

Learning Python

Getting results for beamlines and scientific programming

Intermediate python: GUI programming with WxPython, part 1

Outline of topics to be covered

1. Background on wxPython; other GUI choices
2. wxPython documentation
3. Concepts used in GUI building, sample widgets
4. Event-driven code
5. Getting started with wxPython





BACKGROUND ON WXPYTHON

Some history of WxPython

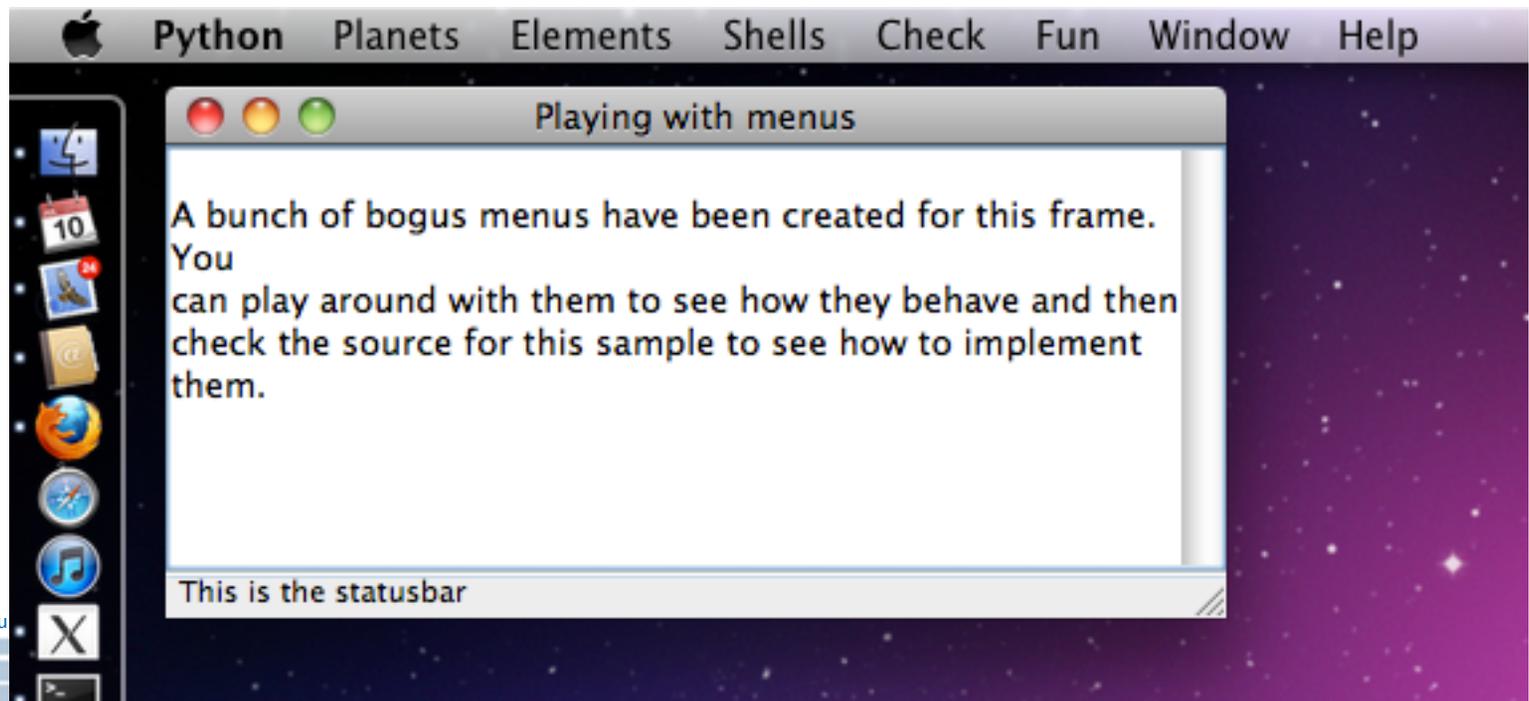
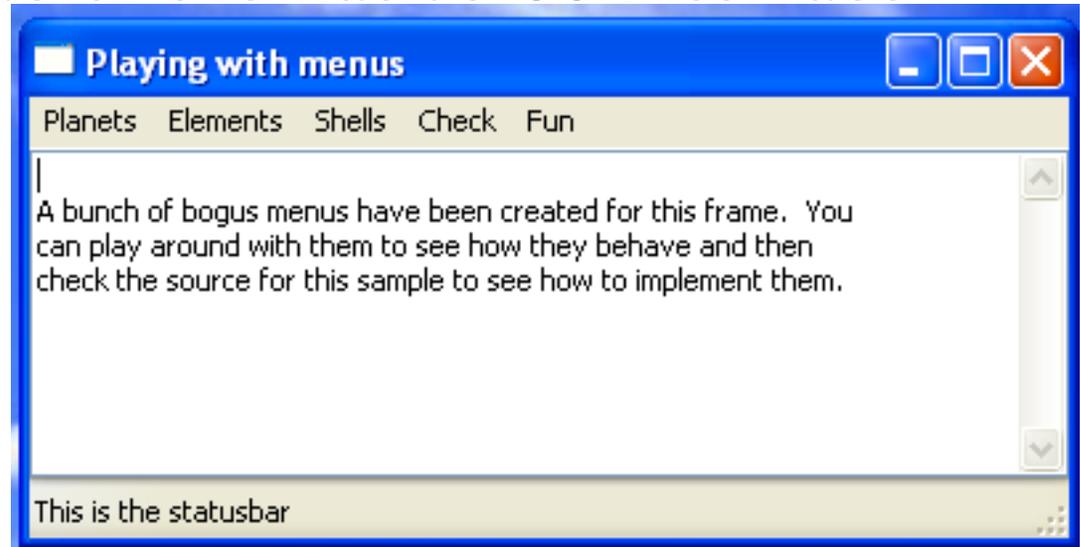
- Every modern operating system has at least one code library designed to creating Graphical User Interface (GUI) programs.
 - Windows provides something called Microsoft Foundation Classes (MFC)
- WxWindows was a C++ GUI development library modeled after the Microsoft Foundations Classes GUI library, except that WxWindows was intended to be cross-platform so one code could run on many platforms.
 - The name was changed to WxWidgets when lawyers from some company threatened to sue over the use of “windows” in the name.
- A Python binding was developed for WxWidgets; that package is known as WxPython.

