wxPython in a Nutshell

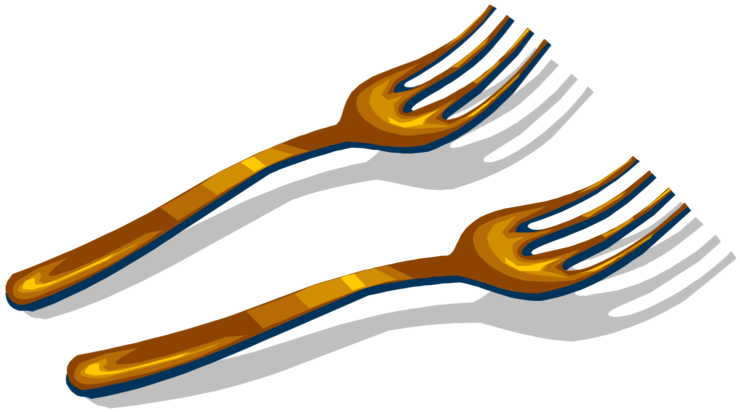
Robin Dunn

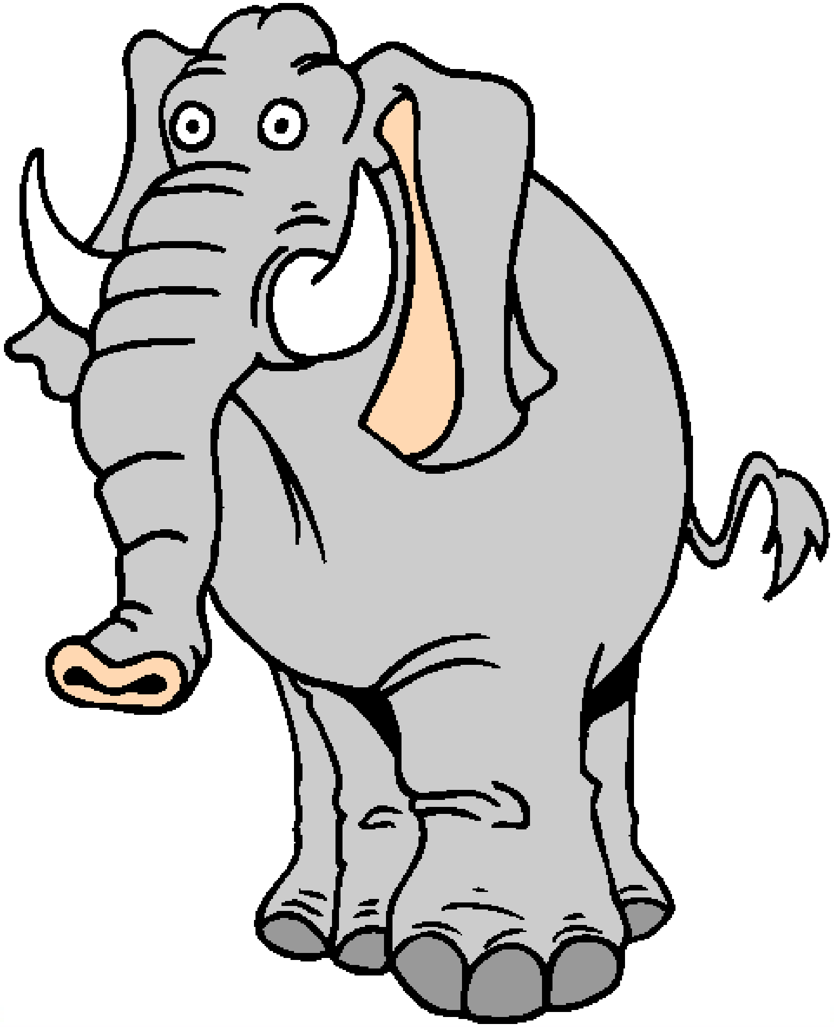
http://wxPython.org/

O’Reilly Open Source Convention

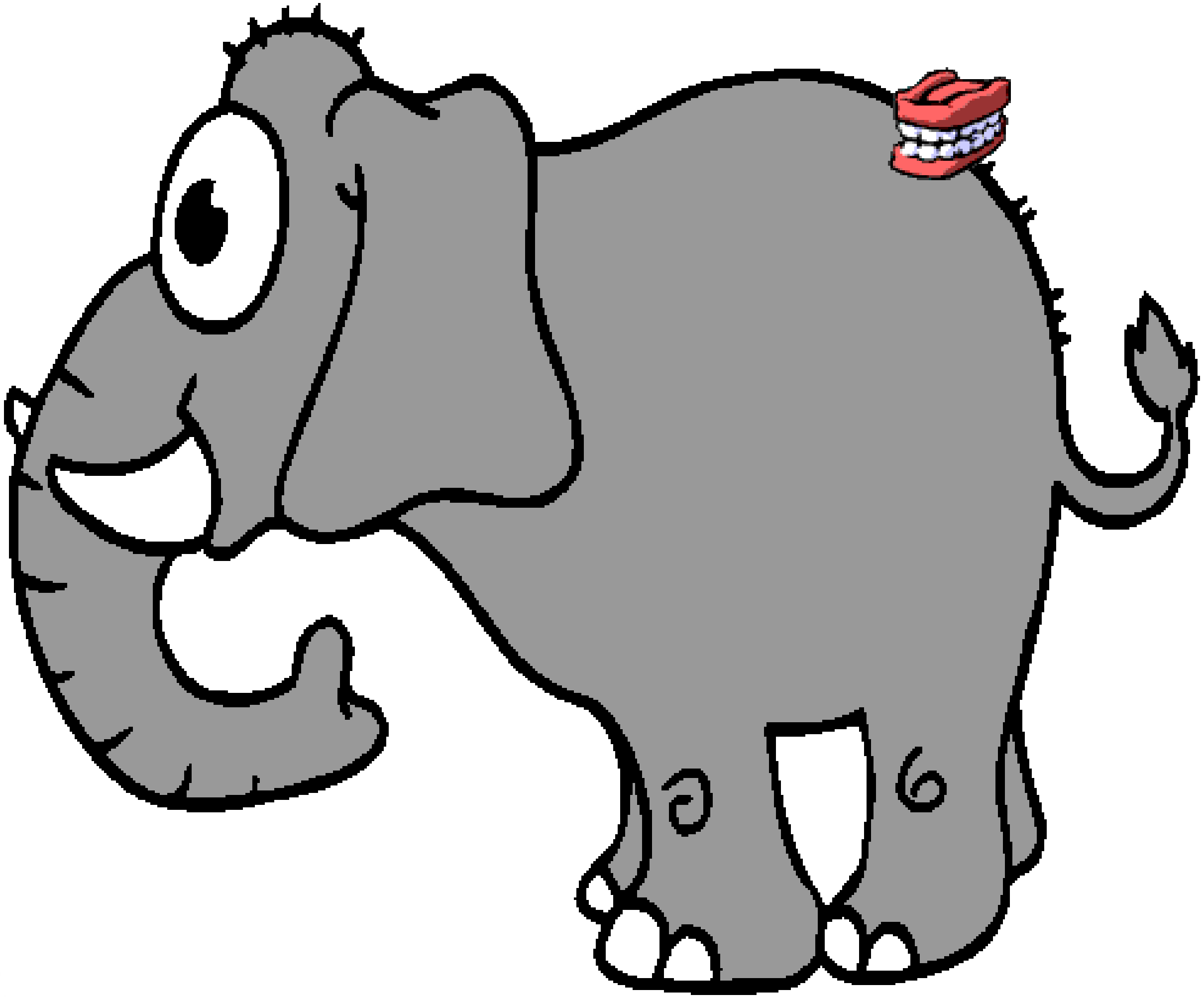
July 26–30, 2004

The best way to eat an elephant…





…is one bite at a time



Why wxPython?

- wxPython is an **open source** GUI toolkit based on the wxWidgets (formerly wxWindows) library
- Designed to be cross-platform and supports most Unix/Linux platforms, MS Windows and Mac OS X
- Uses **native widgets** wherever possible to preserve native Look and Feel.
- Extensive sample programs, helpful and capable community
- Mature, well established projects.

– wxWidgets:	1992
– wxPython:	1996

Getting started with wxPython

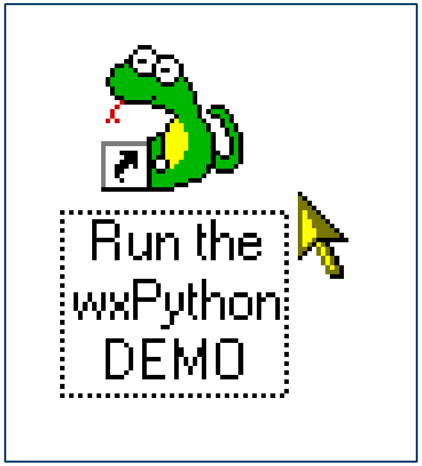
- Choose an installer
  - http://wxPython.org/downloads.php
  - Windows *.exe installers, Linux RPMs or OSX *.dmg
  - Can be built from source with a few prerequisites
- Which version of Python do you use?
  - 2.3, 2.4, 2.5
- Unicode or ANSI?
  - Unicode builds available on all platforms, but be careful with Win9x/ME
  - ANSI available for platforms, but may be phased out soon.

