Wing IDE 101 Reference Manual
Wing IDE 101

Wingware
www.wingware.com

Version 3.1.2
July 1, 2008

Contents

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

Introduction

Thanks for choosing Wingware's Wing IDE 101! 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 and debugger.

The rest of this chapter describes how to install and start using Wing IDE 101. 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. 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, and Vista. 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. 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**.

Only Python 2.2 and later are supported for Mac OS X. OS X 10.3 and later come with a standard version of Python already installed.

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.

Some contributed builds of Wing IDE for other operating systems may be available from time to time.

1.2. Supported Python versions

Wing supports CPython 2.0 through 2.5, Stackless Python 2.4 and 2.5, 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 compilation are supported on Windows and Linux. 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.

1.3. 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:

- Downloaded or CD version 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 Prerequisities for Mac OS X:

- An X11 window server, such as Apple X11 for OS X (available on the OS X install disks) or XDarwin.
- A window manager. Apple's X11 Server includes one; other options include Window Maker and OroborOSX

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

1.4. 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: On all platforms, the installation location for Wing IDE is referred to as `WINGHOME`.

Windows 2000 and XP

Install Wing IDE by running the downloaded executable. Wing's files are installed by default in `C:\Program Files\Wing IDE 101 3.1`, 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 3.1 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-1013.1` or at the selected location when installing from the tar file. Wing will also create a **User Settings Directory** in `~/.wing101-3`, which is used to store preferences and other settings.

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

Mac OS X 10.3+

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**.

1.5. 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-101.exe` (located inside `WINGHOME`).

On Linux/Unix, just execute `wing-101-3.1` (or `wing-101` located inside `WINGHOME`)

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

1.6. 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 -- `~/.wing101-3` (a sub-directory of your home directory)

Windows -- In `Wing 101 3` 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 101 3
```

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