The goal of WxWidgets & WxPython is to provide a single set of routines for developing GUI-based programs that have the “look and feel” of the native platform



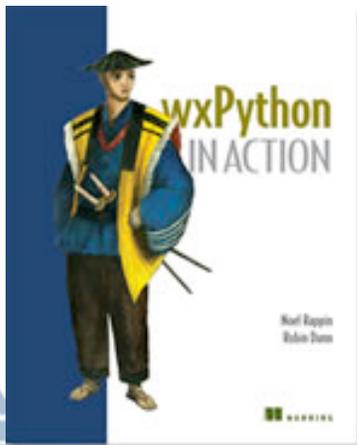
wxPython follows style of the native OS interface

wxPython has the expected platform look & feel: On the Mac, the menu bar is integrated into the system at the usual place, not inside the application Frame.



Getting more information on wxPython

- Visit the wxPython class description <http://www.wxpython.org/docs/api/>
 - note that usually one must look a parents of a class to find all the defined methods (functions for a widget).
- wxPython demos: wxPython/demo/demo.py (to be discussed later)
- The wxPython wiki: <http://wiki.wxpython.org/>
- Google discussion group: <http://groups.google.com/group/wxpython-users/topics>



- Book: ***wxPython in Action*** by Noel Rappin and Robin Dunn (March, 2006)

Other GUI choices for Python: TkInter

TkInter is a Python implementation of the Tk package, originally written for the Tcl/Tk scripting language. It is implemented in Python through the TkInter package which adds a wrapped Tcl/Tk interpreter.

Advantages:

- Included as part of the base Python distribution
- GUI's look exactly the same on all platforms
- Pretty simple

Disadvantages:

- Somewhat clumsy, since an extra layer of code is needed to access Tcl variables and data structures
- GUI's look exactly the same on all platforms (poor OS “look and feel”)
- Relatively primitive



Other GUI choices for Python: Qt

Qt (pronounced as “cute” officially, but many people say “que-tee”) is a GUI framework run owned by a subsidiary of Nokia. It is available both open source and with commercial support. It is used in many commercial products as well as many open source projects.

