shift-F4: `close-all`** - Close all documents in the current window, or in all windows if in one-window-per-editor windowing policy. Leave currently visible documents (or active window in one-window-per-editor-mode) if `omit_current` is True. Abandons changes rather than saving them when `ignore_changes` is True. Close empty window and quit if all document windows closed when `close_window` is True.

**Ctrl-Shift-F5: `debug-stop`** - Pause free-running execution at current program counter

**Ctrl-Shift-F5: `debug-stop`** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: `debug-all-tests`**

**Ctrl-Shift-F6: `debug-all-tests`**

**Ctrl-Shift-F7: `debug-current-tests`**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-G: find-points-of-use**

**Ctrl-Shift-G: search-backward** - Search again using the search manager's current settings in backward direction

**Ctrl-Shift-H: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-I: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-ISO\_Left\_Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-Insert: toggle-overtyping** - Toggle status of overtyping mode

**Ctrl-Shift-J: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, op-

tionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Shift-K: search-backward** - Search again using the search manager's current settings in backward direction

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-L: swap-lines** - Swap the line at start of current selection with the line that follows it, or the preceding line if previous is True.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-O: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If `skip_if_unique` is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Ctrl-Shift-P: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-R: batch-replace** - Display search and replace in files tool.

**Ctrl-Shift-R: open-from-project** - Open document from the project via the Open From Project dialog. The given fragment is used as the initial fragment filter and if it is None, the selected text or the symbol under the cursor is used. If `skip_if_unique` is true, the file is opened without the dialog being displayed if only one filename matches the fragment.

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-S: save-all** - Save all unsaved items, prompting for names for any new items that don't have a filename already.

**Ctrl-Shift-S: save-as** - Save active document to a new file

**Ctrl-Shift-Space: show-panel(panel\_type="source-assistant")** - Show most recently visited panel instance of given type. If no such panel exists, add one to the primary window and show it. Returns the panel view object or None if not shown. Focus is shifted to panel if `grab_focus` is specified and is true; if `grab_focus` is not specified, it defaults to the value of `flash`.

The valid panel types are:

project (\*) browser (\*\*) batch-search (\*) interactive-search source-assistant (\*\*) debug-data debug-stack debug-io debug-exceptions debug-breakpoints (\*\*) debug-probe (\*\*) debug-watch (\*\*) debug-modules (\*\*) python-shell messages (\*) help indent (\*\*) bookmarks (\*\*) testing (\*\*) open-files (\*) os-command (\*\*) snippets (\*\*) diff (\*\*) uses (\*\*) refactoring (\*\*) versioncontrol.svn (\*\*) versioncontrol.hg (\*\*) versioncontrol.git (\*\*) versioncontrol.bzr (\*\*) versioncontrol.cvs (\*\*) versioncontrol.perforce (\*\*)

(\*) Wing Personal and Pro only (\*\*) Wing Pro only

**Ctrl-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-Shift-T: find-symbol** - Allow user to visit point of definition of a source symbol in the current editor context by typing a fragment of the name

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-U: batch-search(look\_in="Current File")** - Search on current selection using the Search in Files tool. The look\_in argument gets entered in the look in field if not None or ". The current selection is put into the search field if it doesn't span multiple lines and either use\_selection is true or there's nothing in the search field. The given search text is used instead, if provided

**Ctrl-Shift-U: isearch-backward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search backward from the cursor position, optionally entering the given search string.

**Ctrl-Shift-Up: move-line-up** - Move the current line or lines up one line, optionally indenting to match the new position

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Up: previous-scope** - Select the previous scope. Specify a count of more than 1 to go backward multiple scopes. If sibling\_only is true, move only to other scopes of the same parent.

**Ctrl-Shift-V: duplicate-line** - Duplicate the current line or lines. Places the duplicate on the line following the selection if pos is 'below' or before the selection if it is 'above'.

**Ctrl-Shift-W: close-all** - Close all documents in the current window, or in all windows if in one-window-per-editor windowing policy. Leave currently visible documents (or active window in one-window-per-editor-mode) if omit\_current is True. Abandons changes



rather than saving them when `ignore_changes` is `True`. Close empty window and quit if all document windows closed when `close_window` is `True`.

**Ctrl-Shift-X: lower-case**

**Ctrl-Shift-Y: duplicate-line-above** - Duplicate the current line or lines above the selection.

**Ctrl-Shift-Y: upper-case**

**Ctrl-Slash: comment-out-region** - Comment out the selected region. The style of commenting can be controlled with the `style` argument: `'indented'` uses the default comment style indented at end of leading white space and `'block'` uses a block comment in column zero. If not given, the style configured with the Editor / Block Comment Style preference is used. Each call adds a level of commenting.

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-U: execute-file** - Execute the file at the given location or use the active view if `loc` is `None`.

**Ctrl-U: isearch-forward** - Action varies according to focus: *Active Editor Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string ; *Document Viewer Commands*: Initiate incremental mini-search forward from the cursor position, optionally entering the given search string.

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if `>0` and `<1.0` then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if `>0` and `<1.0` then percent of screen. Set

move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-V: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Ctrl-W: close** - Close active document. Abandon any changes when ignore\_changes is True. Close empty windows when close\_window is true and quit if all document windows closed when can\_quit is true.

**Ctrl-X: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Y: redo** - Redo last action

**Ctrl-Z: undo** - Undo last action

**Ctrl-\*\*: \*\*uncomment-out-region** - Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting.

**Ctrl-]: brace-match** - Match brace at current cursor position, selecting all text between the two and highlighting the braces

**Ctrl-greater: indent-region** - Indent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-less: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Ctrl-parenleft: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**Ctrl-parenright: stop-kbd-macro** - Stop definition of a keyboard macro

**Ctrl-question: uncomment-out-region** - Uncomment out the selected region if commented out. If one\_level is True then each call removes only one level of commenting.

**Ctrl-space: show-autocompleter** - Show the auto-completer for current cursor position

**Ctrl-|: indent-lines(lines=1)** - Indent selected number of lines from cursor position.

Set lines to None to indent all the lines in current selection. Set levels to indent more than one level at a time.