Getting started with wxPython

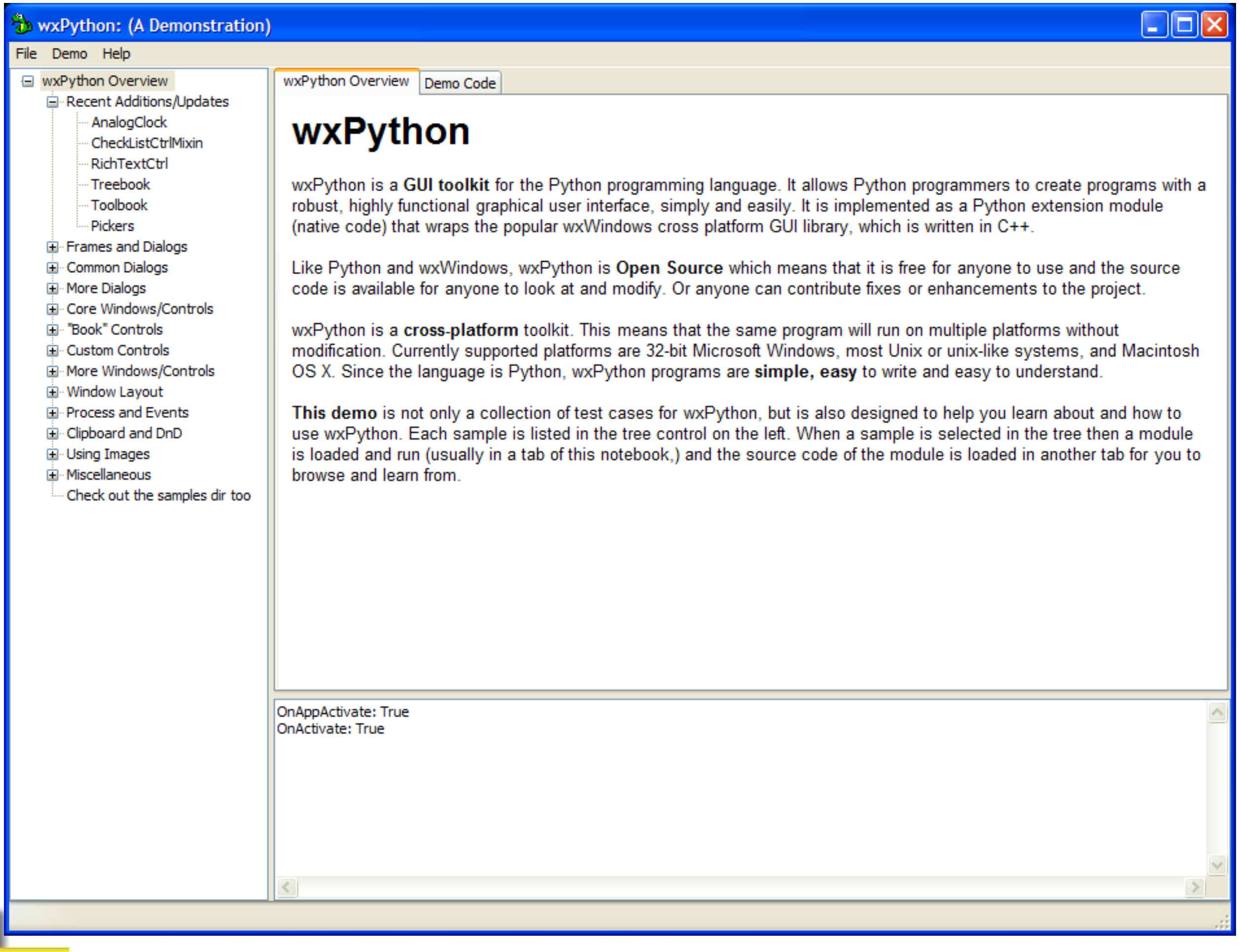
- Choose an editor or development environment:
  - Boa Constructor
  - WingIDE
  - SPE
  - SCiTE
  - Emacs, vi, etc.
- It’s just plain text, so any ordinary editor and command line will do.

Getting started with wxPython

- Ready, set, go!
- The wxPython Demo is a great way to learn about the capabilities of the toolkit.



Getting started with wxPython



wxPython

wxPython is a **GUI toolkit** for the Python programming language. It allows Python programmers to create programs with a robust, highly functional graphical user interface, simply and easily. It is implemented as a Python extension module (native code) that wraps the popular wxWindows cross platform GUI library, which is written in C++.