There are two different Python bindings for Qt: PyQt and PySide

- PyQt is older and more widely used
- PySide is supported by Nokia and is distributed as part of the Enthought Python Distribution. PySide appears to be similar to PyQt except that deprecated Qt routines are not supported and PySide only supports the “API 2” approach used in PyQt.





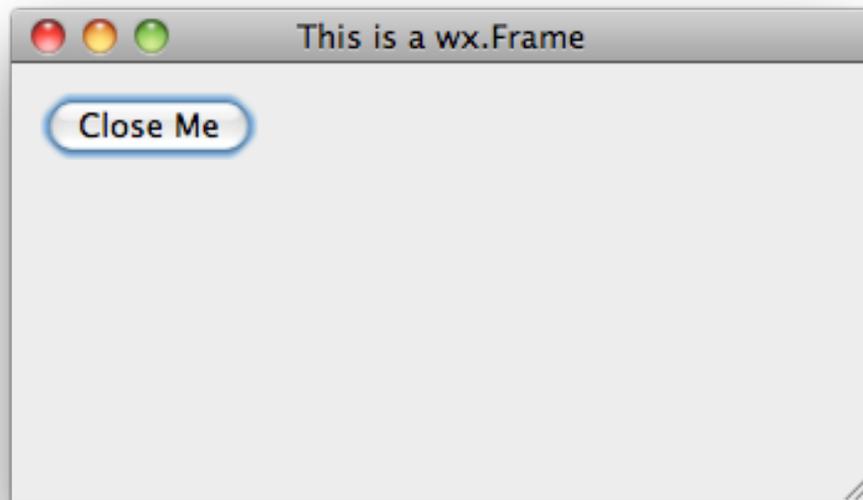
CONCEPTS IN GUI DEVELOPMENT



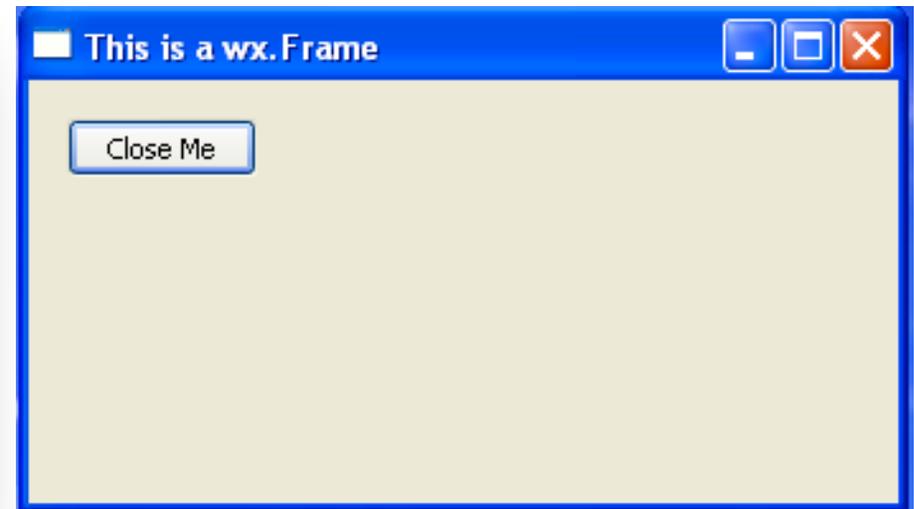
Components of a GUI: The window

A GUI-based program usually consists of one or more windows, such as the one below

- to make life interesting, wx calls windows “Frames”
- The appearance of the Window is controlled by the OS/Window manager; it usually contains a title and has system controls.
- The Frame is populated with widgets such as buttons, labels, and devices for input
 - to make life a bit too interesting, wx calls widgets “windows”



Mac



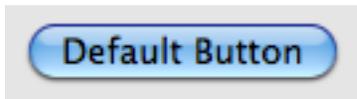
Window



Other common GUI components

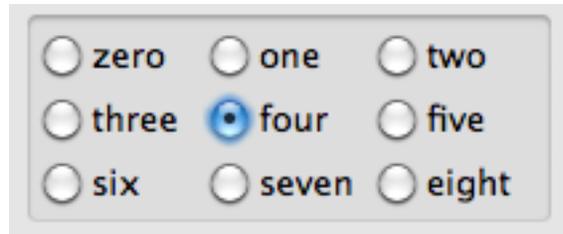
Some commonly used GUI components (widgets):