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**F1**: Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F1**: Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F11: debug-continue** - Continue (or start) running, to next breakpoint

**F11: frame-up** - Move up the current debug stack

**F11: frame-up** - Move up the current debug stack

**F12: focus-current-editor** - Move focus back to the current editor, out of any tool, if there is an active editor.

**F12: frame-down** - Move down the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F4: show-panel(panel\_type="browser")** - Show most recently visited panel instance of given type. If no such panel exists, add one to the primary window and show it. Returns the panel view object or None if not shown. Focus is shifted to panel if grab\_focus is specified and is true; if grab\_focus is not specified, it defaults to the value of flash.

The valid panel types are:

project (\*) browser (\*\*) batch-search (\*) interactive-search source-assistant (\*\*) debug-data debug-stack debug-io debug-exceptions debug-breakpoints (\*\*) debug-probe (\*\*) debug-watch (\*\*) debug-modules (\*\*) python-shell messages (\*) help indent (\*\*) bookmarks (\*\*) testing (\*\*) open-files (\*) os-command (\*\*) snippets (\*\*) diff (\*\*) uses (\*\*) refactoring (\*\*) versioncontrol.svn (\*\*) versioncontrol.hg (\*\*) versioncontrol.git (\*\*) versioncontrol.bzr (\*\*) versioncontrol.cvs (\*\*) versioncontrol.perforce (\*\*)

(\*) Wing Personal and Pro only (\*\*) Wing Pro only

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F5: step-into** - Step into current execution point, or start debugging at first line

**F6: step-over** - Step over current execution point

**F6: step-over** - Step over current execution point

**F6: step-over** - Step over current execution point

**F7: step-into** - Step into current execution point, or start debugging at first line

**F7: step-into** - Step into current execution point, or start debugging at first line

**F7: step-out** - Step out of the current function or method

**F8: debug-continue** - Continue (or start) running, to next breakpoint

**F8: step-out** - Step out of the current function or method

**F8: step-out** - Step out of the current function or method

**F9:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**F9:** Multiple commands (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: toggle-overtyp** - Toggle status of overtyping mode

**Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Insert: toggle-overtyp** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*:

Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next: forward-page** - Move cursor forward one page

**Next: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Return: new-line** - Place a new line at the current cursor position

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.



**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N: diff-next**

**Shift-Alt-N: diff-next**

**Shift-Alt-P: diff-previous**

**Shift-Alt-P: diff-previous**

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut

selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-Enter: new-line-after** - Place a new line after the current line

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F2**: Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F2**: Multiple commands (first available is executed):

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas

and toolbar, saving previous state so it can be restored later with `exit_fullscreen`

- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F6: run-all-tests**

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F7: run-current-tests**

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position.

If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Return: new-line-before** - Place a new line before the current line

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Tab: backward-tab** - Outdent line at current position

**Shift-Tab: outdent-region** - Outdent the selected region one level of indentation. Set sel to None to use preference to determine selection behavior, or "never-select" to unselect after indent, "always-select" to always select after indent, or "retain-select" to retain current selection after indent.

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

**Visual-Esc: exit-visual-mode** - Exit visual mode and return back to default mode

## 13.7. Brief Personality

This section documents all the default key bindings for the **Brief** keyboard personality, set by the **Personality** preference.

**Alt-0: set-bookmark(mark="0")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-1: fold-python-methods**

**Alt-1: set-bookmark(mark="1")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-2: fold-python-classes**

**Alt-2: set-bookmark(mark="2")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-3: fold-python-classes-and-defs**

**Alt-3: set-bookmark(mark="3")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-4: set-bookmark(mark="4")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-5: set-bookmark(mark="5")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-6: set-bookmark(mark="6")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-7: set-bookmark(mark="7")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-8: set-bookmark(mark="8")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-9: set-bookmark(mark="9")** - Set a bookmark at current location on the editor. Mark is the project-wide textual name of the bookmark.

**Alt-A: toggle-mark-command(select\_right=2)**



**Alt-Apostrophe:** `enclose(start="""", end="""")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-BackSpace:** `backward-delete-word` - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Braceleft:** `enclose(start="{", end="}")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Bracketleft:** `enclose(start="[", end="]")` - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-C:** `toggle-mark-command(style="block")`

**Alt-D:** `delete-selected-lines`

**Alt-D:** `kill-line` - Kill rest of line from cursor to end of line, and place it into the clipboard with any other contiguously removed lines. End-of-line is removed only if there is nothing between the cursor and the end of the line.

**Alt-Delete:** `backward-delete-word` - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Alt-Down:** `fold-expand-more-current` - Expand the current fold point one more level

**Alt-E:** `open-gui` - Open a file from disk, prompting with file selection dialog if necessary

**Alt-End:** `fold-expand-all` - Expand all fold points in the current file

**Alt-F11:** `prev-points-of-use-match`

**Alt-F12:** `next-points-of-use-match`

**Alt-F3:** `search` - Bring up the search manager in search mode.

**Alt-F4:** `close-window` - Close the current window and all documents and panels in it

**Alt-F5:** `run-to-cursor` - Run to current cursor position

**Alt-F5:** `search-sel-backward` - Search backward using current selection

**Alt-F6:** `run-failed-tests`

**Alt-F7: run-last-tests**

**Alt-G: goto-line** - Position cursor at start of given line number

**Alt-H: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and `other_split` is `True`.

**Alt-Home: fold-collapse-all** - Collapse all fold points in the current file

**Alt-I: toggle-overtyping** - Toggle status of overtyping mode

**Alt-J: show-bookmarks** - Show a list of all currently defined bookmarks

**Alt-K: kill-line** - Kill rest of line from cursor to end of line, and place it into the clipboard with any other contiguously removed lines. End-of-line is removed only if there is nothing between the cursor and the end of the line.