```
c:\Users\${username}\AppData\Roaming\Wing 101 3
```

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**.

1.7. 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**

1.7.1. Fixing a Failed Upgrade

In rare cases when upgrading within minor versions (for example 3.0 to 3.0.3), the upgrade 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 wingide3.1`. Then go into `/usr/lib/wingide3.1` and remove any remaining files and directories.

Linux Debian

If you installed Wing IDE for Linux from Debian package, issue the command `dpkg -r wingide3.1`. Then go into `/usr/lib/wingide3.1` 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.

1.8. Installation Details and Options

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

1.8.1. 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-1013.1`.

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-1013.1-3.1.2-1.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-1013.1` directory and the `wing-1013.1` 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. To install, run `dpkg -i wingide-1013.1-3.1.2-1.i386.deb` as root or use your favorite package administration tool to install. Most files for Wing are placed under the `/usr/lib/wingide-1013.1` directory and the `wing-1013.1` command is placed in the `/usr/bin` directory.

On 64-bit systems, you can install from the Debian package but need to first install the `ia32-libs` 32 bit compatibility libraries and then install Wing with the command `dpkg -i --force-architecture wingide-1013.1-3.1.2-1.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.

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-1013.1`. Unpacking this archive with `tar -zxvf wingide-101-3.1.2-1-i386-linux.tar.gz` will create a `wingide-101-3.1.2-1-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-1013.1`. These locations default to `/usr/local/lib/wingide-101` 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.

1.8.2. Installing Extra Documentation

If you are using Linux/Unix, the Python manual is not included in most installations, so you may also wish to download and install local copies of these pages.

Place the top-level of the HTML formatted Python manual (where `index.html` is found) into `python-manual/#.#` within your Wing IDE installation. Substitute for `#.#` the major and minor version of the corresponding Python interpreter (for example, for the Python 2.3.x manual, use `python-manual/2.3`).

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.

1.9. Removing Wing IDE

Windows

On Windows, use the Add/Remove Programs control panel, select `Wing IDE 101 3` and remove it.

Linux/Unix

To remove an RPM installation on Linux, type `rpm -e wingide-1013.1`.

To remove an Debian package installation on Linux, type `dpkg -r wingide-1013.1`.

To remove a tar archive installation on Linux/Unix, invoke the `wing-uninstall` script in `WINGHOME`. 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.

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 Vim, Emacs, Visual Studio, and Brief emulation
- The action of the tab key can be configured
- Many other options are available through preferences

2.1. Editor Personalities

The default editor 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, or Brief user will want to do is to set the editor personality to emulate their editor of choice. This is done with the **Keyboard / Personality** user interface 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 editor 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 **Key Equivalents** for details.

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 style of windows used when working with the IDE, the layout of tools within windows, display text font, size, and color, the style and content of the toolbar, and the overall look or "theme" (including white on black and many others).

2.2.1. 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, but this can be altered in **Preferences**) 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 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.

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

2.2.2. 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.

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.

2.2.3. 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**.

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.

3.1. 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.

3.2. 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.

3.3. 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.

Moving to other files can also be done with the `Window` menu, which lists all open files.

3.4. 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.

3.5. Transient vs. non-Transient Editors

Wing can open files in two modes:

Transient Mode -- Files opened when debugging or navigating to point of definition are opened in transient mode and will be automatically closed when hidden.

Non-Transient Mode -- Files opened normally from the File menu, from the keyboard file selector, or by double clicking on items in the Project tool will be opened in non-transient mode, and kept open until they are explicitly closed.

A file can be switched between transient and non-transient mode 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 recently visited files (blue items were transient, black items non-transient).

Transient files that are edited are also automatically converted to non-transient mode.

3.6. Indentation

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

3.6.1. 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. Wing then matches new indentation as the file is edited to the form already found in the file. If mixed forms of indentation are found, the most common form is used. If no indentation is found, space-only indents are inserted using the size set in preferences.

3.6.2. Indentation Preferences

The following preferences affect how the indentation features behave:

1) The **Default Indent Size** preference defines the default size of each level of indent, in spaces. This is used in new empty files only. 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.

2) 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.

3.6.3. 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.

3.6.4. The Tab Key

By default, the tab key either indents according to context or increases the indent depth at the current cursor position by one level (this depends on the selected editor **Personality**). If one or more lines are selected, this instead operates on the indentation of all selected lines by one level.

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.

3.6.5. 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.

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.

3.7. 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.

3.8. 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.

3.9. Notes on Copy/Paste

There are a number of ways to 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.

- In emacs mode, ctrl-k (`kill-line`) will cut one line at a time into the private emacs clipboard. This is kept separate from the system-wide clipboard and is pasted using ctrl-y (`yank-line`). On Windows and Mac OS X, ctrl-y will paste the contents of the system-wide clipboard only if the emacs clipboard is empty.
- In VI mode, named text registers are supported.
- 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

It is important to note which actions use the system-wide clipboard, which use the emacs private clipboard or VI registers, and which use the X windows selection (X Windows only). Otherwise, these commands are interchangeable in their effects.

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.

4.1. 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.

4.2. 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.
- **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.

4.3. 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

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.

5.1. 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
- `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.

6.1. 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).

6.2. 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.

- Use one of the key bindings given in the `Debug` menu.

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**.

6.3. 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:

- **Gray** -- There is no debug process.
- **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.

6.4. 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

6.5. 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.

6.6. 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.

6.6.1. 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.