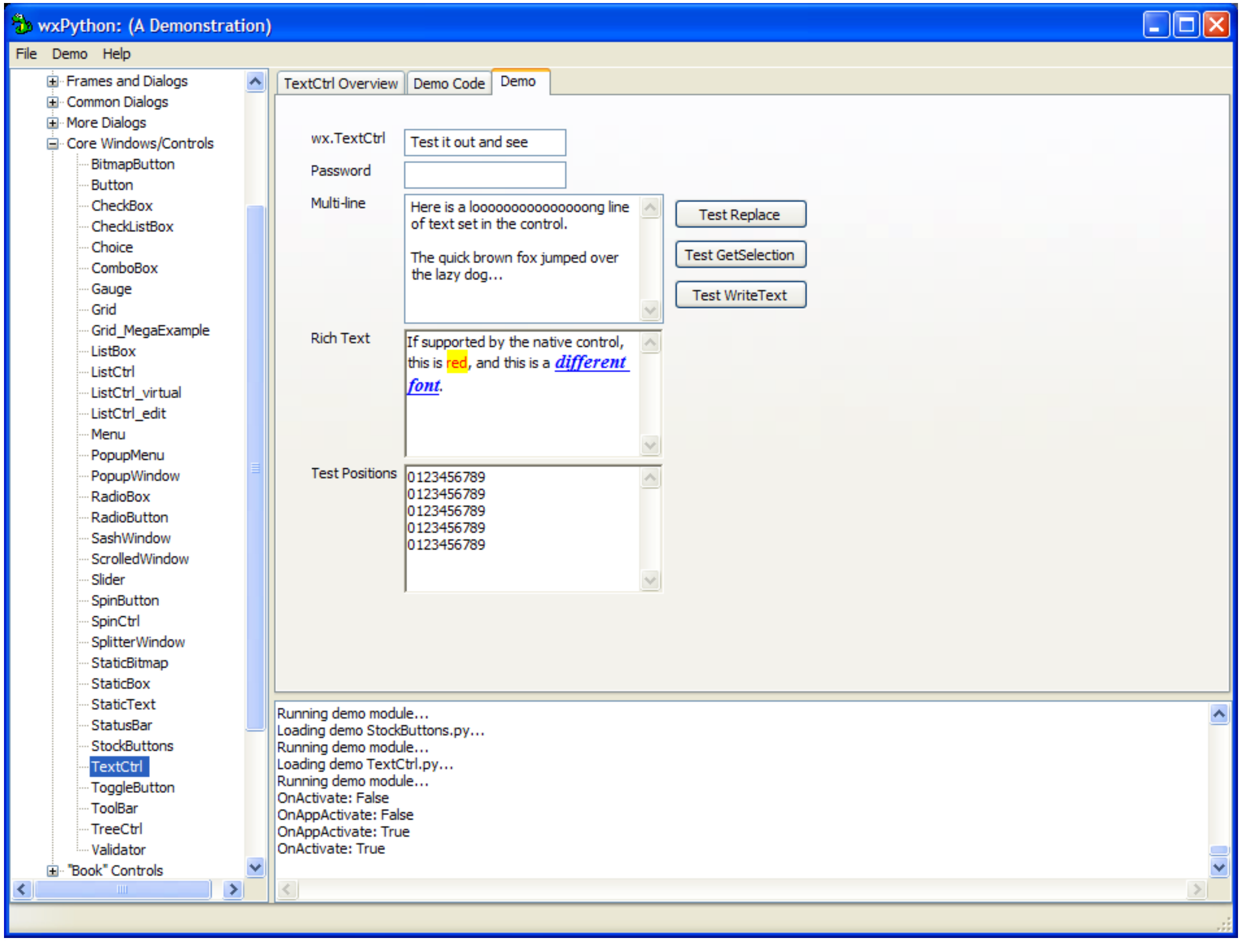
Like Python and wxWindows, wxPython is **Open Source** which means that it is free for anyone to use and the source code is available for anyone to look at and modify. Or anyone can contribute fixes or enhancements to the project.

wxPython is a **cross-platform** toolkit. This means that the same program will run on multiple platforms without modification. Currently supported platforms are 32-bit Microsoft Windows, most Unix or unix-like systems, and Macintosh OS X. Since the language is Python, wxPython programs are **simple, easy** to write and easy to understand.

This demo is not only a collection of test cases for wxPython, but is also designed to help you learn about and how to use wxPython. Each sample is listed in the tree control on the left. When a sample is selected in the tree then a module is loaded and run (usually in a tab of this notebook,) and the source code of the module is loaded in another tab for you to browse and learn from.

```
OnAppActivate: True
OnActivate: True
```

Getting started with wxPython



wx.TextCtrl: Test it out and see

Password:

Multi-line: Here is a looooooooooooooong line of text set in the control.

The quick brown fox jumped over the lazy dog...