**Alt-L: toggle-mark-command(style="line")**

**Alt-Left: visit-history-previous** - Move back in history to previous visited editor position

**Alt-Less: enclose(start="<", end=">")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-M: toggle-mark-command(select\_right=1)**

**Alt-Minus: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Alt-N: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Alt-Page\_Down: fold-expand-all-current** - Expand the current fold point completely

**Alt-Page\_Up: fold-collapse-all-current** - Collapse the current fold point completely

**Alt-Parenleft: enclose(start="(", end=")")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-Quotedbl: enclose(start="\"", end="\"")** - Enclose the selection or the rest of the current line when there is no selection with the given start and end strings. The caret is moved to the end of the enclosed text.

**Alt-R: insert-file** - Insert a file at current cursor position, prompting user for file selection

**Alt-Return: new-line** - Place a new line at the current cursor position

**Alt-Right: visit-history-next** - Move forward in history to next visited editor position

**Alt-S: search** - Bring up the search manager in search mode.

**Alt-Slash: fold-toggle** - Toggle the current fold point

**Alt-T: replace** - Bring up the search manager in replace mode.

**Alt-U: undo** - Undo last action

**Alt-Up: fold-collapse-more-current** - Collapse the current fold point one more level

**Alt-W: save** - Save active document. Also close it if close is True.

**Alt-X: quit** - Quit the application.

**BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty.  
; *Toolbar Search Commands*: Delete character behind the cursor

**Ctrl-1: activate-file-option-menu** - Activate the file menu for the editor.

**Ctrl-2: activate-symbol-option-menu-1** - Activate the 1st symbol menu for the editor.

**Ctrl-3: activate-symbol-option-menu-2** - Activate the 2nd symbol menu for the editor.

**Ctrl-4: activate-symbol-option-menu-3** - Activate the 3rd symbol menu for the editor.

**Ctrl-5: activate-symbol-option-menu-4** - Activate the 4th symbol menu for the editor.

**Ctrl-6: activate-symbol-option-menu-5** - Activate the 5th symbol menu for the editor.

**Ctrl=: indent-to-match** - Indent the current line or selected region to match indentation of preceding non-blank line. Set toggle=True to indent instead of one level higher if already at the matching position.

**Ctrl-Alt-Down: goto-next-bookmark(current\_file\_only=True)** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when current\_file\_only is True.

**Ctrl-Alt-F6: debug-failed-tests**

**Ctrl-Alt-F7: debug-last-tests**

**Ctrl-Alt-Left: goto-previous-bookmark** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Right: goto-next-bookmark** - Go to the next bookmark, or the first one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-Alt-Up: goto-previous-bookmark(current\_file\_only=True)** - Go to the previous bookmark in the bookmark list, or the last one if no bookmark is selected. Stays within the file in the current editor when `current_file_only` is `True`.

**Ctrl-BackSpace: backward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word behind of the cursor ; *Toolbar Search Commands*: Delete word behind the cursor

**Ctrl-C: center-cursor** - Scroll so cursor is centered on display

**Ctrl-C: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-D: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-Down: scroll-text-down** - Scroll text down a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-E: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if  $>0$  and  $<1.0$  then percent of screen. Set `move_cursor` to `False` to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-End: end-of-document** - Move cursor to end of document

**Ctrl-F12: command-by-name** - Execute given command by name, collecting any args as needed

**Ctrl-F3: search-sel-forward** - Search forward using current selection

**Ctrl-F4: close** - Close active document. Abandon any changes when `ignore_changes` is True. Close empty windows when `close_window` is true and quit if all document windows closed when `can_quit` is true.

**Ctrl-F5: debug-kill** - Stop debugging

**Ctrl-F8: start-select-line** - Turn on auto-select mode line by line

**Ctrl-F9: break-clear-all** - Clear all breakpoints

**Ctrl-Home: start-of-document** - Move cursor to start of document

**Ctrl-Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-K: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Add: zoom-in** - Zoom in, increasing the text display size temporarily by one font size

**Ctrl-KP\_Delete: forward-delete-word** - Action varies according to focus: *Active Editor Commands*: Delete one word in front of the cursor ; *Toolbar Search Commands*: Delete word in front of the cursor

**Ctrl-KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-KP\_End: end-of-document** - Move cursor to end of document

**Ctrl-KP\_Home: start-of-document** - Move cursor to start of document

**Ctrl-KP\_Insert: copy** - Action varies according to focus: *Active Editor Commands*: Copy selected text ; *Document Viewer Commands*: Copy any selected text. ; *Exceptions Commands*: Copy the exception traceback to the clipboard ; *Search Manager Instance Commands*: Copy selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-KP\_Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains

the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-KP\_Next: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Down: forward-page** - Move cursor forward one page

**Ctrl-KP\_Page\_Up: backward-page** - Move cursor backward one page

**Ctrl-KP\_Prior: backward-page** - Move cursor backward one page

**Ctrl-KP\_Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-KP\_Subtract: zoom-out** - Zoom out, increasing the text display size temporarily by one font size

**Ctrl-KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Left: backward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word

**Ctrl-Minus: kill-buffer** - Close the current text file

**Ctrl-Next: forward-page** - Move cursor forward one page

**Ctrl-PageDown: end-of-document** - Move cursor to end of document

**Ctrl-PageUp: beginning-of-document**

**Ctrl-Page\_Down: next-document** - Move to the next document alphabetically in the list of documents open in the current window

**Ctrl-Page\_Up: previous-document** - Move to the previous document alphabetically in the list of documents open in the current window

**Ctrl-Pointer\_Button1: goto-clicked-symbol-defn** - Goto the definition of the source symbol that was last clicked on

**Ctrl-Prior: backward-page** - Move cursor backward one page

**Ctrl-Quoteleft:** `begin-visited-document-cycle(move_back=True, back_key="Ctrl-Quoteleft", forward_key="Ctrl-AsciiTilde")` - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-R: initiate-repeat-4** - Enter a sequence of digits indicating number of times to repeat the subsequent command or keystroke.