Button



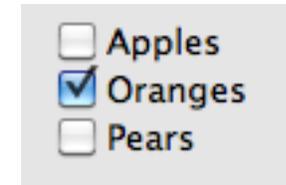
Radio Button

(exclusive selection)

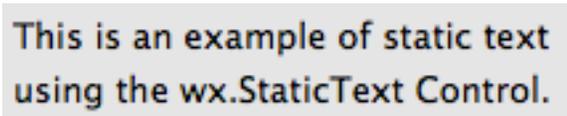


Check button

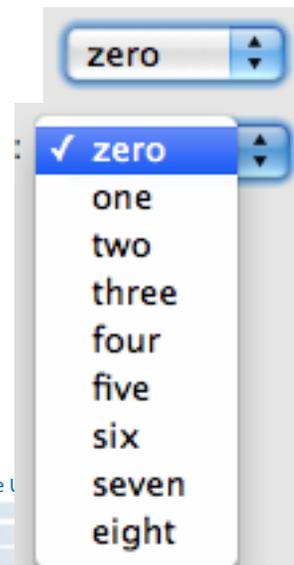
(multiple selection)



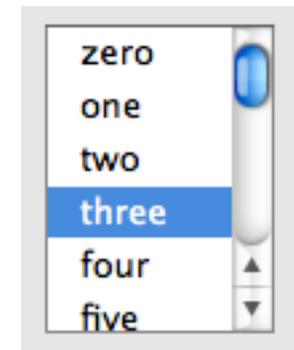
Label



Choice control



List Box

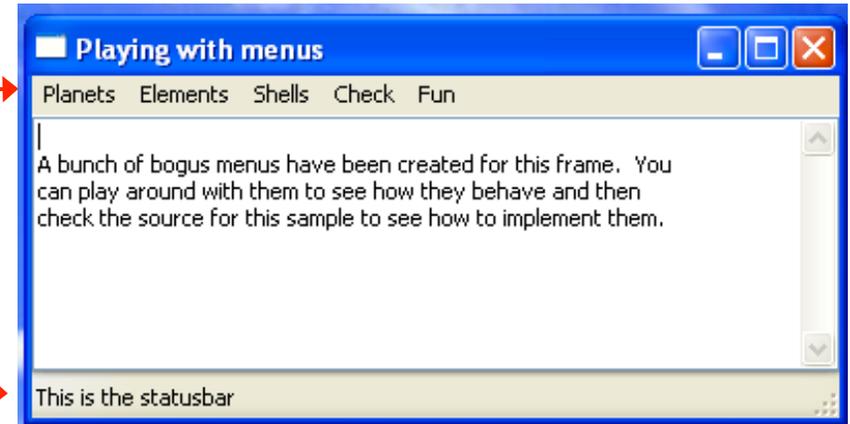


Text Entry



Components in typical GUI applications

- More sophisticated GUIs have a menu and often a status bar.



