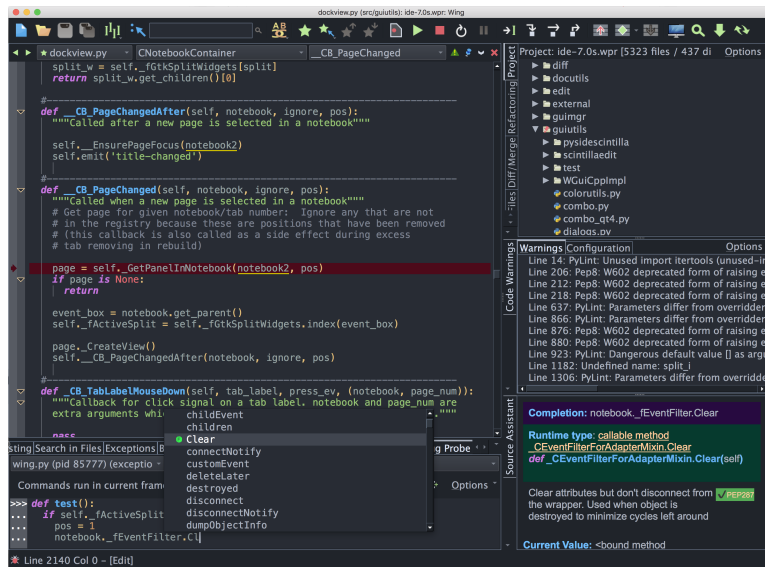




Wing Personal Quick Start Guide



This is a minimalist guide for getting started quickly with Wing Personal. For a more in-depth introduction, try the [Tutorial](#).

Wing Personal is a simple Python IDE designed for students and hobbyists or those using Python occasionally for fairly simple development tasks. If you are working intensively with Python, you should try Wing Pro instead. You will be much more productive, and chances are you will have no problem understanding and appreciating Wing Pro's more advanced features.

Let's get started with Wing Personal!

Install Python

If you don't already have Python on your system, install it now. Two good options are:

- Obtain the standard Python distribution from python.org
- Use [Anaconda](#) for seamless access to many third party Python libraries. See [Anaconda package lists](#) for a list of the available libraries.

See [Supported Python Versions](#) for other options.

You may need to restart Wing after installing Python, so that it can recognize the new installation.

Set up a Project

After Wing is running, create a new project from the **Project** menu. If you want to create a new virtualenv to use with your project, select the **Create New Virtualenv** project type, enter the name and parent directory for the new virtualenv, and select the base Python interpreter to use.

Alternatively, if you don't want to create a new virtualenv, select **Empty Python Project** project type. Then configure your project with the following steps:

1. Use **Add Existing Directory** in the **Project** menu to add your sources to the project. It's best to constrain this to the directories you are actively working with and let Wing find the libraries you use through the **Python Path**.
2. Use **Project Properties** in the **Project** menu to set **Python Executable** to the **python.exe** or other interpreter executable you want to use with your project. If Python is not on the **PATH**, set this to the full path that is in **sys.executable** in the desired Python installation.
3. If your code alters **sys.path** or loads modules in a non-standard way then you may need to set **Python Path** in **Project Properties** so that Wing can find your modules for auto-completion, refactoring, debugging, testing, and other features.
4. You may want to right-click on your main entry point in the **Project** tool and select **Set As Main Entry Point** so that debugging always starts there.
5. Use **Save Project As** in the **Project** menu to save your project to disk.

See [Project-Wide Properties](#) and [Per-File Properties](#) for a description of all available properties.

Notice that Wing also offers other project types in the **New Project** dialog, including a project type for each of the frameworks, tools, and libraries listed in [How-Tos](#).

Wing may consume significant CPU time when it first analyzes your code base. Progress is indicated in the lower left of the IDE window. Once this is done, the results are cached across sessions and Wing should run with a snappy and responsive interface. See [Source Code Analysis](#) to learn how Wing's source analysis system works.

Basic Configuration

You are now ready to start working with code, but may want to make a few configuration changes first:

Display Colors - The **User Interface > Color Palette** preference chooses the colors used in the editor or the whole user interface if the **User Interface > Use Color Palette Throughout the UI** preference is enabled.

Key Bindings - Wing can emulate VI/Vim, Visual Studio, Emacs, Eclipse, XCode, MATLAB, and Brief editors, as selected from **Keyboard Personality** in the **Edit** menu or with the **User Interface > Keyboard > Personality** preference.

Tab Key - The default tab key action depends on the selected keyboard personality and in some cases file type, context, and whether or not there is a selection in the editor. This can be changed from the **User Interface > Keyboard > Tab Key Action** preference.

Completion Keys - By default, the auto-completer uses the **Tab** key for completion, but other keys can be added using the **Editor > Auto-completion > Completion Keys** preference.

There are many other options in **Preferences**.

Navigating Code

Wing Personal provides a number of different ways to navigate the structure of your code, and several methods for quickly finding symbols or files by name:

Source Index menus at the top of the editor provide quick access to other parts of a source file.

Goto-definition is available from the **Source** menu, and by right-clicking on symbols in the editor and , **Python Shell**. Use the forward/back history buttons at the top left of the editor to return from the point of definition.

Find Symbol in the **Source** menu in Wing Pro and Wing Personal jumps to a symbol defined in the current file when you type a fragment of its name.

Open From Project in the **File** menu in Wing Pro and Wing Personal provides a similar interface for quickly opening project files.

See [Navigating Source](#) for details on the above.

Source Browser in the **Tools** menu in Wing Pro and Wing Personal provides module or class oriented display of the structure of your code. [Details](#)

Source Assistant in the **Tools** menu shows detailed information about symbols selected in the editor, auto-completer, **Source Browser**, **Python Shell**, **Project**, and other tools. [Details](#)

Searching

Wing Personal provides several different interfaces for searching your code. Which you use depends on what you want to search and how you prefer to interact with the search and replace functionality:

Toolbar search is a quick way to search the current file. [Details](#)

Search in the **Tools** menu shows the **Search** tool, which provides incremental text, wildcard, and regular expression search and replace in selections and the current file or documentation page. [Details](#)

Mini-search in Wing Pro and Wing Personal provides powerful keyboard-driven search and replace. The key bindings listed in the **Mini-search** area of the **Edit** menu display the search entry area at the bottom of the window. [Details](#)

Search in Files in the **Tools** menu in Wing Pro and Wing Personal shows the **Search in Files** tool, which provides wildcard and regular expression search and replace in filtered sets of files, directories, named file sets, and within the project and documentation. [Details](#).

Editing Code

Wing Personal's editor is designed to speed up the process of writing and modifying Python code, and to reduce the incidence of coding errors. Its features include:

Auto-completion in Wing's editor and , **Python Shell** speeds up typing and reduces coding errors. The auto-completer uses **Tab** by default for completion, but this can be changed in the **Editor > Auto-completion > Completion Keys** preference. This feature is disabled by default in Wing 101. [Details](#)

Auto-indent in Wing Pro and Wing Personal matches the file's existing indentation. When multiple lines are pasted, they are re-indented according to context. A single **Undo** reverts an unwanted indentation change. A selected range of code may be re-indented as a block using **Indentation** in the **Source** menu or the indentation toolbar group. The **Indentation** tool may be used to convert a whole file's indentation style. [Details](#)

Multiple Selections can be made with **Multiple Selections** in the **Edit** menu, the multiple selections toolbar item, and by pressing **Ctrl+Alt** (or **Command+Option** on OS X) while making a selection with the mouse. Once multiple selections have been made, edits made will be applied to all the selections at once. [Details](#)

Quick Selection operations in the **Edit > Select** menu allow selecting whole statements, blocks, or scopes before copying, editing, or searching through them. [Details](#)

Debugging Code

Wing's debugger is a powerful tool for finding and fixing bugs, understanding unfamiliar code, and writing new code interactively. You can launch code from the **Debug** menu or toolbar, [from the Python Shell](#), or from outside of the IDE either [on the same machine](#) or [on another host](#).

[Breakpoints](#) can be set by clicking on the breakpoint margin to the left of the editor. [Stepping operations](#) are in the **Debug** menu and toolbar.

The **Stack Data** tool is used to inspect or change program data. Hovering the mouse over a [symbol in the editor](#) shows the value for that symbol in a tooltip, if available on the active debug stack.

Debug process I/O is shown in the **Debug I/O** tool, or [optionally in an external console](#).

Other debugger features include:

Launch Configurations in the **Project** menu in Wing Pro and Wing Personal define different runtime environments for debugging, executing, and unit testing your code. [Details](#).

Named Entry Points in the **Debug** menu in Wing Pro and Wing Personal provide a way to launch the same file with different debug environments. [Details](#)

Other Features

Wing Personal includes a number of other features designed to make Python coding easier and more productive:

Python Shell -- Wing's **Python Shell** lets you try out code in an independent sandbox process. To enable debugging, click the bug icon in the top right of the **Python Shell**. In Wing Pro and Wing Personal, the shell provides auto-completion, goto-definition, and is integrated with the **Source Assistant**. [Details](#)

OS Commands in the **Tools** menu in Wing Pro and Wing Personal's displays the **OS Commands** tool, which execute external tools for build, code generation, and other purposes. [Details](#).

Preferences in the **Edit** menu (or **WingPersonal** menu on OS X) gives you control of the overall layout and color of the IDE, among many other options. Right click on tool and editor tabs for layout options, or drag tabs to move them or create new splits. Right-click on the toolbar to configure which tools are visible or to add your own. See [Customization](#) for details.

Perspectives in Wing Pro and Wing Personal let you save named tool panel layouts. [Details](#).

Other Features like [bookmarks](#), [code folding](#), [keyboard macros](#) are also available, and you can [extend Wing by writing Python scripts](#).

Further Reading

As you work with Wing Personal on your own software development projects, the following resources may be useful:

- [Wing Support Website](#) which includes a Q&A support forum, mailing lists, documentation, links to social media, and other information for Wing users.
- [Wing Reference Manual](#) which documents all the features in detail.
- [How-Tos](#) with instructions for using Wing with third party frameworks, applications, and tool, like Django, Jupyter, matplotlib, Autodesk Maya, Raspberry Pi, pygame, and many others.
- Wing displays a useful tip at startup, to help you continue learning about the feature set over time. These are also accessible from **Wing Tips** in the **Help** menu.

Thanks for using Wing Personal!