**Ctrl-Return: new-line-after** - Place a new line after the current line

**Ctrl-Right: forward-word** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word

**Ctrl-Shift-Delete: delete-lines**

**Ctrl-Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-F3: search-sel-backward** - Search backward using current selection

**Ctrl-Shift-F5: debug-stop** - Pause free-running execution at current program counter

**Ctrl-Shift-F6: debug-all-tests**

**Ctrl-Shift-F7: debug-current-tests**

**Ctrl-Shift-F9:** Multiple commands (first available is executed):

- **break-disable-all** - Disable all breakpoints
- **break-enable-all** - Enable all breakpoints

**Ctrl-Shift-Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-I: add-current-file-to-project** - Add the frontmost currently open file to project

**Ctrl-Shift-ISO\_Left\_Tab:** `begin-visited-document-cycle(move_back=False)` -

Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-KP\_End: end-of-document-extend** - Move cursor to end of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Home: start-of-document-extend** - Move cursor to start of document, adjusting the selection range to new position

**Ctrl-Shift-KP\_Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-KP\_Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-Shift-Left: backward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one word, adjusting the selection range to



new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move backward one word, extending the selection

**Ctrl-Shift-Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Ctrl-Shift-Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Ctrl-Shift-Right: forward-word-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one word, adjusting the selection range to new position. Optionally, provide a string that contains the delimiters to define which characters are part of a word. Gravity may be "start" or "end" to indicate whether cursor is placed at start or end of the word.; *Toolbar Search Commands*: Move forward one word, extending the selection

**Ctrl-Shift-Tab: begin-visited-document-cycle(move\_back=False)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-Shift-Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Ctrl-T: forward-tab** - Action varies according to focus: *Active Editor Commands*: Place a tab character at the current cursor position ; *Search Manager Instance Commands*: Place a forward tab at the current cursor position in search or replace string

**Ctrl-Tab: begin-visited-document-cycle(move\_back=True)** - Start moving between documents in the order they were visited. Starts modal key interaction that ends when a key other than tab is seen or ctrl is released.

**Ctrl-U: redo** - Redo last action

**Ctrl-Up: scroll-text-up** - Scroll text up a line w/o moving cursor's relative position on screen. Repeat is number of lines or if >0 and <1.0 then percent of screen. Set move\_cursor to False to leave cursor in current position within the source, otherwise it is moved so the cursor remains on same screen line.

**Ctrl-V: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Ctrl-X: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Ctrl-Z: undo** - Undo last action

**Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**End: cursor-end**

**End: end-of-document** - Move cursor to end of document

**End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**End End End: end-of-document** - Move cursor to end of document

**F1:** Multiple commands (first available is executed):

- **show-horizontal-tools** - Show the horizontal tool area
- **minimize-horizontal-tools** - Minimize the horizontal tool area

**F10: command-by-name** - Execute given command by name, collecting any args as needed

**F11: frame-up** - Move up the current debug stack

**F12: frame-down** - Move down the current debug stack

**F2:** Multiple commands (first available is executed):

- **show-vertical-tools** - Show the vertical tool area
- **minimize-vertical-tools** - Minimize the vertical tool area

**F3: search-forward** - Search again using the search manager's current settings in forward direction

**F3: split-vertically** - Split current view vertically. Create new editor in new view when new==1.

**F4: goto-selected-symbol-defn** - Goto the definition of the selected source symbol, optionally showing the definition in another split if one is available and other\_split is True.

**F4: unsplit** - Unsplit all editors so there's only one. Action specifies how to choose the remaining displayed editor. One of:

```
current -- Show current editor
close   -- Close current editor before unsplitting
recent  -- Change to recent buffer before unsplitting
recent-or-close -- Change to recent buffer before closing
split, or close the current buffer if there is only
one split left.
```

NOTE: The parameters for this command are subject to change in the future.

**F5: debug-continue** - Continue (or start) running, to next breakpoint

**F5: search** - Bring up the search manager in search mode.

**F6: replace** - Bring up the search manager in replace mode.

**F6: step-over** - Step over current execution point

**F7: start-kbd-macro** - Start definition of a keyboard macro. If register=None then the user is prompted to enter a letter a-z under which to file the macro. Otherwise, register 'a' is used by default.

**F7: step-into** - Step into current execution point, or start debugging at first line

**F8: execute-kbd-macro** - Execute most recently recorded keyboard macro. If register is None then the user is asked to enter a letter a-z for the register where the macro is filed. Otherwise, register 'a' is used by default.

**F8: step-out** - Step out of the current function or method

**F9: Multiple commands** (first available is executed):

- **break-set** - Set a new regular breakpoint on current line
- **break-clear** - Clear the breakpoint on the current line

**Home: beginning-of-line-text** - Move to end of the leading white space, if any, on

the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Home: cursor-home**

**Home: start-of-document** - Move cursor to start of document

**Home Home Home: start-of-document** - Move cursor to start of document

**ISO\_Left\_Tab: backward-tab** - Outdent line at current position

**Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Insert: toggle-overtyping** - Toggle status of overtyping mode

**KP\_Add: copy-line** - Copy the current lines(s) to clipboard

**KP\_Delete: forward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character in front of the cursor ; *Toolbar Search Commands*: Delete character in front of the cursor

**KP\_Down: next-line** - Move to screen next line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**KP\_End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**KP\_Enter: new-line** - Place a new line at the current cursor position

**KP\_Home: beginning-of-line-text** - Move to end of the leading white space, if any, on the current line. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**KP\_Insert: toggle-overtyping** - Toggle status of overtyping mode

**KP\_Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**KP\_Multiply: undo** - Undo last action

**KP\_Next: forward-page** - Move cursor forward one page

**KP\_Page\_Down: forward-page** - Move cursor forward one page

**KP\_Page\_Up: backward-page** - Move cursor backward one page

**KP\_Prior: backward-page** - Move cursor backward one page

**KP\_Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**KP\_Subtract: cut-line** - Cut the current line(s) to clipboard.