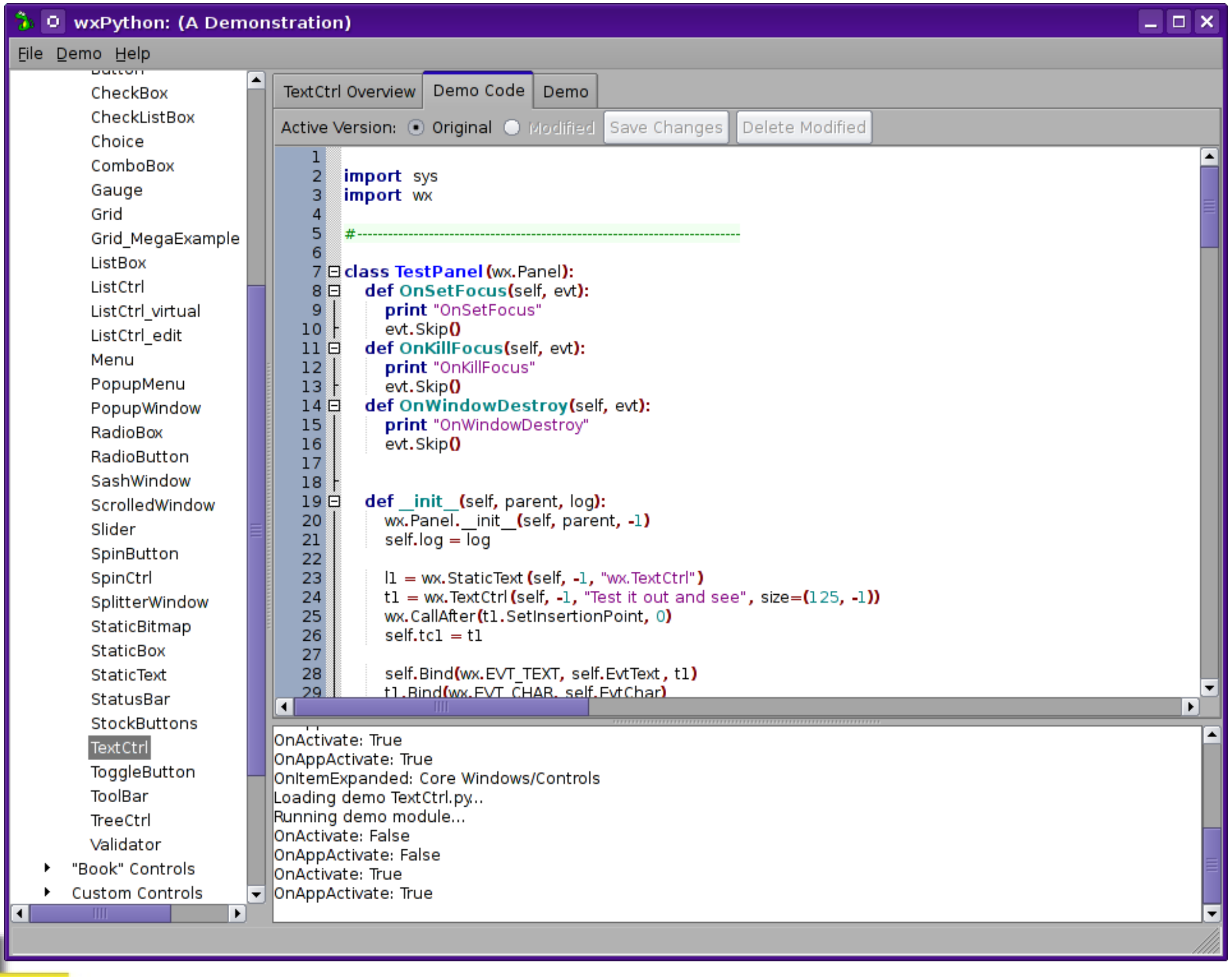
Rich Text: If supported by the native control, this is red, and this is a ***different font***.

Test Positions:

```
0123456789
0123456789
0123456789
0123456789
0123456789
```

```
Running demo module...
Loading demo StockButtons.py...
Running demo module...
Loading demo TextCtrl.py...
Running demo module...
OnActivate: False
OnAppActivate: False
OnAppActivate: True
OnActivate: True
```

Getting started with wxPython

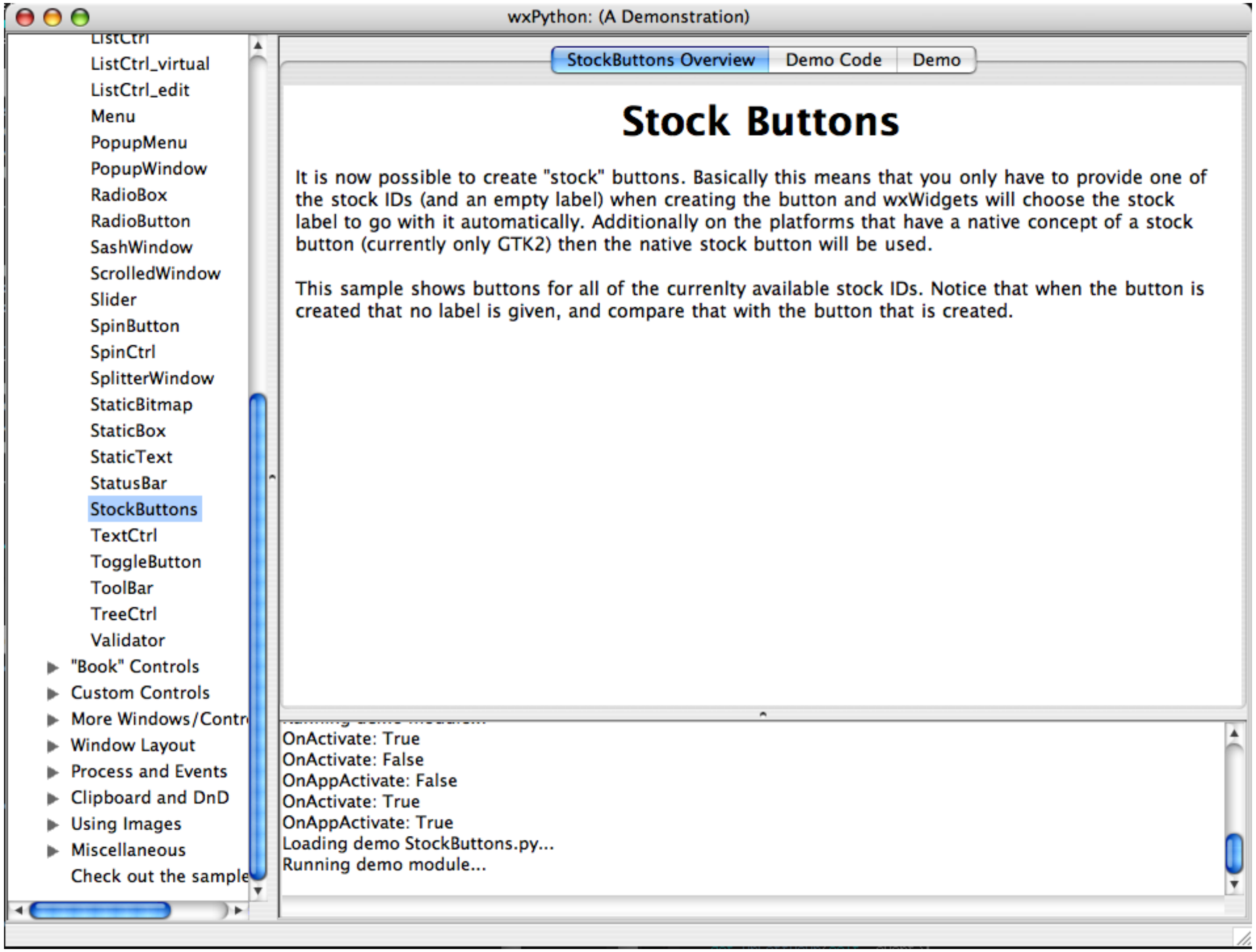


```python

import sys
import wx

#---------------------------------------------------------------------------

class TestPanel(wx.Panel):
    def OnSetFocus(self, evt):
        print "OnSetFocus"
        evt.Skip()
    def OnKillFocus(self, evt):
        print "OnKillFocus"
        evt.Skip()
    def OnWindowDestroy(self, evt):
        print "OnWindowDestroy"
        evt.Skip()


    def __init__(self, parent, log):
        wx.Panel.__init__(self, parent, -1)
        self.log = log

        l1 = wx.StaticText(self, -1, "wx.TextCtrl")
        t1 = wx.TextCtrl(self, -1, "Test it out and see", size=(125, -1))
        wx.CallAfter(t1.SetInsertionPoint, 0)
        self.tc1 = t1

        self.Bind(wx.EVT_TEXT, self.EvtText, t1)
        t1.Bind(wx.EVT_CHAR, self.EvtChar)
```

```
OnActivate: True
OnAppActivate: True
OnItemExpanded: Core Windows/Controls
Loading demo TextCtrl.py...
Running demo module...
OnActivate: False
OnAppActivate: False
OnActivate: True
OnAppActivate: True
```

wxPython: Cross Platform GUI Toolkit

Getting started with wxPython



Stock Buttons

It is now possible to create "stock" buttons. Basically this means that you only have to provide one of the stock IDs (and an empty label) when creating the button and wxWidgets will choose the stock label to go with it automatically. Additionally on the platforms that have a native concept of a stock button (currently only GTK2) then the native stock button will be used.

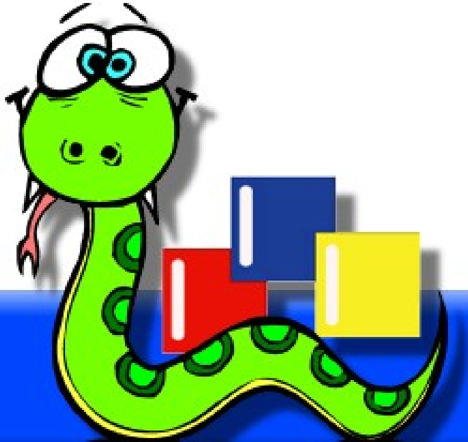
This sample shows buttons for all of the currenlty available stock IDs. Notice that when the button is created that no label is given, and compare that with the button that is created.

```
OnActivate: True
OnActivate: False
OnAppActivate: False
OnActivate: True
OnAppActivate: True
Loading demo StockButtons.py...
Running demo module...
```

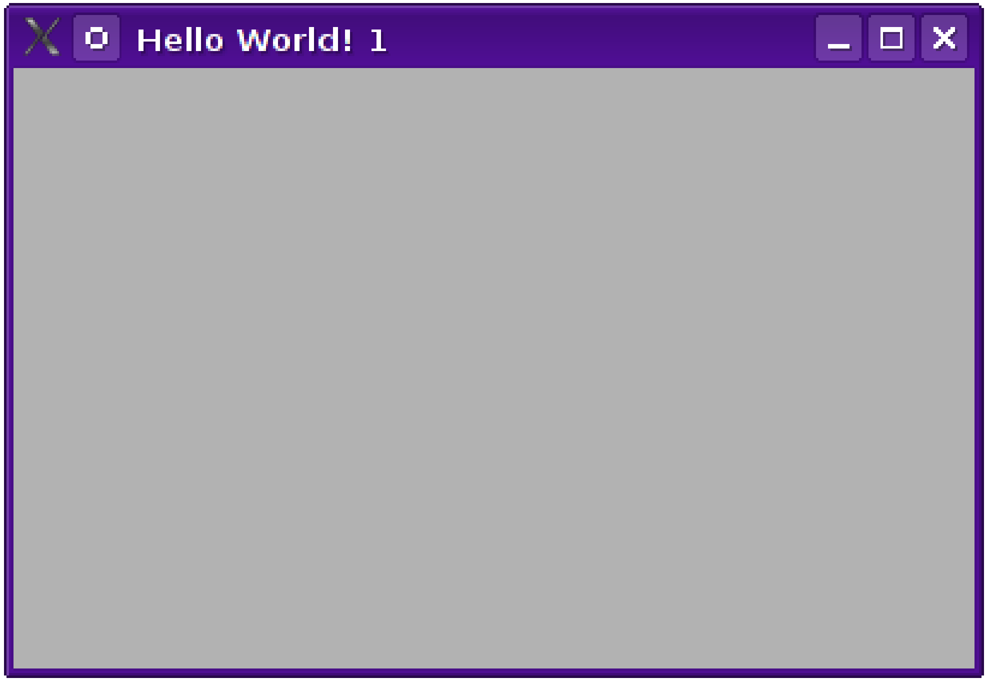
Demo time!

Hello World!

```
# ex01.py
import wx

class App(wx.App):
    def OnInit(self):
        frame = wx.Frame(parent=None, title="Hello World! 1")
        frame.Show()
        return True

app = App()
app.MainLoop()
```



Hello World!



wxPython Fundamentals

- Every application needs an instance of the wx.App class
  - Some parts of the C++ library are not initialized until the app is created, so it must be done before most other things.
  - APIs for starting and stopping the application
  - Provides the central *event loop* and dispatches events to handlers
  - Other per-application functionality
- Traditionally, you subclass wx.App and override OnInit for creating the initial application widgets
  - Not strictly needed any longer
  - wx.App can be used without subclassing
  - But it often still makes sense for design purposes

Hello World!

```
# ex02.py
import wx

app = wx.App()
frame = wx.Frame(parent=None, title="Hello World! 2")
frame.Show()
app.MainLoop()
```

wxPython Fundamentals

- wx.App can redirect standard output
  - Sends print statements and writes to sys.stdout or sys.stderr to a window or a file
  - An easy way to view status messages or tracebacks
  - Controlled by parameters to wx.App.__init__

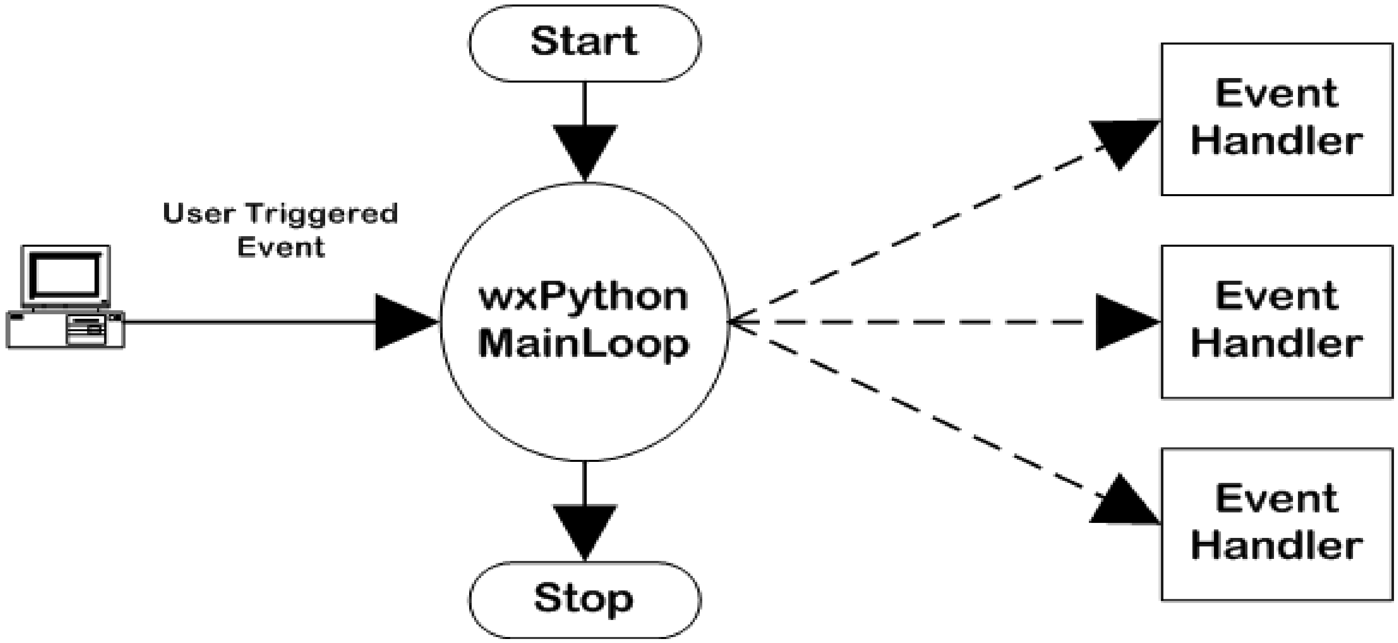
Hello World!

```
# ex03.py
import wx

class Frame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, parent=None, title="Hello World! 3")
        b1 = wx.Button(self, label="Hello", pos=(20,20))
        b2 = wx.Button(self, label="World", pos=(20,60))
        self.Bind(wx.EVT_BUTTON, self.OnHelloWorld)

    def OnHelloWorld(self, evt):
        print "Hello World!"

app = wx.App(redirect=True)
Frame().Show()
app.MainLoop()
```

Hello World!

```
# ex03.py
import wx

class Frame(wx.Frame):
    def __init__(self):
        wx.Frame.__init__(self, parent=None, title="Hello World! 3")
        b1 = wx.Button(self, label="Hello", pos=(20,20))
        b2 = wx.Button(self, label="World", pos=(20,60))
        self.Bind(wx.EVT_BUTTON, self.OnHelloWorld)

    def OnHelloWorld(self, evt):
        print "Hello World!"

app = wx.App(redirect=True)
Frame().Show()
app.MainLoop()
```

Event handling



Simple sample

```
import wx

class MyFrame(wx.Frame):
def __init__(self, parent, title):
        wx.Frame.__init__(self, parent, -1, title,
                          pos=(150, 150), size=(350, 200))

        menuBar = wx.MenuBar()
        menu = wx.Menu()
        menu.Append(wx.ID_EXIT, "E&xit\tAlt-X",
                   "Exit this simple sample")

        self.Bind(wx.EVT_MENU, self.OnTimeToClose,
                  id=wx.ID_EXIT)

        menuBar.Append(menu, "&File")
        self.SetMenuBar(menuBar)
        self.CreateStatusBar()
```

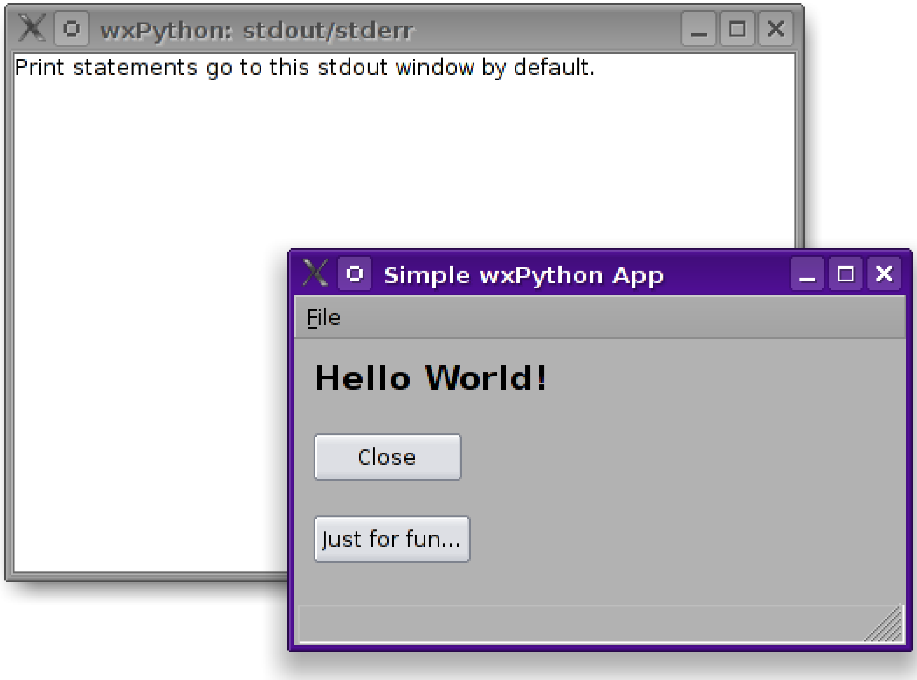
Simple sample

```
panel = wx.Panel(self)

text = wx.StaticText(panel, -1, "Hello World!")
text.SetFont(wx.Font(14, wx.SWISS, wx.NORMAL, wx.BOLD))

btn = wx.Button(panel, -1, "Close")
funbtn = wx.Button(panel, -1, "Just for fun...")

self.Bind(wx.EVT_BUTTON, self.OnTimeToClose, btn)
self.Bind(wx.EVT_BUTTON, self.OnFunButton, funbtn)

sizer = wx.BoxSizer(wx.VERTICAL)
sizer.Add(text, 0, wx.ALL, 10)
sizer.Add(btn, 0, wx.ALL, 10)
sizer.Add(funbtn, 0, wx.ALL, 10)
panel.SetSizer(sizer)
panel.Layout()
```

Simple sample

```
    def OnTimeToClose(self, evt):
        self.Close()

    def OnFunButton(self, evt):
        print "Having fun yet?"


class MyApp(wx.App):
    def OnInit(self):
        frame = MyFrame(None, "Simple wxPython App")
        frame.Show(True)
        self.SetTopWindow(frame)
        print "Print statements go to this stdout window by
   default."
        return True

app = MyApp(True)
app.MainLoop()
```

Simple sample



Print statements go to this stdout window by default.

Hello World!

More information

• wxPython website:	http://wxPython.org
• wxPyWiki:	http://wiki.wxPython.org
• Mailists:	wxPython-users, wx-users
• wxWidgets website:	http://wxWidgets.org
• *wxPython in Action*