- Menus can be multi-level

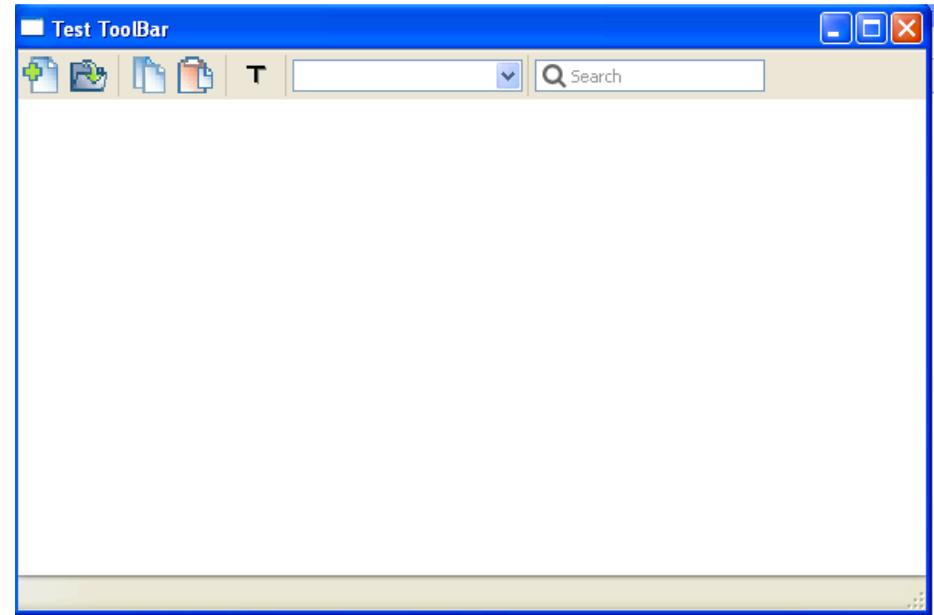


(from the Menu demo in Core Windows/Controls section of wxPython demos).



Other nice components...

- Toolbars are good ways to provide quick access to commonly-needed menu items. (*Core Windows/ToolBar demo*)

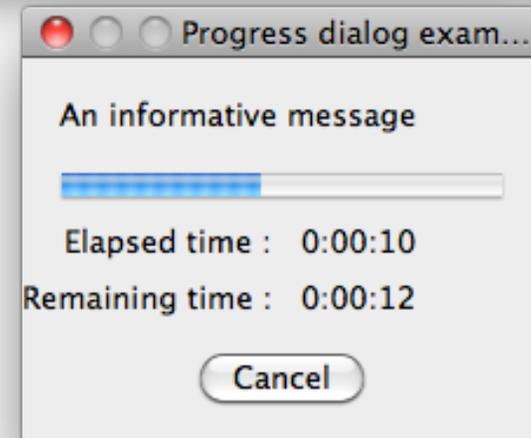
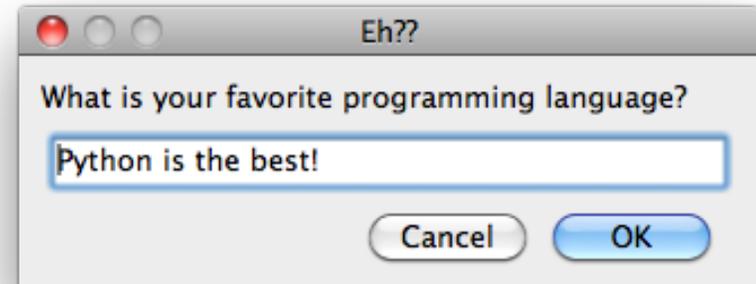
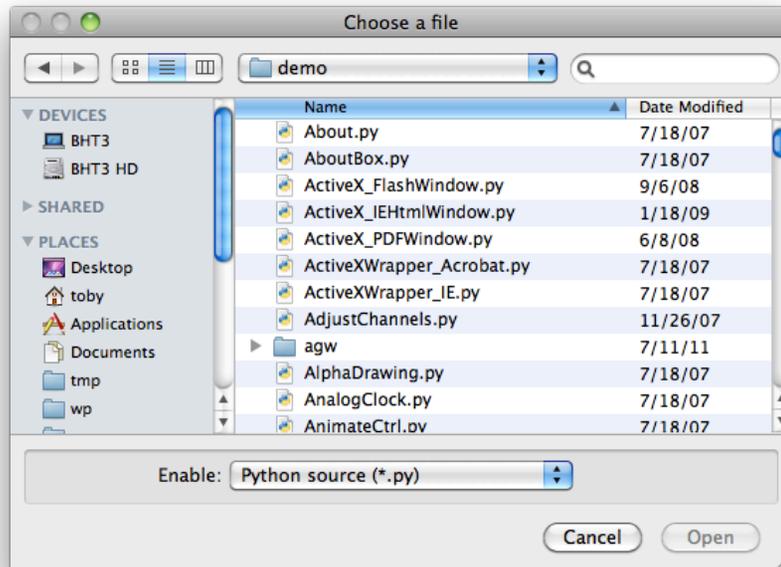


- About boxes are a good place to provide information on the application. (*Common dialogs/AboutBox demo*)



Common Dialogs

Many things need to be done repeatedly in GUI program where simple (or sometimes complex) windows are created to display some information or get an item of input. Optimally, these dialogs follow the style of the OS.



Differences between writing traditional code and GUI code

Most (non-GUI) programs have a linear structure. There is a set of routines for the beginning, middle and end. These routines are run in a specific sequence (as possibly dictated by the program input).

GUI programs are event-driven, which means most of the actions of the program are performed in response to “actions” such as a window being resized or exposed, user clicking a mouse button, the expiration of a timer,...

- A GUI program has a beginning, with code that will be run before control is handed off to the user.
- Then the program goes into a sort of infinite loop, where the program waits for events to occur and then calls routines to handle the “actions”. This is called the event loop.
- Thus a GUI program has two sections: program initialization and event handlers
- End routines are a special case of an event handler

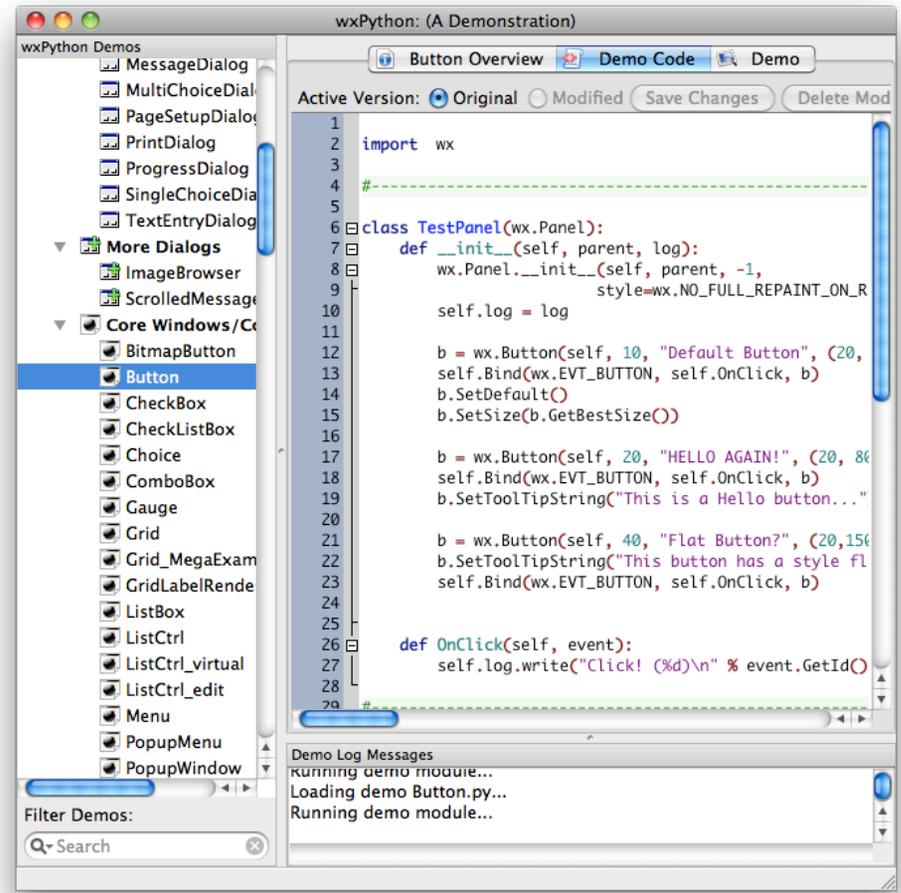
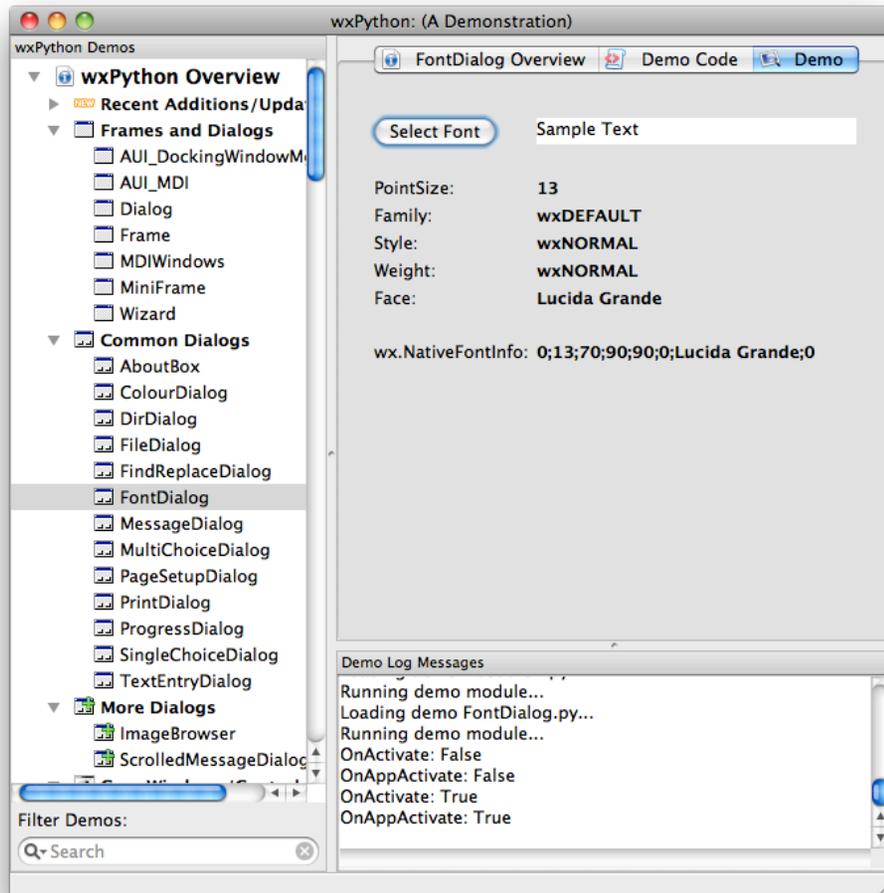




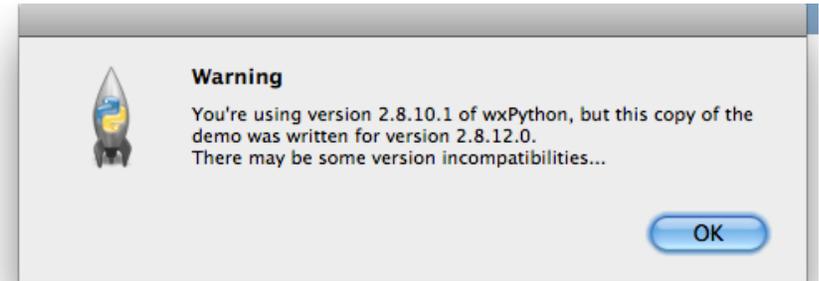
LEARNING ABOUT WXPYTHON

How do I find out what components wxPython has?

wxPython offers a hundreds of options for widgets. These can be viewed, along with example code using the wxPython/demo/demo.py demo application.



Installing the wxPython/demo



- Installing the wxPython demo files is strongly recommended for developers using wx. Alas, these files are not distributed in EPD. See <http://www.wxpython.org/download.php> to download the latest version of the demos. They are packaged by platform, but the source files for Linux will work on any platform.
- You should download a version that is close to what you have (the 2.9 demos will not work in wx 2.8).
- If you download a version that does not exactly match the installed wx version, you will get an annoying, but usually inconsequential error message (above).
- Best is to find out what version of wx is in use (see above). Then download the demo file from the matching directory in <http://sourceforge.net/projects/wxpython/files/wxPython/> for example, <http://sourceforge.net/projects/wxpython/files/wxPython/2.8.10.1/wxPython-demo-2.8.10.1.tar.bz2>

```
bht3:work toby$ python
Enthought Python Distribution
Version: 7.0-2 (32-bit)
>>> import wx
>>> wx.__version__
'2.8.10.1'
```



Important Terminology used in wxPython

- Application – this refers to the program. wxPython requires a complex extra layer of code that takes care of interactions with the OS – fortunately this is very simple to invoke.
- Frame – this is the wxPython name for a window
- Modal vs. Non-Modal windows
 - A modal window is one that “locks up” the application until it is closed, such as what happens in most programs when you use “Save as”.
 - A non-modal window is one that behaves independently of other windows open in the same program (e.g. browser windows or document windows in Word.)
- Panel – a panel is an invisible widget that holds other objects. While not always strictly necessary, always use at least one.
- Sizer – a way to collect and arrange widgets