**KP\_Tab: tab-key** - Implement the tab key, the action of which is configurable by preference

**KP\_Up: previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Left: backward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character ; *Toolbar Search Commands*: Move backward one character

**Next: forward-page** - Move cursor forward one page

**Page\_Down: forward-page** - Move cursor forward one page

**Page\_Up: backward-page** - Move cursor backward one page

**Prior: backward-page** - Move cursor backward one page

**Return: new-line** - Place a new line at the current cursor position

**Right: forward-char** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character ; *Toolbar Search Commands*: Move forward one character

**Shift-Alt-A: diff-merge-a-b**

**Shift-Alt-B: diff-merge-b-a**

**Shift-Alt-Down: next-line-extend-rect** - Move to next screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Alt-Left: backward-char-extend-rect** - Move cursor backward one character, adjusting the rectangular selection range to new position

**Shift-Alt-N: diff-next**

**Shift-Alt-P: diff-previous**

**Shift-Alt-Right: forward-char-extend-rect** - Move cursor forward one character, adjusting the rectangular selection range to new position

**Shift-Alt-Up: previous-line-extend-rect** - Move to previous screen line, adjusting the rectangular selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-BackSpace: backward-delete-char** - Action varies according to focus: *Active Editor Commands*: Delete one character behind the cursor, or the current selection if not empty. ; *Toolbar Search Commands*: Delete character behind the cursor

**Shift-Ctrl-F8: start-select-block** - Turn on auto-select block mode

**Shift-Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-End: end-of-line** - Action varies according to focus: *Active Editor Commands*: Move to end of current line; *Toolbar Search Commands*: Move to the end of the toolbar search entry

**Shift-End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-F1: move-focus** - Move the keyboard focus forward within the Window to the next editable area

**Shift-F11: frame-show** - Show the position (thread and stack frame) where the debugger originally stopped

**Shift-F2: Multiple commands (first available is executed):**

- **enter-fullscreen** - Hide both the vertical and horizontal tool areas and toolbar, saving previous state so it can be restored later with `exit_fullscreen`
- **exit-fullscreen** - Restore previous non-fullscreen state of all tools and tool bar

**Shift-F3: search-backward** - Search again using the search manager's current settings in backward direction

**Shift-F4: find-points-of-use**

**Shift-F5: debug-file** - Start debugging the current file (rather than the main entry point)

**Shift-F5: search-forward** - Search again using the search manager's current settings in forward direction

**Shift-F6: replace-and-search** - Replace current selection and search again.

**Shift-F6: run-all-tests**

**Shift-F7: run-current-tests**

**Shift-F7: stop-kbd-macro** - Stop definition of a keyboard macro

**Shift-F8: start-select-char** - Turn on auto-select mode character by character

**Shift-F9:** Multiple commands (first available is executed):

- **break-enable** - Enable the breakpoint on the current line
- **break-disable** - Disable the breakpoint on current line

**Shift-Home: beginning-of-line** - Action varies according to focus: *Active Editor Commands*: Move to beginning of current line. When toggle is True, moves to the end of the leading white space if already at the beginning of the line (and vice versa).; *Toolbar Search Commands*: Move to the beginning of the toolbar search entry

**Shift-Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Delete: cut** - Action varies according to focus: *Active Editor Commands*: Cut selected text ; *Search Manager Instance Commands*: Cut selected text ; *Toolbar Search Commands*: Cut selection

**Shift-KP\_Down: next-line-extend** - Move to next screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-KP\_End: end-of-line-extend** - Action varies according to focus: *Active Editor Commands*: Move to end of current line, adjusting the selection range to new position ; *Toolbar Search Commands*: Move to the end of the toolbar search entry, extending the selection

**Shift-KP\_Home: beginning-of-line-text-extend** - Move to end of the leading white space, if any, on the current line, adjusting the selection range to the new position. If toggle is True, moves to the beginning of the line if already at the end of the leading white space (and vice versa).

**Shift-KP\_Insert: paste** - Action varies according to focus: *Active Editor Commands*: Paste text from clipboard ; *Search Manager Instance Commands*: Paste text from clipboard ; *Toolbar Search Commands*: Paste from clipboard

**Shift-KP\_Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection

**Shift-KP\_Next: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Down: forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-KP\_Page\_Up: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Prior: backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-KP\_Right: forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-KP\_Up: previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Shift-Left: backward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor backward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move backward one character, extending the selection



**Shift-Next:** **forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Down:** **forward-page-extend** - Move cursor forward one page, adjusting the selection range to new position

**Shift-Page\_Up:** **backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Prior:** **backward-page-extend** - Move cursor backward one page, adjusting the selection range to new position

**Shift-Return:** **new-line-before** - Place a new line before the current line

**Shift-Right:** **forward-char-extend** - Action varies according to focus: *Active Editor Commands*: Move cursor forward one character, adjusting the selection range to new position ; *Toolbar Search Commands*: Move forward one character, extending the selection

**Shift-Tab:** **backward-tab** - Outdent line at current position

**Shift-Up:** **previous-line-extend** - Move to previous screen line, adjusting the selection range to new position, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Tab:** **tab-key** - Implement the tab key, the action of which is configurable by preference

**Up:** **previous-line** - Move to previous screen line, optionally repositioning character within line: 'same' to leave in same horizontal position, 'start' at start, 'end' at end, or 'fmb' for first non-blank char.

**Visual-Esc:** **exit-visual-mode** - Exit visual mode and return back to default mode



# License Information

Wing IDE is a commercial product that is based on a number of open source technologies. Although the product source code is available for Wing IDE Professional users (with signed non-disclosure agreement) the product is not itself open source.

The following sections describe the licensing of the product as a whole (the End User License Agreement) and provide required legal statements for the incorporated open source components.

## 14.1. Wing IDE Software License

This End User License Agreement (EULA) is a CONTRACT between you (either an individual or a single entity) and Wingware, which covers your use of "Wing IDE Personal" and related software components. All such software is referred to herein as the "Software Product." A software license and a license key or serial number ("Software Product License"), issued to a designated user only by Wingware or its authorized agents, is required for each user of the Software Product. If you do not agree to the terms of this EULA, then do not install or use the Software Product or the Software Product License. By explicitly accepting this EULA you are acknowledging and agreeing to be bound by the following terms:

### 1. EVALUATION LICENSE WARNING

This Software Product can be used in conjunction with a free evaluation Software Product License. If you are using such an evaluation Software Product License, you may use the Software Product only to evaluate its suitability for purchase. Evaluation Software Product Licenses have an expiration date and most of the features of the software will be disabled after that date. WINGWARE BEARS NO LIABILITY FOR ANY DAMAGES RESULTING FROM USE (OR ATTEMPTED USE AFTER THE EXPIRATION DATE) OF THE SOFTWARE PRODUCT, AND HAS NO DUTY TO PROVIDE ANY SUPPORT BEFORE OR AFTER THE EXPIRATION DATE OF AN EVALUATION LICENSE.

## 2. GRANT OF NON-EXCLUSIVE LICENSE

Wingware grants the non-exclusive, non-transferable right for a single user to use this Software Product for each license purchased. Each additional user of the Software Product requires an additional Software Product License. This includes users working on operating systems where the Software Product is compiled from source code by the user or a third party.

You may make copies of the Software Product as reasonably necessary for its use. Each copy must reproduce all copyright and other proprietary rights notices on or in the Software Product.

You may install your Software Product License only on computer systems and user accounts that are used by you, the licensee. You may also make copies of the Software Product License as necessary for backup and/or archival purposes. No other copies or installations may be made.

All rights not expressly granted to you are retained by Wingware.

## 3. INTELLECTUAL PROPERTY RIGHTS RESERVED BY WINGWARE

The Software Product is owned by Wingware and is protected by United States and international copyright laws and treaties, as well as other intellectual property laws and treaties. You must not remove or alter any copyright notices on any copies of the Software Product. This Software Product copy is licensed, not sold. You may not use, copy, or distribute the Software Product, except as granted by this EULA, without written authorization from Wingware or its designated agents. Furthermore, this EULA does not grant you any rights in connection with any trademarks or service marks of Wingware. Wingware reserves all intellectual property rights, including copyrights, and trademark rights.

## 4. NO RIGHT TO TRANSFER

You may not rent, lease, lend, or in any way distribute or transfer any rights in this EULA or the Software Product to third parties.

## 5. INDEMNIFICATION

You hereby agree to indemnify Wingware against and hold harmless Wingware from any claims, lawsuits or other losses that arise out of your breach of any provision of this EULA.

## 6. THIRD PARTY RIGHTS

Any software provided along with the Software Product that is associated with a separate license agreement is licensed to you under the terms of that license agreement. This license does not apply to those portions of the Software Product. Copies of these third party licenses are included in all copies of the Software Product.

## 7. SUPPORT SERVICES

Wingware may provide you with support services related to the Software Product. Use of any such support services is governed by Wingware policies and programs described in online documentation and/or other Wingware-provided materials.

As part of these support services, Wingware may make available bug lists, planned feature lists, and other supplemental informational materials. WINGWARE MAKES NO WARRANTY OF ANY KIND FOR THESE MATERIALS AND ASSUMES NO LIABILITY WHATSOEVER FOR DAMAGES RESULTING FROM ANY USE OF THESE MATERIALS. FURTHERMORE, YOU MAY NOT USE ANY MATERIALS PROVIDED IN THIS WAY TO SUPPORT ANY CLAIM MADE AGAINST WINGWARE.

Any supplemental software code or related materials that Wingware provides to you as part of the support services, in periodic updates to the Software Product or otherwise, is to be considered part of the Software Product and is subject to the terms and conditions of this EULA.

With respect to any technical information you provide to Wingware as part of the support services, Wingware may use such information for its business purposes without restriction, including for product support and development. Wingware will not use such technical information in a form that personally identifies you without first obtaining your permission.

## 9. TERMINATION WITHOUT PREJUDICE TO ANY OTHER RIGHTS

Wingware may terminate this EULA if you fail to comply with any term or condition of this EULA. In such event, you must destroy all copies of the Software Product and Software Product Licenses.

## 10. U.S. GOVERNMENT USE

If the Software Product is licensed under a U.S. Government contract, you acknowledge that the software and related documentation are "commercial items," as defined in 48 C.F.R. 2.01, consisting of "commercial computer software" and "commercial computer software documentation," as such terms are used in 48 C.F.R. 12.212 and 48 C.F.R. 227.7202-1. You also acknowledge that the software is "commercial computer software"

as defined in 48 C.F.R. 252.227-7014(a)(1). U.S. Government agencies and entities and others acquiring under a U.S. Government contract shall have only those rights, and shall be subject to all restrictions, set forth in this EULA. Contractor/manufacturer is Wingware, P.O. Box 400527, Cambridge, MA 02140-0006, USA.

## 11. EXPORT RESTRICTIONS

You will not download, export, or re-export the Software Product, any part thereof, or any software, tool, process, or service that is the direct product of the Software Product, to any country, person, or entity -- even to foreign units of your own company -- if such a transfer is in violation of U.S. export restrictions.

## 12. NO WARRANTIES

YOU ACCEPT THE SOFTWARE PRODUCT AND SOFTWARE PRODUCT LICENSE "AS IS," AND WINGWARE AND ITS THIRD PARTY SUPPLIERS AND LICENSORS MAKE NO WARRANTY AS TO ITS USE, PERFORMANCE, OR OTHERWISE. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, WINGWARE AND ITS THIRD PARTY SUPPLIERS AND LICENSORS DISCLAIM ALL OTHER REPRESENTATIONS, WARRANTIES, AND CONDITIONS, EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NON-INFRINGEMENT. THE ENTIRE RISK ARISING OUT OF USE OR PERFORMANCE OF THE SOFTWARE PRODUCT REMAINS WITH YOU.

## 13. LIMITATION OF LIABILITY

THIS LIMITATION OF LIABILITY IS TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW. IN NO EVENT SHALL WINGWARE OR ITS THIRD PARTY SUPPLIERS AND LICENSORS BE LIABLE FOR ANY COSTS OF SUBSTITUTE PRODUCTS OR SERVICES, OR FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, OR LOSS OF BUSINESS INFORMATION) ARISING OUT OF THIS EULA OR THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF WINGWARE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, WINGWARE'S, AND ITS THIRD PARTY SUPPLIERS' AND LICENSORS', ENTIRE LIABILITY ARISING OUT OF THIS EULA SHALL BE LIMITED TO THE LESSER OF THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE

PRODUCT OR THE PRODUCT LIST PRICE; PROVIDED, HOWEVER, THAT IF YOU HAVE ENTERED INTO A WINGWARE SUPPORT SERVICES AGREEMENT, WINGWARE'S ENTIRE LIABILITY REGARDING SUPPORT SERVICES SHALL BE GOVERNED BY THE TERMS OF THAT AGREEMENT.

#### 14. HIGH RISK ACTIVITIES

The Software Product is not fault-tolerant and is not designed, manufactured or intended for use or resale as on-line control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communication systems, air traffic control, direct life support machines, or weapons systems, in which the failure of the Software Product, or any software, tool, process, or service that was developed using the Software Product, could lead directly to death, personal injury, or severe physical or environmental damage ("High Risk Activities"). Accordingly, Wingware and its suppliers and licensors specifically disclaim any express or implied warranty of fitness for High Risk Activities. You agree that Wingware and its suppliers and licensors will not be liable for any claims or damages arising from the use of the Software Product, or any software, tool, process, or service that was developed using the Software Product, in such applications.

#### 15. GOVERNING LAW; ENTIRE AGREEMENT ; DISPUTE RESOLUTION

This EULA is governed by the laws of the Commonwealth of Massachusetts, U.S.A., excluding the application of any conflict of law rules. The United Nations Convention on Contracts for the International Sale of Goods shall not apply.

This EULA is the entire agreement between Wingware and you, and supersedes any other communications or advertising with respect to the Software Product; this EULA may be modified only by written agreement signed by authorized representatives of you and Wingware.

Unless otherwise agreed in writing, all disputes relating to this EULA (excepting any dispute relating to intellectual property rights) shall be subject to final and binding arbitration in the State of Massachusetts, in accordance with the Licensing Agreement Arbitration Rules of the American Arbitration Association, with the losing party paying all costs of arbitration. Arbitration must be by a member of the American Arbitration Association. If any dispute arises under this EULA, the prevailing party shall be reimbursed by the other party for any and all legal fees and costs associated therewith.

#### 16. GENERAL

If any provision of this EULA is held invalid, the remainder of this EULA shall continue in full force and effect.

A waiver by either party of any term or condition of this EULA or any breach thereof, in any one instance, shall not waive such term or condition or any subsequent breach thereof.

## 17. OUTSIDE THE U.S.

If you are located outside the U.S., then the provisions of this Section shall apply. Les parties aux présentes confirment leur volonté que cette convention de même que tous les documents y compris tout avis qui s’y rattache, soient rédigés en langue anglaise. (translation: "The parties confirm that this EULA and all related documentation is and will be in the English language.") You are responsible for complying with any local laws in your jurisdiction which might impact your right to import, export or use the Software Product, and you represent that you have complied with any regulations or registration procedures required by applicable law to make this license enforceable.

## 18. TRADEMARKS

The following are trademarks or registered trademarks of Wingware: Wingware, the feather logo, Wing IDE, Wing IDE 101, Wing IDE Personal, Wing IDE Professional, Wing IDE Enterprise, Wing Debugger, and "The Intelligent Development Environment for Python Programmers"

## 19. CONTACT INFORMATION

If you have any questions about this EULA, or if you want to contact Wingware for any reason, please direct all correspondence to: Wingware, P.O. Box 400527, Cambridge, MA 02140-0006, United States of America or send email to [info at wingware.com](mailto:info@wingware.com).

# 14.2. Open Source License Information

Wing IDE incorporates the following open source technologies, most of which are under [OSI Certified Open Source](#) licenses except as indicated in the footnotes:

- [atk](#) -- GUI accessibility toolkit by Bill.Haneman, Marc.Mulcahy, and Padraig.Obriain -- LGPL [1]
- [Crystal Clear](#) -- An icon set by [Everaldo](#) -- LGPL [1]
- [docutils](#) -- reStructuredText markup processing by David Goodger and contributors-- Public Domain [2]



- [expat](#) -- XML parsing library by the Thai Open Source Software Center Ltd, Clark Cooper, and contributors -- MIT License
- [fontconfig](#) -- Font configuration detection and support by Keith Packard -- MIT License
- [freetype](#) -- High quality text rendering library by Werner Lemberg, David Turner, and contributors -- FreeType License
- [glib](#) -- Object development support library by Hans Breuer, Matthias Clasen, Tor Lillqvist, Tim Janik, Havoc Pennington, Ron Steinke, Owen Taylor, Sebastian Wilhelmi, and contributors -- LGPL [1]
- [gtk+](#) -- Cross-platform GUI library by Jonathan Blandford, Hans Breuer, Matthias Clasen, Tim Janik, Tor Lillqvist, Federico Mena Quintero, Kristian Rietveld, Søren Sandmann, Manish Singh, Owen Taylor, and contributors.-- LGPL [1]
- [gtk-engines](#) -- GTK theme engines by The Rasterman, Owen Taylor, Randy Gordon -- LGPL [1]
- [gtkscintilla2](#) -- GTK wrapper for Scintilla by Dennis J Houy, Sven Herzberg, and contributors-- LGPL [1]
- [GTK Themes](#) -- Aluminum Alloy by Robert Iszaki (roberTO), AluminumAlloy License [4]; Black-Background based on work by Eric R. Reitz, unspecified [5]; Glider by Link Dupont, LGPL [1]; Glossy P by m5brane, unspecified [5]; gnububble by Kyle Davis, unspecified [5]; H2O by Eric R. Reitz, unspecified [5]; High Contrast, Low Contrast, and Large Print themes by Bill Haneman and T. Liebeck, LGPL [1]; Smokey-Blue by Jakub 'jimmac' Steiner and Paul Hendrick, LGPL [1]; Redmond and Redmond95 by Anonymous, unspecified [5]; Smooth2000 by ajgenius, unspecified [5]; SmoothDesert by Ken Joseph, other [6]; SmoothRetro by Ken Joseph, other [6]; SmoothSeaIce by ajgenius, unspecified [5]
- [gtk-wimp](#) -- GTK theme with Windows native look by Raymond Penners, Evan Martin, Owen Taylor, Arnaud Charlet, and Dom Lachowicz.-- LGPL [1]
- [libiconv](#) -- Unicode conversion library by Bruno Haible -- LGPL [1]
- [libpng](#) -- PNG image support library by Glenn Randers-Pehrson, Andreas Eric Dilger, Guy Eric Schalnat, and contributors -- zlib/libpng License
- [libXft](#) -- X windows font rendering by Keith Packard and contributors -- MIT License
- [libXrender](#) -- X windows rendering extension by Keith Packard and contributors -- MIT License

- [pango](#) -- Text layout and rendering library by Owen Taylor and contributors -- LGPL [1]
- [parsetools](#) -- Python parse tree conversion tools by John Ehresman -- MIT License
- [pexpect](#) -- Sub-process control library by Noah Spurrier, Richard Holden, Marco Molteni, Kimberley Burchett, Robert Stone, Hartmut Goebel, Chad Schroeder, Erick Tryzelaar, Dave Kirby, Ids vander Molen, George Todd, Noel Taylor, Nicolas D. Cesar, Alexander Gattin, Geoffrey Marshall, Francisco Lourenco, Glen Mabey, Karthik Gurusamy, and Fernando Perez -- MIT License
- [py2pdf](#) -- Python source to PDF output converter by Dinu Gherman -- MIT License
- [pygtk](#) -- Python bindings for GTK by James Henstridge and contributors -- LGPL [1]
- [pyscintilla2](#) -- Python bindings for gtkscintilla2 by Roberto Cavada and contributors -- LGPL [1]
- [pysqlite](#) -- Python bindings for sqlite by Gerhard Haering -- BSD-like custom license [7]
- [Python](#) -- The Python programming language by Guido van Rossum, PythonLabs, and contributors -- Python Software Foundation License version 2 [3]
- [render](#) -- Header files for X render extension by Keith Packard -- MIT License
- [scintilla](#) -- Source code editor component by Neil Hodgson and contributors -- MIT License
- [sqlite](#) -- A self-contained, serverless, zero-configuration, transactional SQL database engine -- Public domain [8]
- [Tulliana-1.0](#) -- An icon set by M. Umut Pulat, based on Nuvola created by David Vignoni -- LGPL [1]
- [zlib](#) -- Data compression library by Jean-loup Gailly and Mark Adler -- zlib/libpng License

## Notes

[1] The LGPL requires us to redistribute the source code for all libraries linked into Wing IDE. All of these modules are readily available on the internet. In some cases we may have modifications that have not yet been incorporated into the official versions; if you wish to obtain a copy of our version of the sources of any of these modules, please email us at [info at wingware.com](mailto:info@wingware.com).

[2] Docutils contains a few parts under other licenses (BSD, Python 2.1, Python 2.2, Python 2.3, and GPL). See the COPYING.txt file in the source distribution for details.

[3] The Python Software Foundation License version 2 is an OSI Approved Open Source license. It consists of a stack of licenses that also include other licenses that apply to older parts of the Python code base. All of these are included in the OSI Approved license: PSF License, BeOpen Python License, CNRI Python License, and CWI Python License. The intellectual property rights for Python are managed by the [Python Software Foundation](#).

[4] Not OSI Approved. Wingware has obtained explicit permission from the author to redistribute these themes.

[5] Not OSI Approved. These GTK themes are widely distributed works that are implicitly in the public domain, but without stated license or copyright. They may be removed from Wing IDE without altering the product's base functionality by removing the correspondingly named directories from bin/gtk-bin/share/themes within the Wing IDE installation.

[6] Not OSI Approved. However, license grants permission to modify and use without limitation.

[7] Not OSI Approved, but similar to other OSI approved licenses. The license grants anyone to use the software for any purpose, including commercial applications.

[8] The source code states the author has disclaimed copyright of the source code. The [sqlite.org](#) website states: "All of the deliverable code in SQLite has been dedicated to the public domain by the authors. All code authors, and representatives of the companies they work for, have signed affidavits dedicating their contributions to the public domain and originals of those signed affidavits are stored in a firesafe at the main offices of Hwaci. Anyone is free to copy, modify, publish, use, compile, sell, or distribute the original SQLite code, either in source code form or as a compiled binary, for any purpose, commercial or non-commercial, and by any means."

## Scintilla Copyright

We are required by the license terms for Scintilla to include the following copyright notice in this documentation:

```
Copyright 1998-2003 by Neil Hodgson <neilh@scintilla.org>
```

```
All Rights Reserved
```

```
Permission to use, copy, modify, and distribute this soft-  
ware and its
```

documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation.

NEIL HODGSON DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL NEIL HODGSON BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

### Fontconfig Copyright

We are required by the license terms for Fontconfig to include the following copyright notice in this documentation:

Copyright © 2001,2003 Keith Packard

Permission to use, copy, modify, distribute, and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Keith Packard not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. Keith Packard makes no representations about the suitability of this software for any purpose. It is provided "as is" without express or implied warranty.

KEITH PACKARD DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO

EVENT SHALL KEITH PACKARD BE LIABLE FOR ANY SPECIAL, INDIRECT OR  
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULT-  
ING FROM LOSS OF USE,  
DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER  
TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR  
PERFORMANCE OF THIS SOFTWARE.