6.6.1.1. 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.

6.6.2. 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.

6.7. 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`.

6.8. 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.

Source Code Analysis

Wing's auto-completer, source index menu, goto-definition capability, some of the source reformatting features, and in Wing IDE Professional the source code browser and source assistant 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.

7.1. 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 completely re-analyze your source code from scratch.

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

7.2. Static Analysis Limitations

The following are known limitations affecting features based on 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. In these cases, you may use `isinstance` and/or interface files to inform the analyzer, as described below.
- 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.
- Some newer Python language constructs and possible type inferencing cases are not explicitly supported.

7.3. Helping Wing Analyze Code

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

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. An example is `assert isinstance(obj, CMyClass)`. 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 such assertions go a long way. The more Wing knows about your code, the faster you will be able to edit and navigate it.

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).

To inform the code analysis facility of the contents of an extension module, it is possible to create a `*.pi` (Python interface) file. For example, for a module imported as `mymodule`, the interface file is called `mymodule.pi`. This file is simply a Python skeleton with the appropriate structure and call signature to match the functions, attributes, classes, and methods defined in an extension module. In many cases, these files can be auto-generated from interface files.

Wing will search for `*.pi` files first in the same directory as it finds the extension module (or its 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 subdirectory `#.#` 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 2.3 is being used, Wing will check first in `c:\share\pi\pi-files`, then in `c:\share\pi\pi-files\2.3`. then in `c:\share\pi\pi-files\2.2`, 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.

7.4. Analysis Disk Cache

The source code analyzer writes information about files it has recently examined into the Cache Directory that is specified in the About box accessible 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 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 with no ill effects.

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.

8.1. 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-101-3.1` 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.

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.

8.2. 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 by removed via the Windows control panel. The demo extensions may be disabled by using ShellExView (http://www.snapfiles.com/get/shellexview.html) or a similiar 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 occuring.

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.

8.3. Speeding up Wing

Wing should present a responsive, snappy user interface even on relatively slow hardware. In some cases, Wing may appear sluggish:

1) 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.

2) If you have nVidia desktop manager, disable it for Wing.

3) 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 limit this effect in subsequent sessions, Wing stores its source analysis information to disk in a cache within your **User Settings Directory**.

However, with large projects even reading this cache and checking files for updates may take a while when Wing is first started. This process happens in the background after launch and takes 7-15 seconds per 100,000 lines of code on a Celeron 400 processor and should be almost unnoticable on any modern hardware.

In all cases, Wing will eventually complete this process and should at that time consume almost no CPU during normal editing and debugging.

4) 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.

5) 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.

6) Some users have reported Hummingbird Socks Client for Windows to cause the debugger to slow down substantially, apparently as a result of improperly routed TCP/IP packets.

7) 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.

8.4. 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.

8.5. 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.

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.

9.1. 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 101" 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 concurrent 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:

1a. 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.

1b. NON-COMMERCIAL USE OF SOFTWARE PRODUCT

"Non-Commercial Use" means the use of the Software Product for non-commercial purposes only, and is limitated to the following users: (a) non-profit organizations (charities and other organizations created for the promotion of social welfare), (b) universities, colleges, and other educational institutions (including, but not limited to elementary schools, middle schools, high schools, and community colleges), (c) independent contractors who are under contract by the above-stated organizations and using the Software Product exclusively for such non-profit or educational clients, and (d) other individual users who use the Software Product for personal, non-commercial use only (for example, hobby, learning, or entertainment).

Under no circumstances can the Software Product be used by or for a for-profit organization, or be used to generate income for personal benefit.

Wingware, a Delaware corporation, reserves the right to further clarify the terms of Non-Commercial Use at its sole determination.

2. GRANT OF NON-EXCLUSIVE LICENSE

Wingware grants the non-exclusive, non-transferable right for a single user to use this Software Product for Non-Commercial Use on a single operating system per software license purchased. Each additional concurrent user of the Software Product, and each additional operating system where the product is used, requires an additional Software Product License.

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 each Software Product License on a single computer system. A second installation of the same Software Product License may be made on one other computer system, so long as both copies of the same Software Product License never come into concurrent use. You may also make copies of the Software Product License as necessary for backup and/or archival purposes. Backup and archival copies may not come into active use, together with the Software Product, for any purpose. No other copies may be made. Each copy must reproduce all copyright and other proprietary rights notices on or in the Software Product License. You may not modify or create derivative copies of the Software Product License.

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 without Wingware's written approval, and subject to written agreement by the recipient of the terms of this EULA.

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 redigés 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 dancing bird logo, the feather logo, Wing IDE, Wing IDE 101, Wing IDE Personal, Wing IDE Professional, Wing IDE Enterprise, Wing Debugger, and "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.

9.2. 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]
- 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]
- Crystal Clear -- An icon set by Everaldo -- LGPL [1]
- Tulliana-1.0 -- An icon set by M. Umut Pulat, based on Nuvola created by David Vignoni -- 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]
- python -- The Python programming language by Guido van Rossum, PythonLabs, and contributors -- Python 2.3 License [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
- 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.

[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 2.3 license is an OSI Approved Open Source license. Each version of Python is under a similar but unique license; Wing includes only Python 2.3.

[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 implicitely 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.

Scintilla Copyright

We are required by the license terms for Scintilla to include the following copyright notice in this documentation:

Fontconfig Copyright

We are required by the license terms for Fontconfig to include the following copyright notice in this documentation: