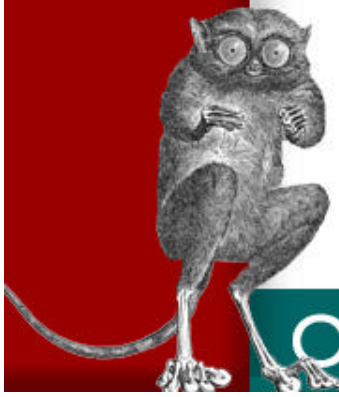
Up & Running With wxPython

Robin Dunn
Patrick O'Brien

O'Reilly Open Source Convention
July 7 - 11, 2003

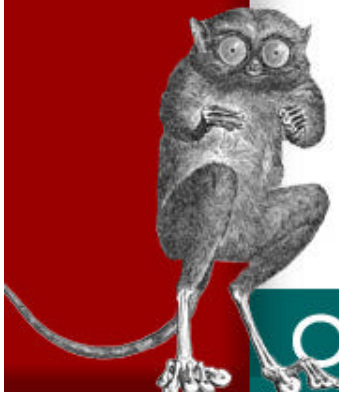
Presentation overview

- Introduction to wxPython
- Getting started
- Application fundamentals
- Widgets galore
- Event handling
- Organizing your layout
- Drawing
- Drag and drop
- Debugging with PyCrust
- Other tools

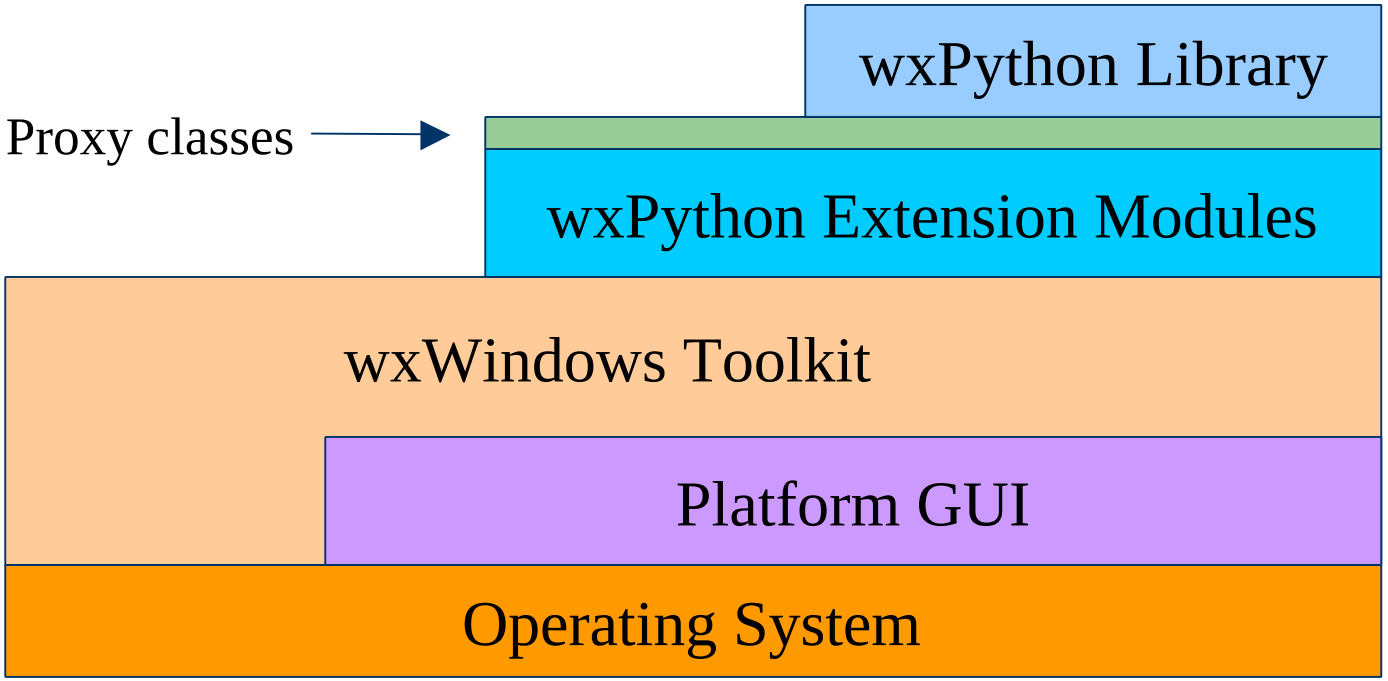


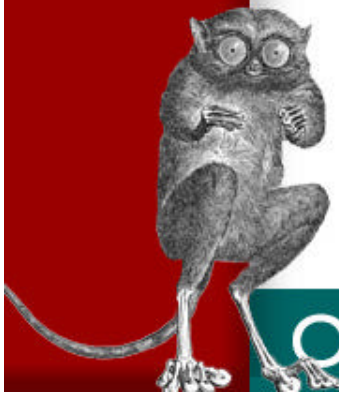
Introduction to wxPython

- wxPython is a GUI toolkit for Python, built upon the wxWindows C++ toolkit.
  - Cross platform: Windows, Linux, Unix, OS X.
  - Uses native widgets/controls, plus many platform independent widgets.
- Mature, well established projects.
  - wxWindows: 1992
  - wxPython: 1996



Introduction: architecture





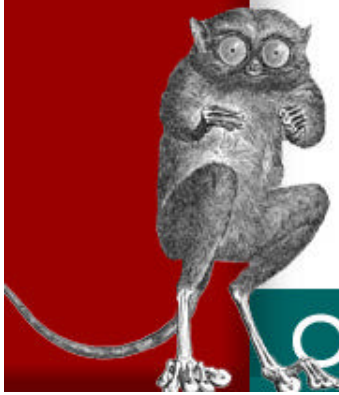
Introduction: new namespace

wxPython is transitioning from the old style…

```
from wxPython.wx import *
class MyFrame(wxFrame):
        . . .
```

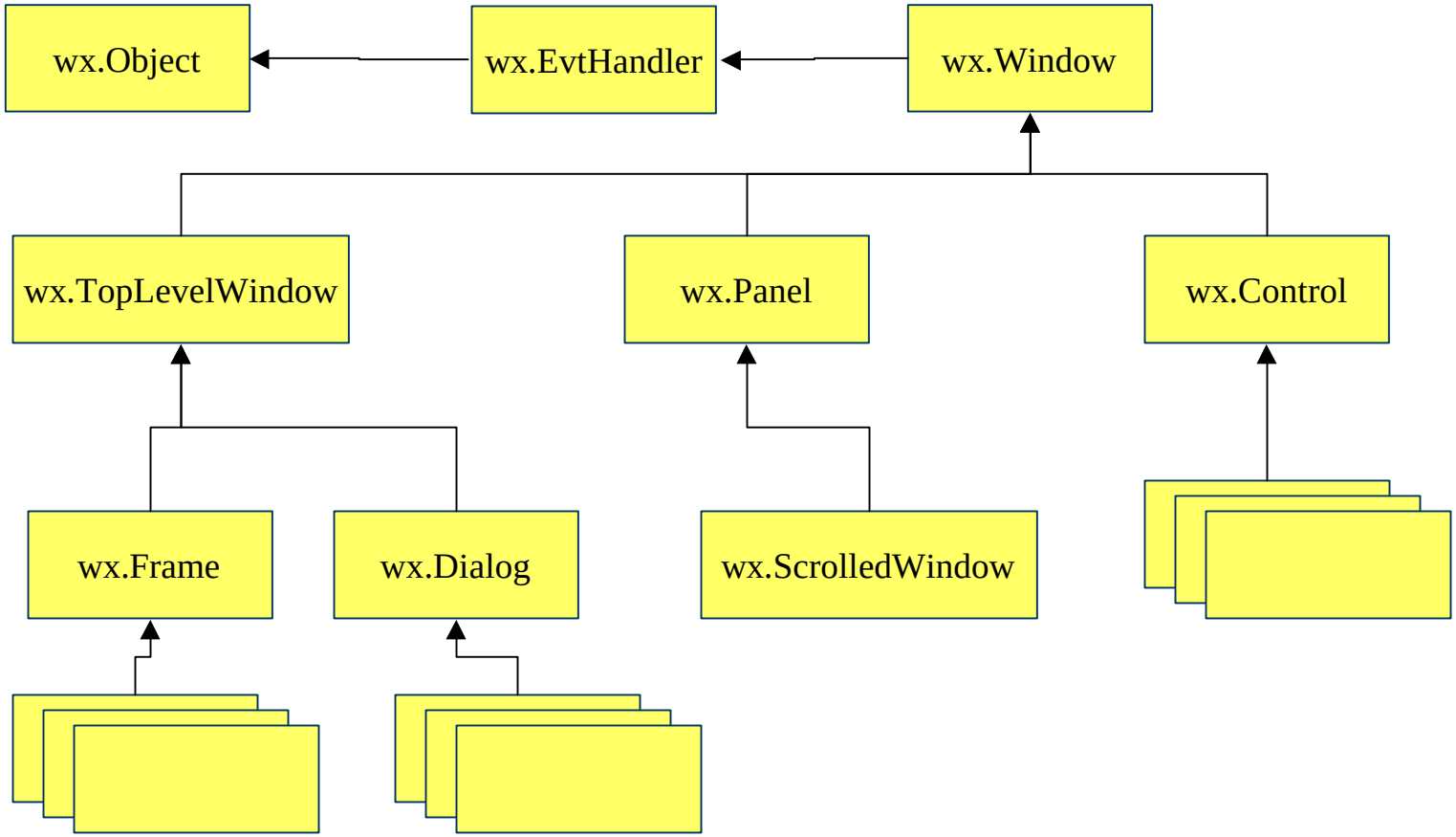
…to the new

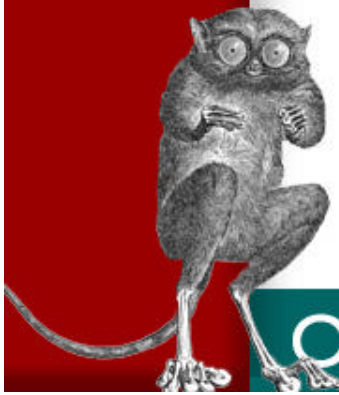
```
import wx
class MyFrame(wx.Frame):
        . . .
```



This tutorial will use the new style.

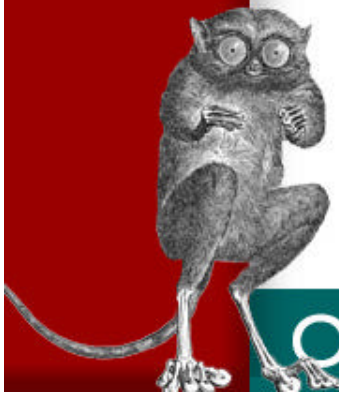
Introduction: partial class hierarchy





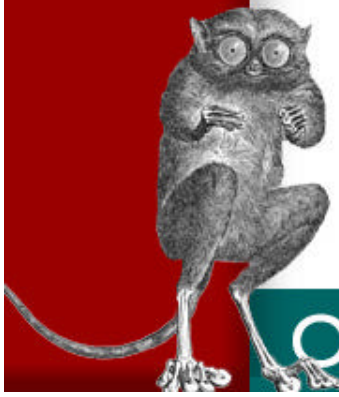
Getting started with wxPython

- Installation is simple -- binary installers are available at SourceForge and via http://wxPython.org/download.php for:
  - Windows: *.exe
  - Linux: *.rpm (and *.deb's are available separately.)
  - OS X: *.dmg, a disk image that contains an Installer package.
- Can be built from source for other Unix-like systems.



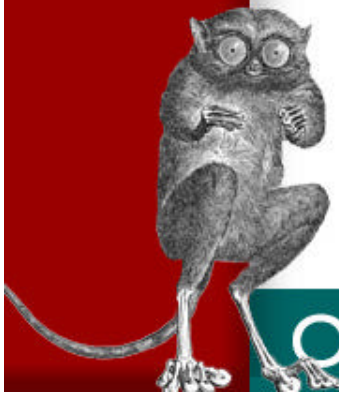
Getting started with wxPython

- Choose an installer.
- Which version of Python do you use?
  - 2.1, 2.2, or 2.3
- Unicode?
  - Windows, but be careful with Win9x/ME
  - OS X, soon
  - Linux/Unix, sooner
- or ANSI?
  - All platforms



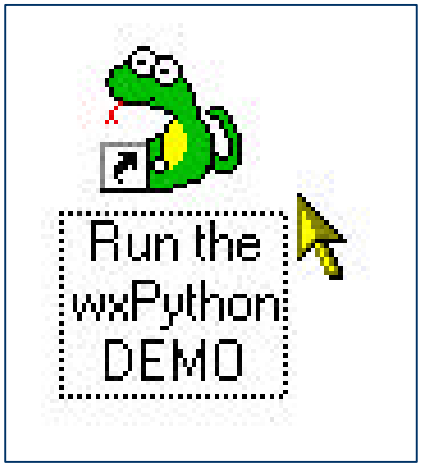
Getting started with wxPython

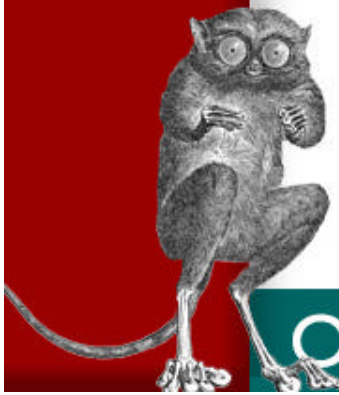
- Choose an editor or development environment:
  - Boa Constructor
  - WingIDE
  - PyAlaMode
  - SCiTE
  - Emacs, vi, etc.
- It’s just plain text, so an ordinary editor and command line will do.



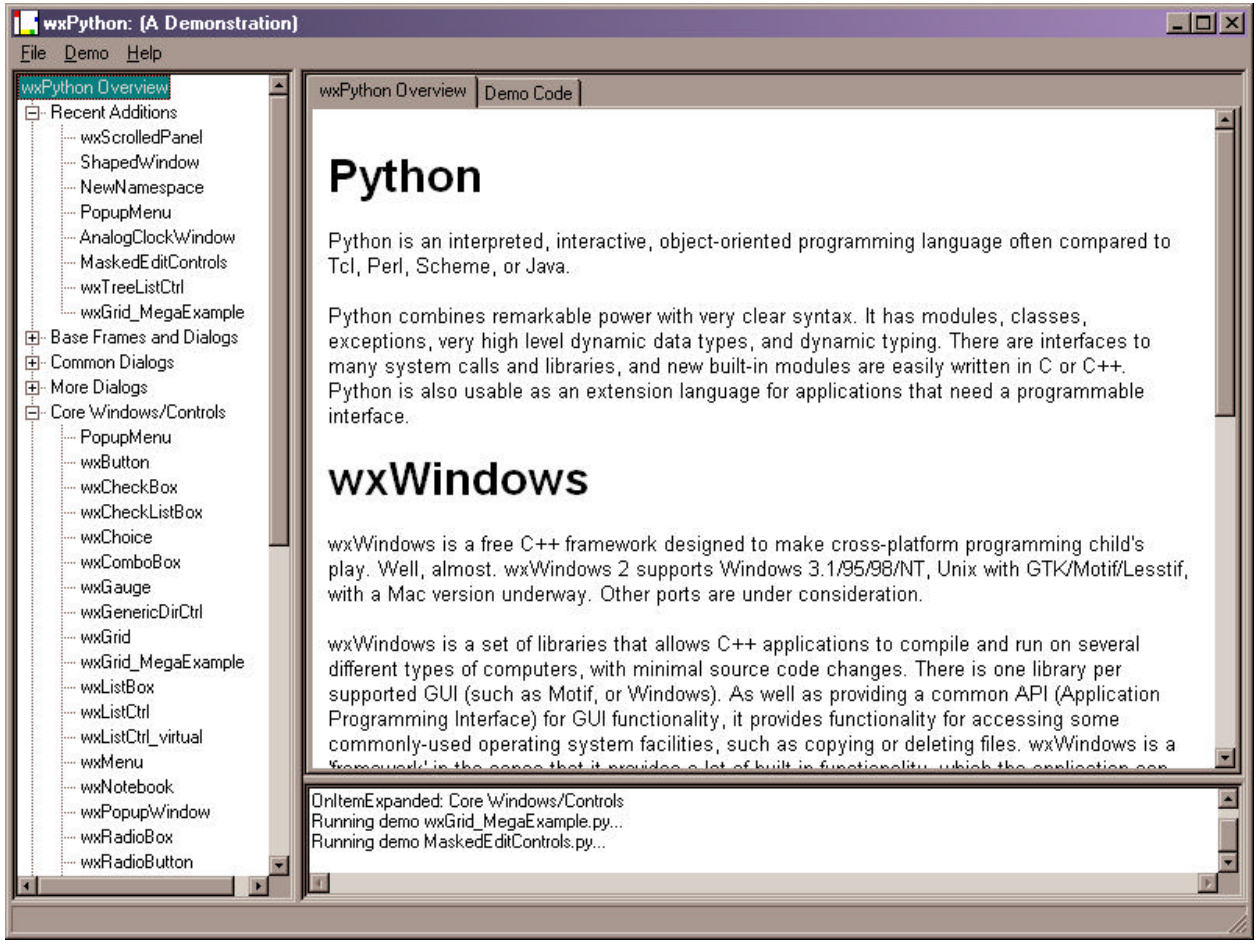
Getting started with wxPython

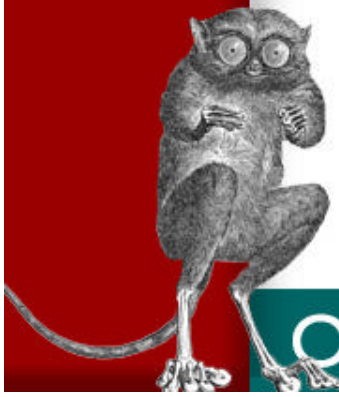
- Ready, set, go!
- The wxPython Demo is a great way to learn about the capabilities of the toolkit.



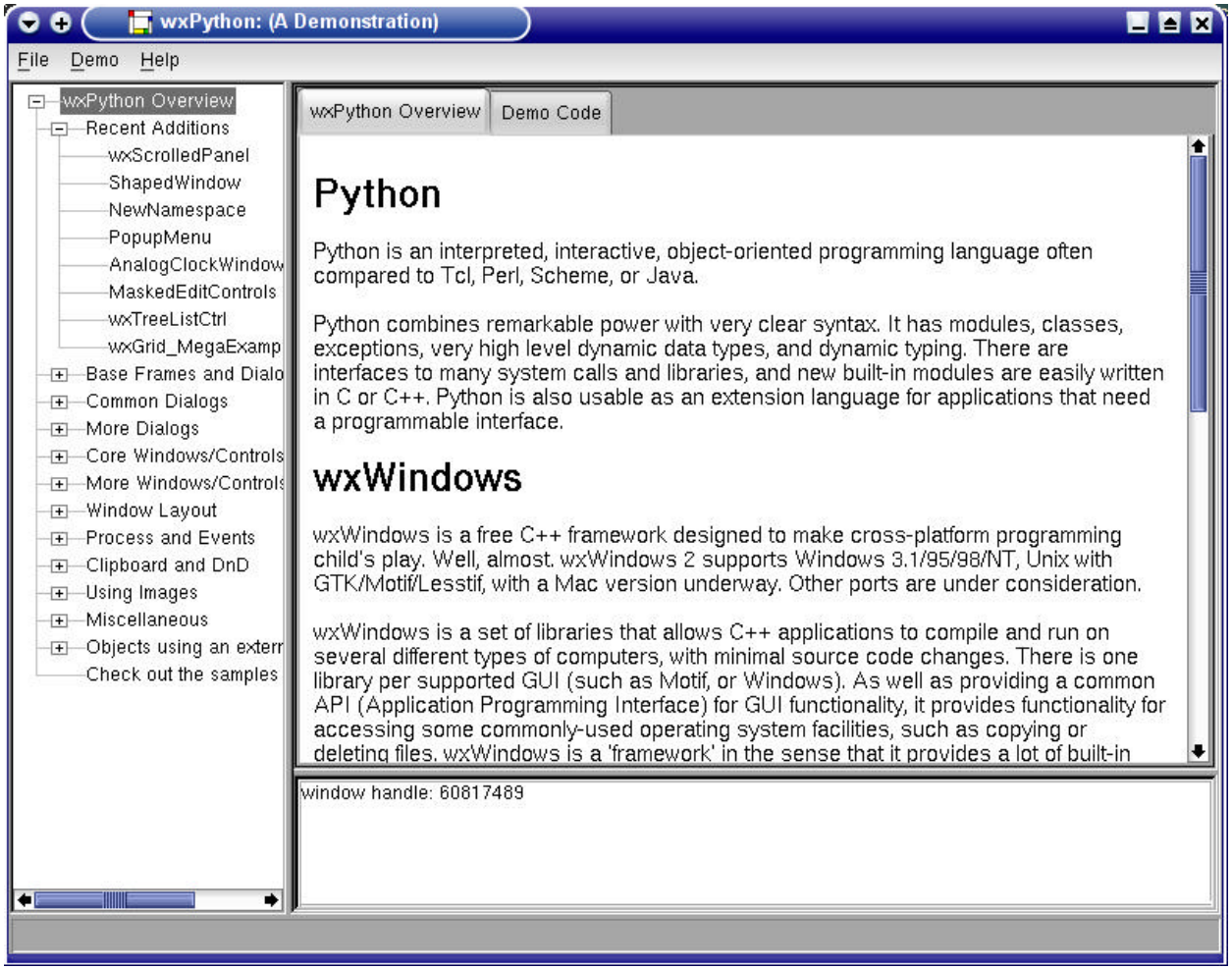


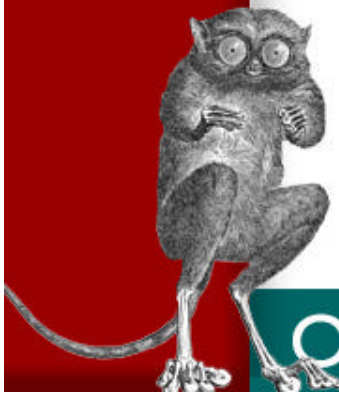
Getting started with wxPython



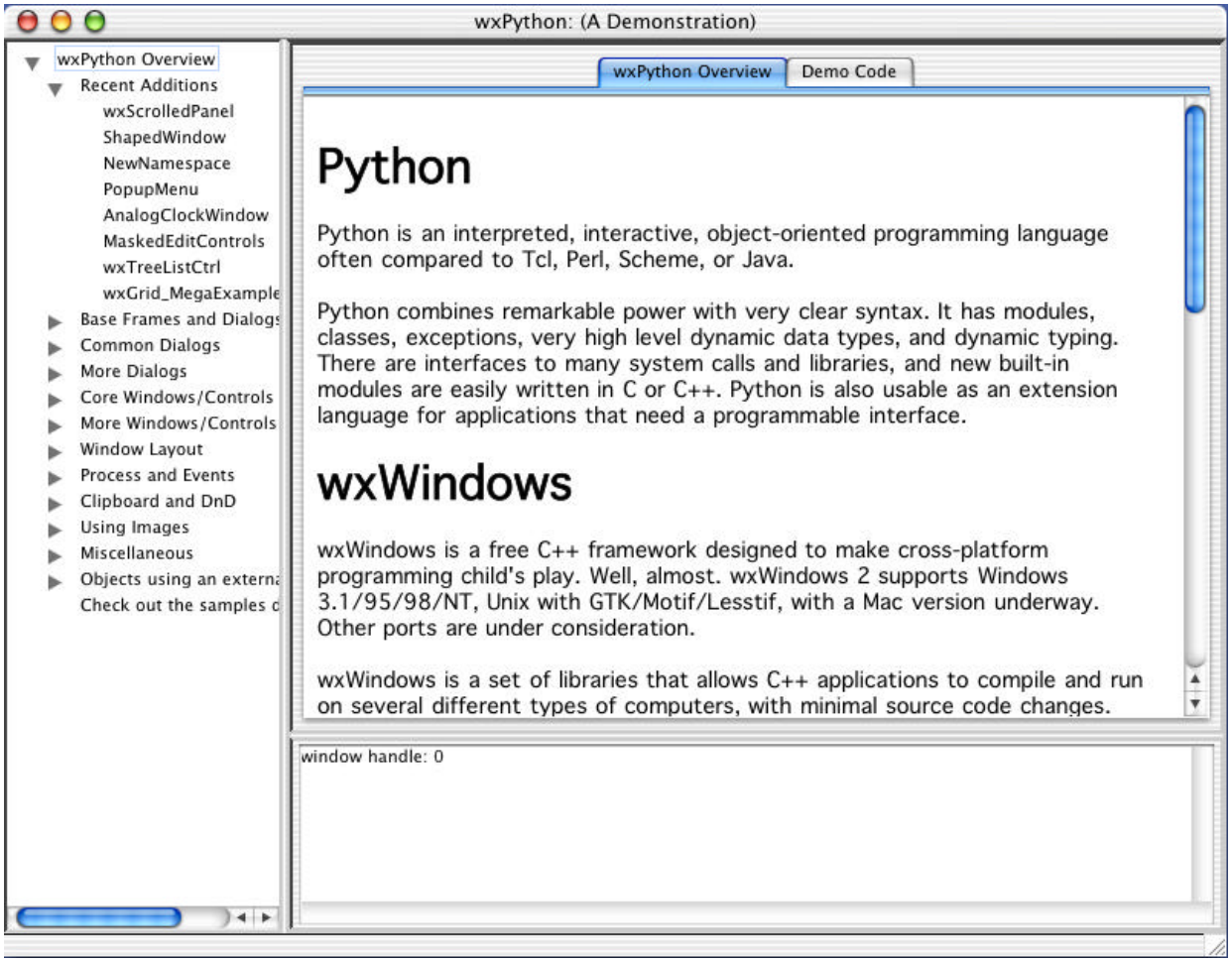


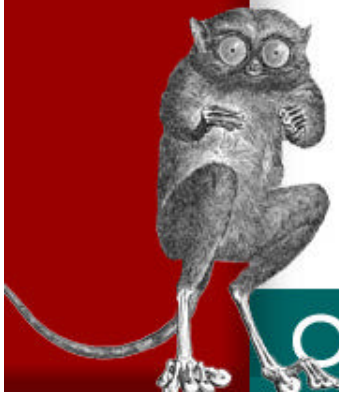
Getting started with wxPython





Getting started with wxPython





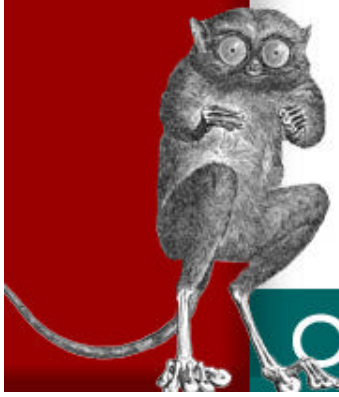
Demo time...

O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR

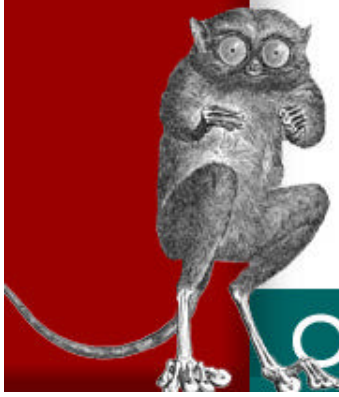
Application fundamentals

```
import wx

class App(wx.App):

    def OnInit(self):
        title = 'Bare Frame'
        frame = wx.Frame(parent=None, id=-1,
                         title=title)
        frame.Show()
        return True

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



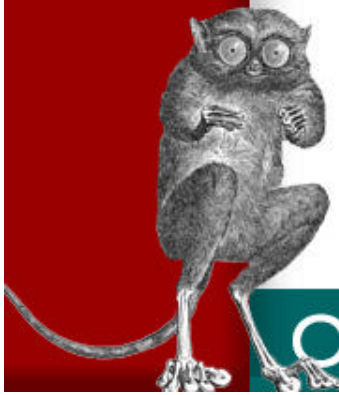
Application fundamentals

```
#!/usr/bin/env python

"""Starting point for simple wxPython programs."""

import wx

class Frame(wx.Frame):
    pass

class App(wx.App):

    def OnInit(self):
        title = 'Spare'
        self.frame = Frame(parent=None, id=-1,
                           title=title)
        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True

if __name__ == '__main__':
    app = App()
    app.MainLoop()
```



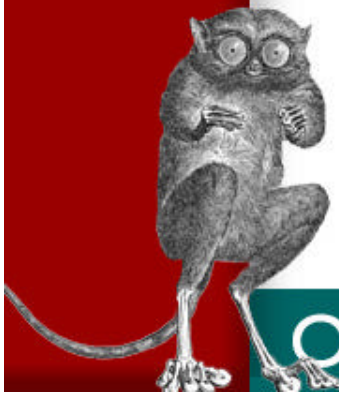
Application fundamentals

```
#!/usr/bin/env python

"""frame.py has a basic Frame class, with App for testing."""

import wx

class Frame(wx.Frame):
    """Frame class."""

    def __init__(self, parent=None, id=-1, title='Title',
                 pos=wx.DefaultPosition, size=(400, 200)):
        """Create a Frame instance."""
        wx.Frame.__init__(self, parent, id, title, pos, size)

class App(wx.App):
    """Application class."""

    def OnInit(self):
        self.frame = Frame()
        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True

def main():
    app = App()
    app.MainLoop()

if __name__ == '__main__':
    main()
```



Application fundamentals

```
#!/usr/bin/env python

"""app.py has a basic application class."""

import wx

from frame import Frame

class App(wx.App):
    """Application class."""

    def OnInit(self):
        self.frame = Frame(title='This is my App')
        self.frame.Show()
        self.SetTopWindow(self.frame)
        return True

def main():
    app = App()
    app.MainLoop()

if __name__ == '__main__':
    main()
```



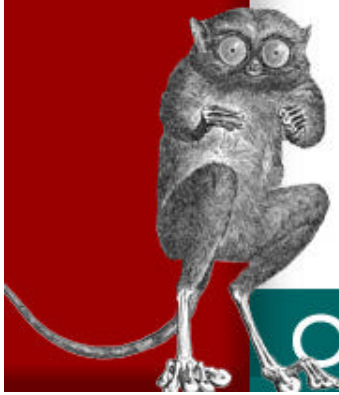
Code break...

O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR

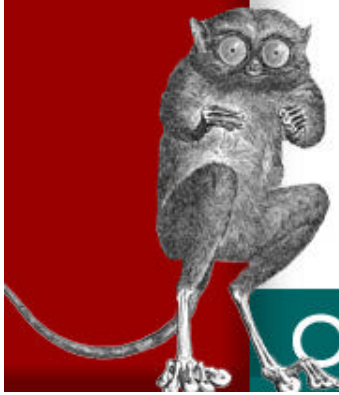
Widgets galore: top level windows

- wx.Frame
  - A container for other windows.
  - Can automatically manage a MenuBar, ToolBar, and a StatusBar.
- wx.Dialog
  - For Modal or Modeless dialog boxes.
- wx.MiniFrame
  - Good for floating tool pallets, etc.
- wx.MDIParentFrame, wx.MDIChildFrame
  - [Take a wild guess :-]

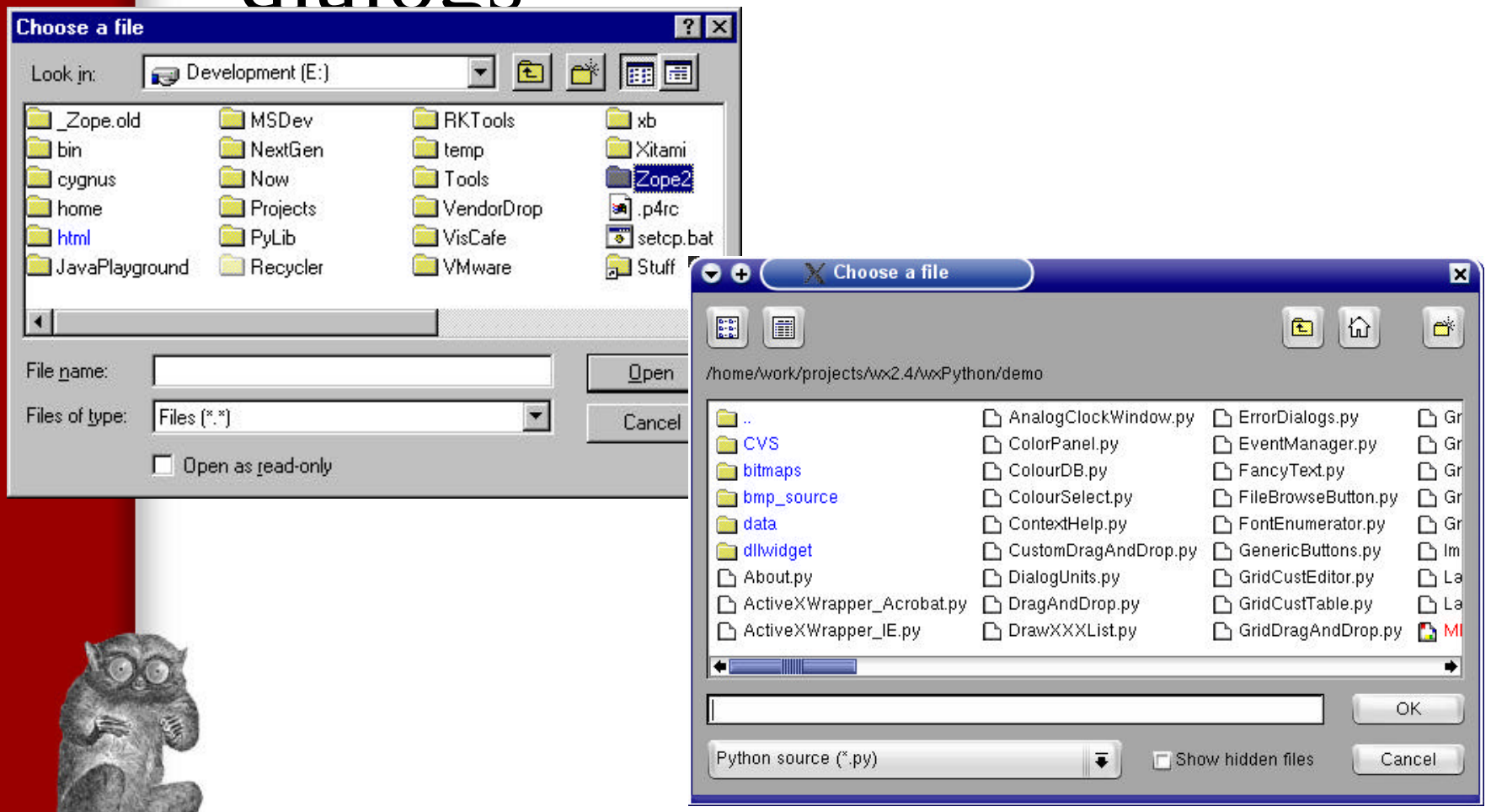


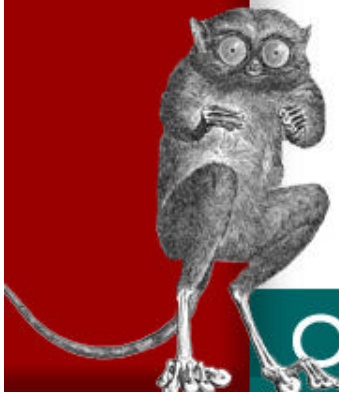
Widgets galore: common dialogs

- All standard Windows common dialogs:
  - Color, Directory, File,
  - Font, PageSetup, Print,
  - Message, Progress,
  - FindReplace, etc.
- For other platforms either native dialogs are used, or suitable recreations in wxWindows are provided.



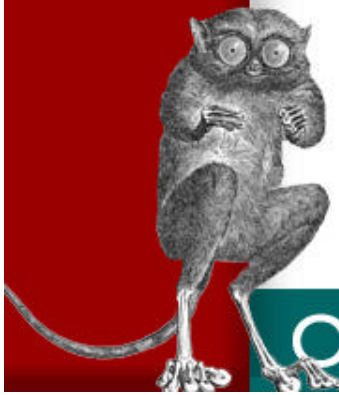
Widgets galore: common dialogs





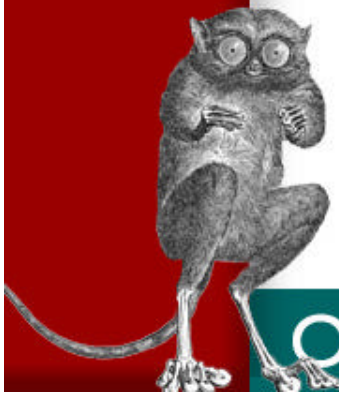
Widgets galore: basic windows

- wx.Window
  - General purpose window.
- wx.Panel
  - Can do tab-traversal of controls.
  - Uses standard system color for the background.
- wx.ScrolledWindow
  - Manages its own scrollbars and scrolling of client area.
  - Transforms coordinates based on scrollbar positions.



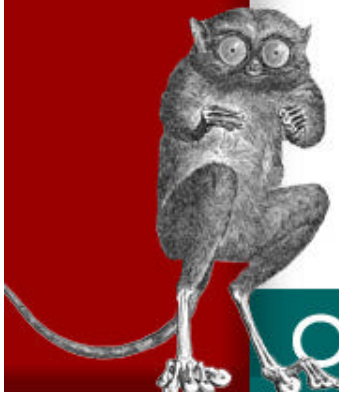
Widgets galore

- wx.SplitterWindow
  - Can be split vertically or horizontally.
  - Draggable sash for redistributing the space between sub-windows.

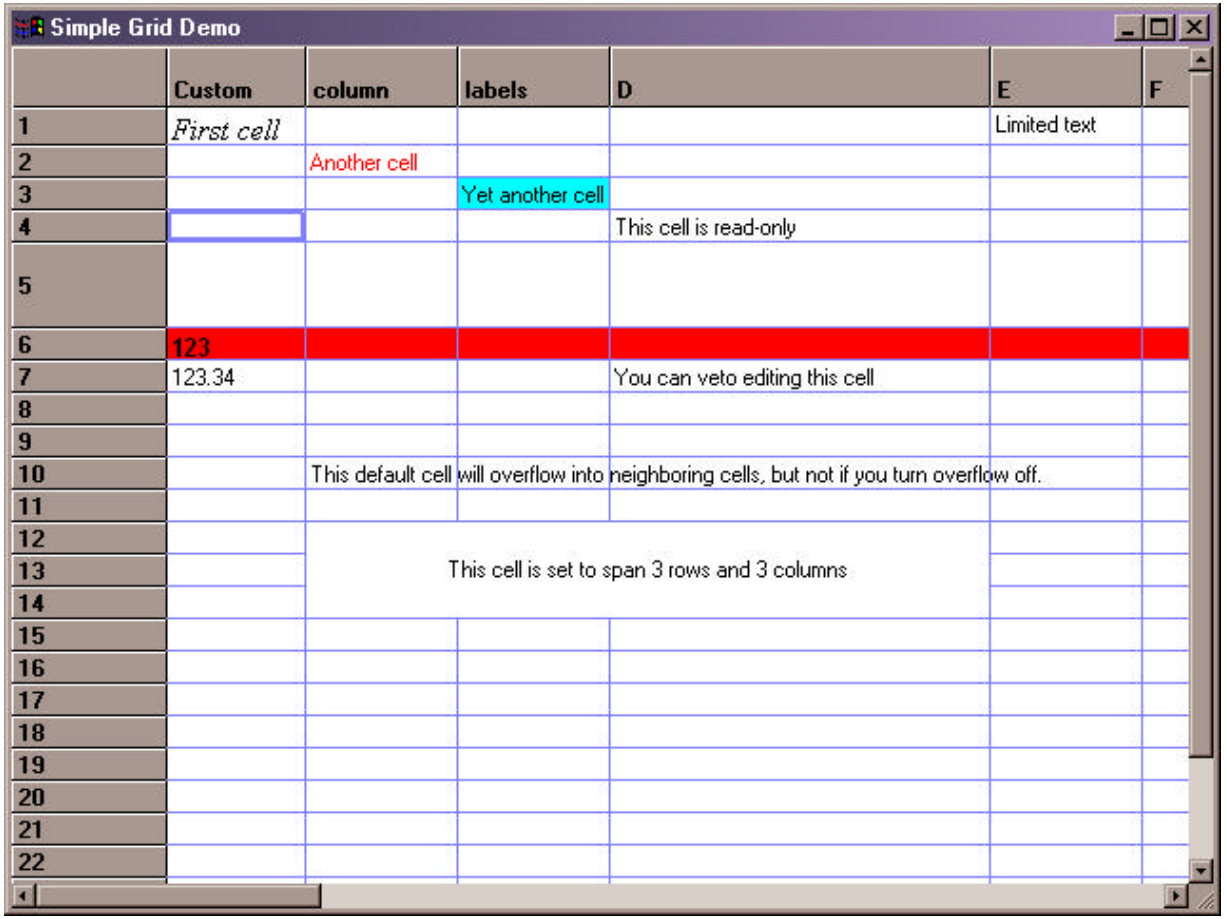


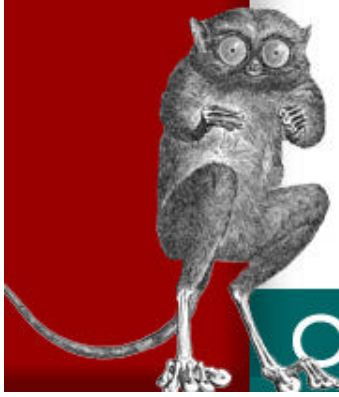
Widgets galore

- wx.grid.Grid
  - Table or spreadsheet-like capabilities.
  - Editors, Renderers, Tables (the data provider) can all be customized and “plugged in”.



Widgets galore

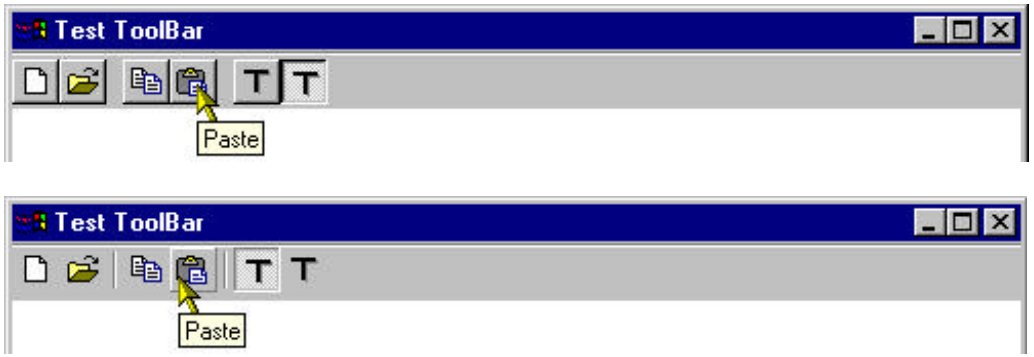


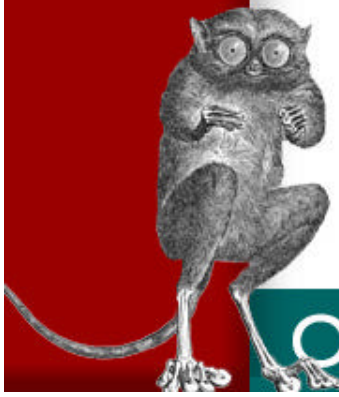


Widgets galore

- wx.StatusBar

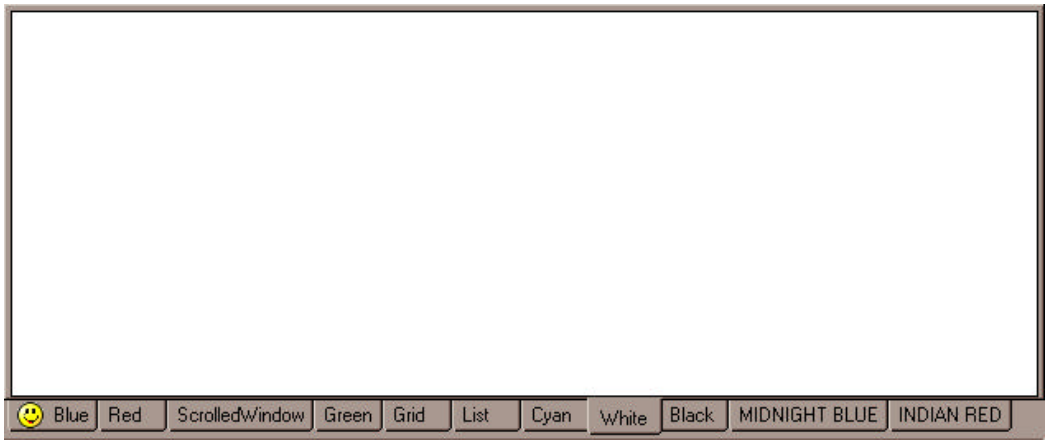
- wx.ToolBar

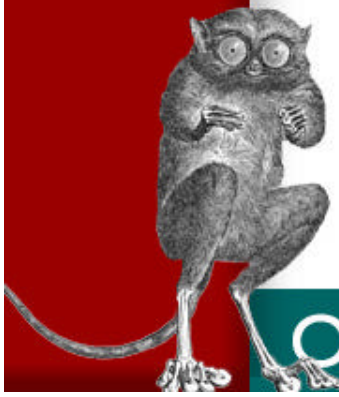




Widgets galore

- wx.Notebook
  - Manages multiple windows with tabs.
  - Tabs can be on any side of the notebook that the platform supports.

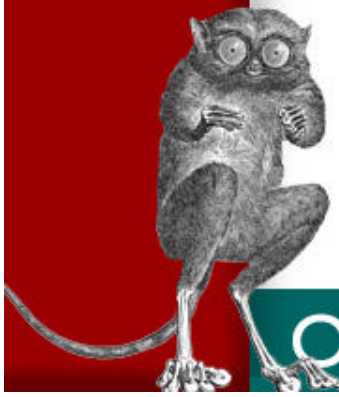




Widgets galore

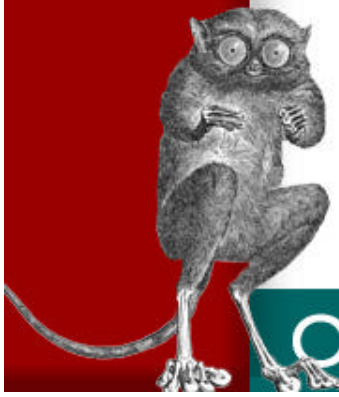
- wx.html.HtmlWindow
  - Capable of parsing and rendering most simple HTML tags.
  - Custom Tag Handlers can change or add to how HTML is rendered.

```
<wxp class="wxButton">
    <param name="label" value="Okay">
    <param name="id"    value="wxID_OK">
</wxp>
```



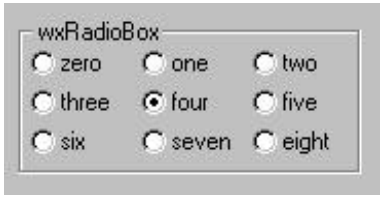
Widgets galore

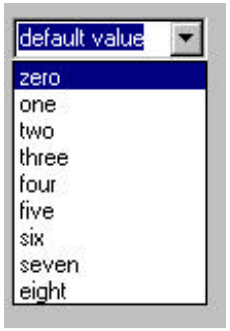
- wx.html.HtmlWindow

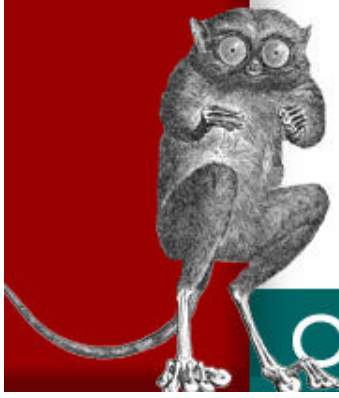


Widgets galore: controls

- wx.Button, wx.BitmapButton
- wx.RadioBox, wx.RadioButton
- wx.CheckBox
- wx.Choice
- wx.ComboBox
- wx.SpinButton

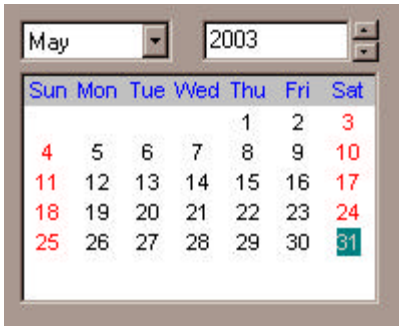


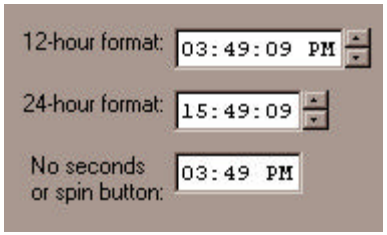


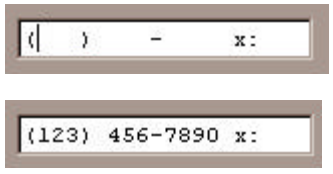


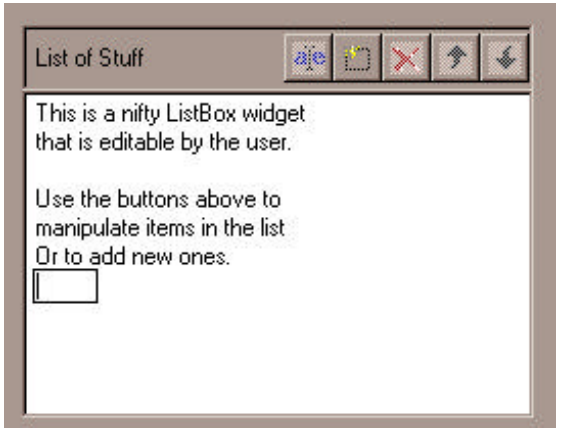
Widgets galore: controls

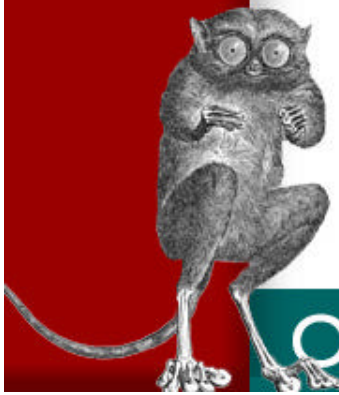
- wx.ToggleButton
- wx.gizmos.EditableListBox
- wx.lib.MaskedEditCtrl
- wx.calendar.CalendarCtrl
- wx.lib.TimeCtrl





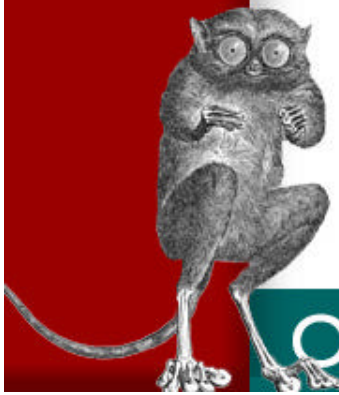






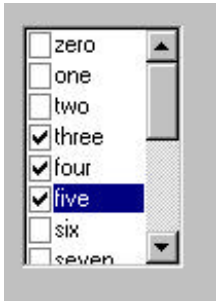
Widgets galore: controls

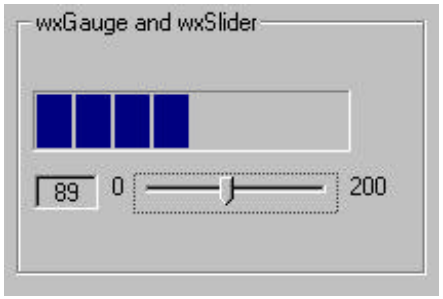
- wx.TextCtrl
  - Optional password masking, multi-line, and simple attributes.

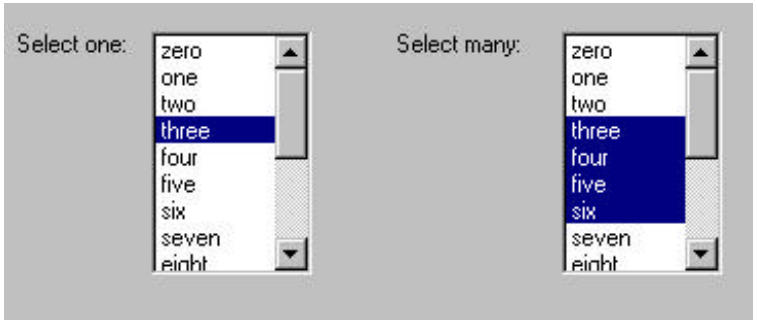


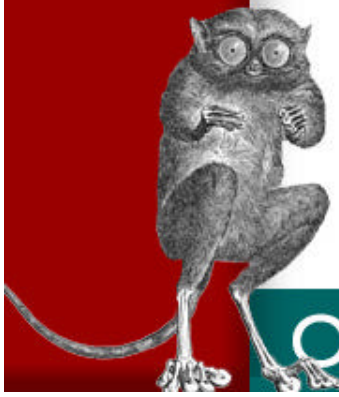
Widgets galore: controls

- wx.ListBox
- wx.CheckListBox
- wx.Gauge
- wx.Slider
- wx.StaticBox



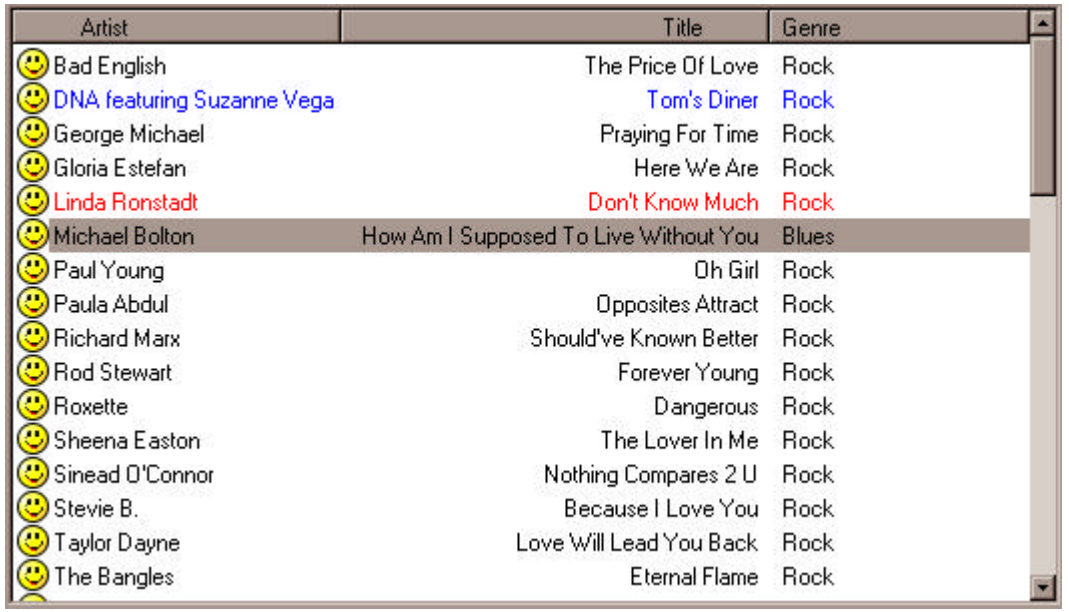




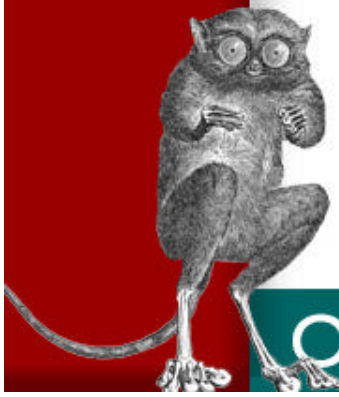


Widgets galore: controls

- wx.ListCtrl
  - Supports list, icon, small icon, report views.
  - Virtual mode, where data items are provided by overloaded methods.

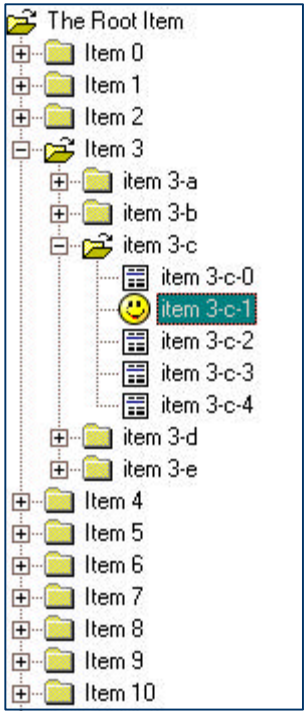


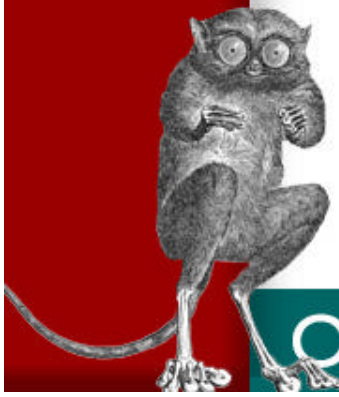
Artist	Title	Genre
Bad English	The Price Of Love	Rock
DNA featuring Suzanne Vega	Tom's Diner	Rock
George Michael	Praying For Time	Rock
Gloria Estefan	Here We Are	Rock
Linda Ronstadt	Don't Know Much	Rock
Michael Bolton	How Am I Supposed To Live Without You	Blues
Paul Young	Oh Girl	Rock
Paula Abdul	Opposites Attract	Rock
Richard Marx	Should've Known Better	Rock
Rod Stewart	Forever Young	Rock
Roxette	Dangerous	Rock
Sheena Easton	The Lover In Me	Rock
Sinead O'Connor	Nothing Compares 2 U	Rock
Stevie B.	Because I Love You	Rock
Taylor Dayne	Love Will Lead You Back	Rock
The Bangles	Eternal Flame	Rock



Widgets galore: controls

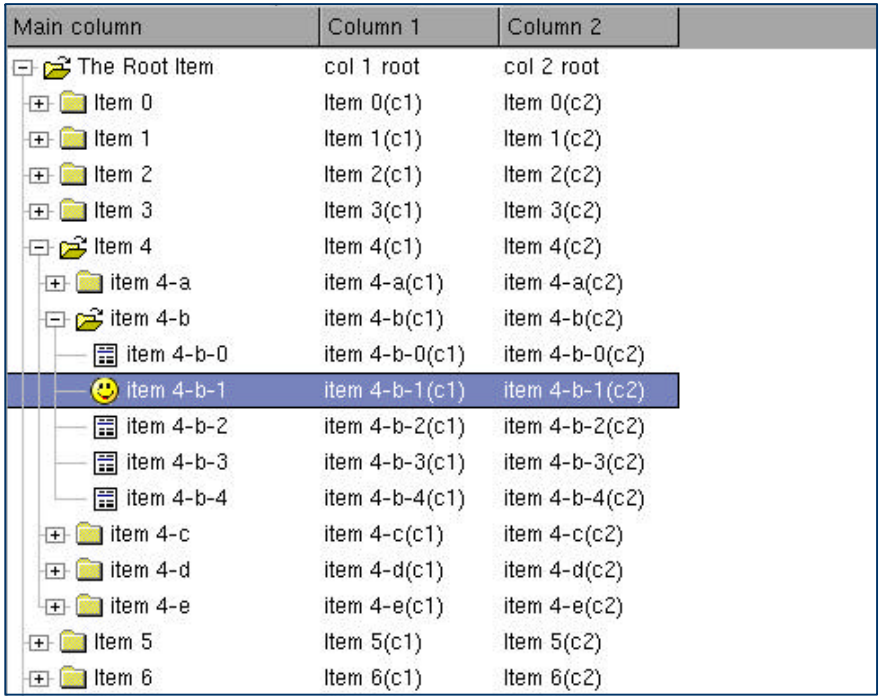
- wx.TreeCtrl
  - Supports images for various node states.
  - Can be virtualized by delaying the adding of items until the parent is expanded.

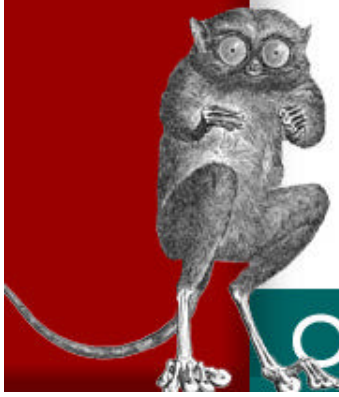




Widgets galore: controls

- wx.gizmos.TreeListCtrl

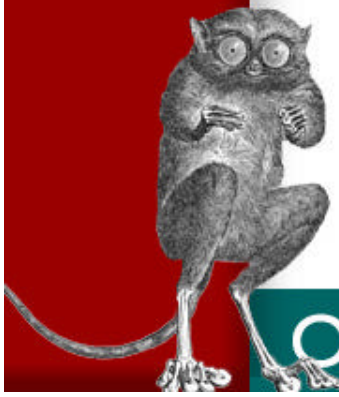




Widgets galore

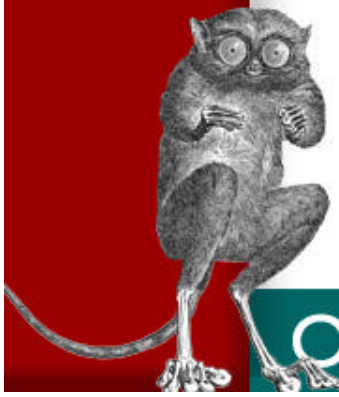
- wx.stc.StyledTextCtrl
  - (wx port of Scintilla)

```
5  #!/bin/env python
6  #----------------------------------------------------------------------
7  # Name:        Main.py
8  # Purpose:     Testing lots of stuff, controls, window types, etc.
9  #
10 # Author:      Robin Dunn
11 #
12 # Created:     A long time ago, in a galaxy far, far away...
13 # RCS-ID:      $Id: Main.py,v 1.76.2.29 2003/05/23 16:47:49 RD Exp $
14 # Copyright:   (c) 1999 by Total Control Software
15 # Licence:     wxWindows license
16 #----------------------------------------------------------------------
17
18 import sys, os, time
19 from   wxPython.wx import *
20 from   wxPython.html import wxHtmlWindow
21
22 import images
```



Event Handling

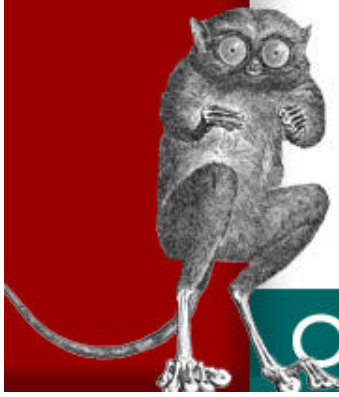
- Most, if not all, GUI systems and toolkits are designed to be event driven, meaning that the main flow of your program is not sequential from beginning to end.
- When something happens that is of interest to you (an event), the system or toolkit calls a bit of your code that deals with that event (event handler).
- When your event handler finishes, control returns to the “main loop” and your program waits for the next event.



Event Handling

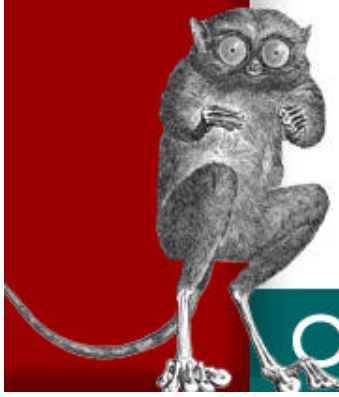
Various event-handling models:

- **Callbacks**: Standalone functions associated with an event by calling a toolkit function. There are encapsulation problems.
- **Message based**: Messages sent to windows for controlling behavior, or for events.
- **Virtual methods**: One for each type of event. Solves encapsulation, but leads to clutter, inflexible classes, and many derived classes just to handle an event differently.
- **Static event tables**: Events are associated with classes and methods at compile time via a table. When the event occurs the tables are searched for a match and the method is invoked.



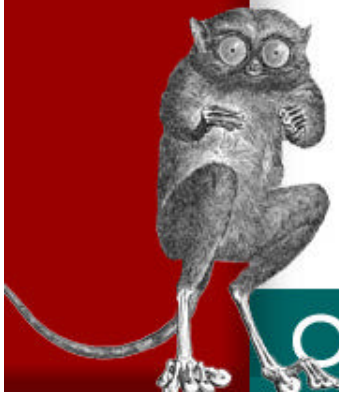
Event Handling

- wxPython uses Dynamic Event Tables
  - Built at run-time.
  - Events can be “connected” to any callable object that will serve as the Event Handler:
    - any method of the class, or other classes,
    - standalone functions, or
    - any object with a __call__ method.
  - Handlers are connected to events with a set of helper functions:
    - EVT_MENU,
    - EVT_PAINT,
    - EVT_SIZE, etc.



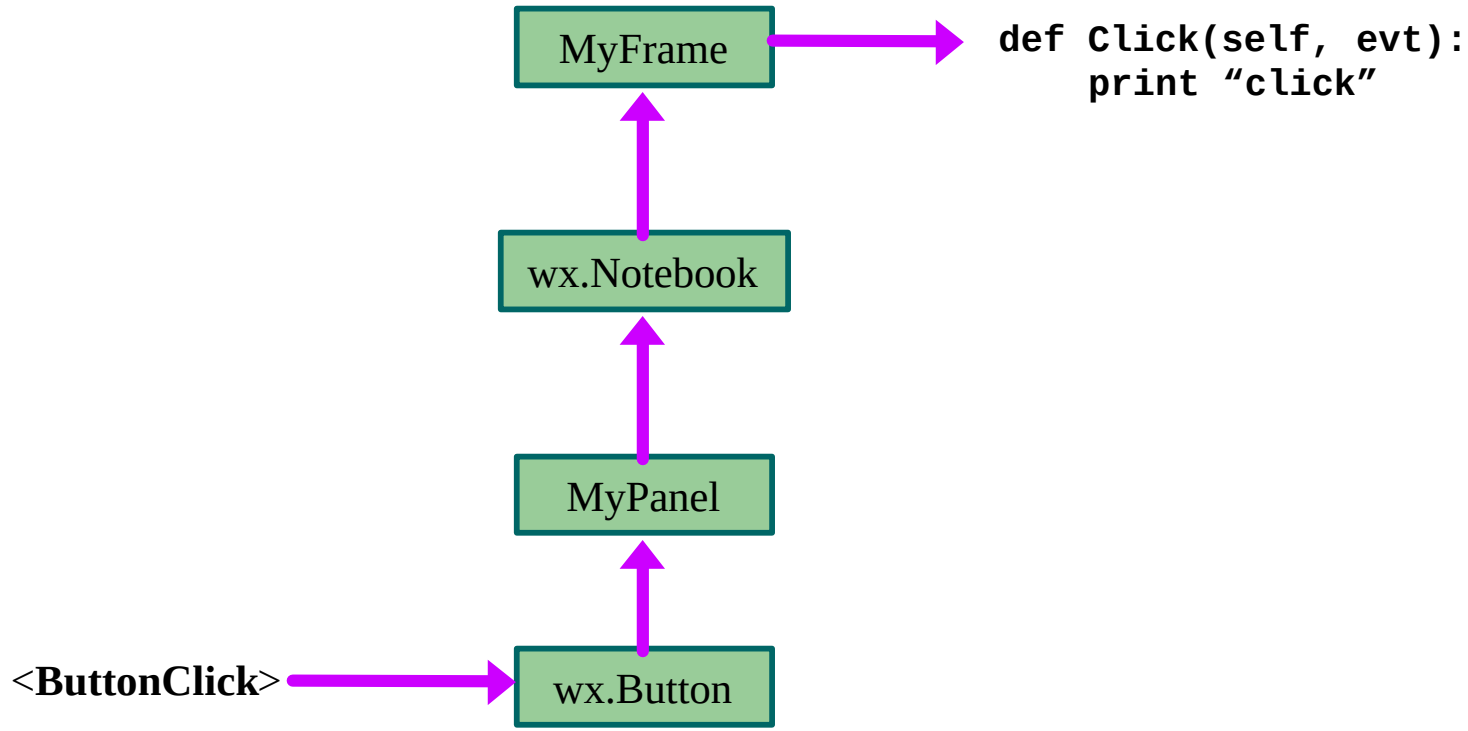
Event Handling

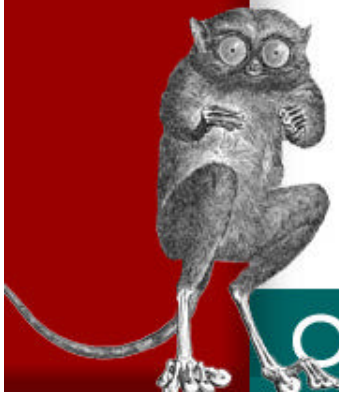
- Each handler is passed an event object when called.
- Two classifications of event objects:
  - Classes derived from **wxEvent**
    - Events that only make sense for the window where the event took place.
  - Classes derived from **wxCommandEvent**
    - Events that may be of interest for any object up the “containment heirarchy.”



In search of Event Handlers…

```
EVT_BUTTON(self, ID, self.Click)
```

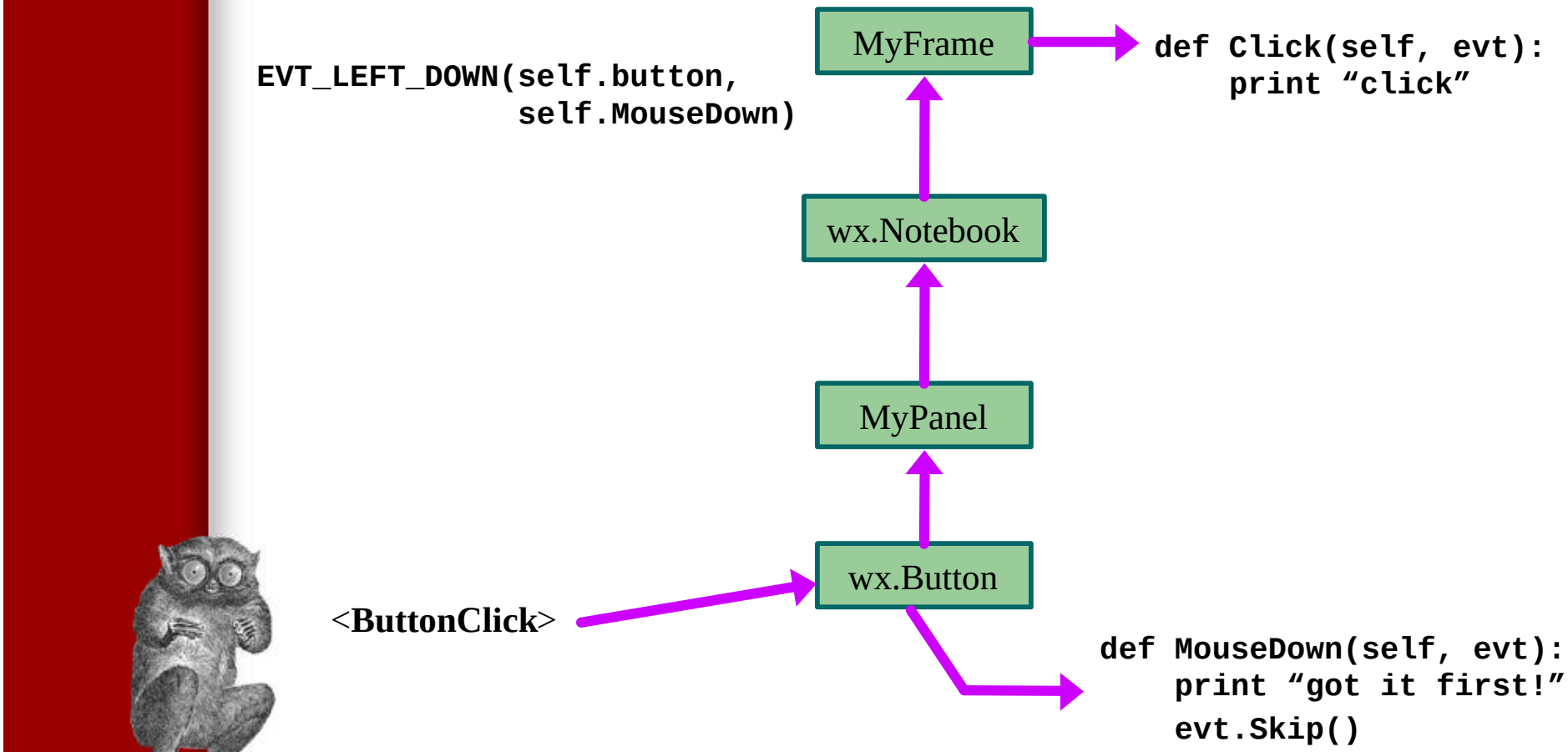




In search of Event Handlers…

```
EVT_BUTTON(self, ID, self.Click)
```

```
EVT_LEFT_DOWN(self.button,
              self.MouseDown)
```




```
def Click(self, evt):
    print “click”
```

```
def MouseDown(self, evt):
    print “got it first!”
    evt.Skip()
```

Code break...

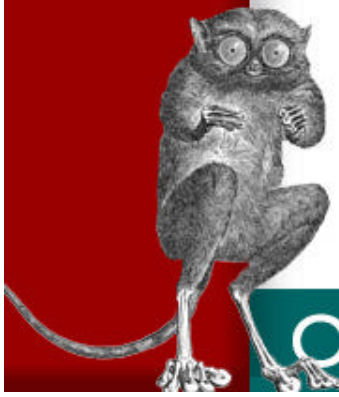
O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR

Organizing your layout

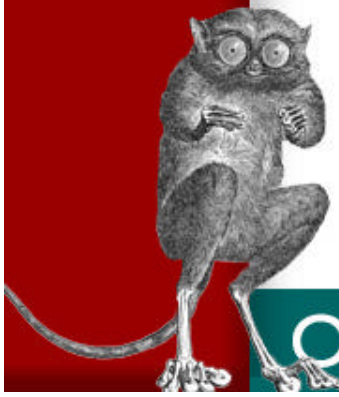
There are various ways to do layout:

- Brute force
  - All widgets are positioned and sized pixel by pixel.
  - Has to be redone in every EVT_SIZE event.
  - Painful, cross-platform issues.
- Layout Constraints
  - Powerful, but complex and verbose.
  - Deals with the relationships between widgets.
  - See the docs and demo for more details.
- Sizers
  - Not as flexible or complex, but powerful enough.
  - Worth the pain.

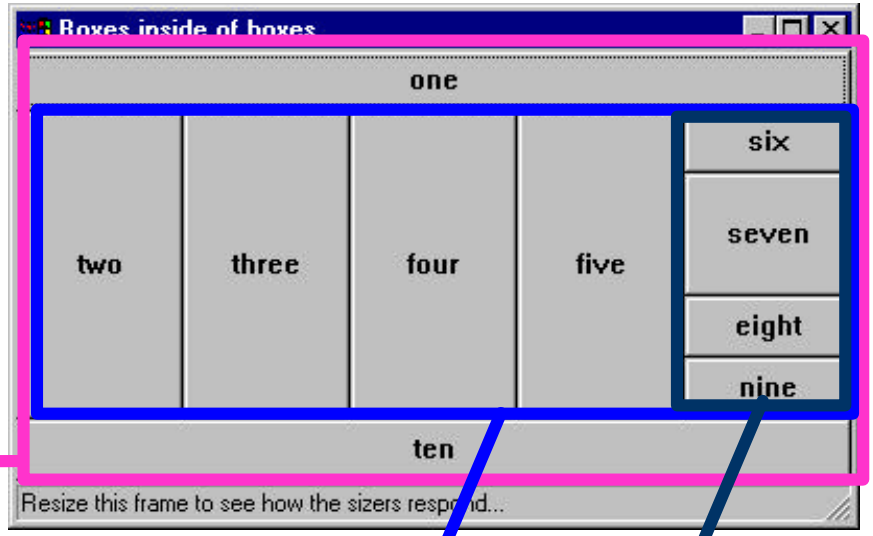


Organizing your layout

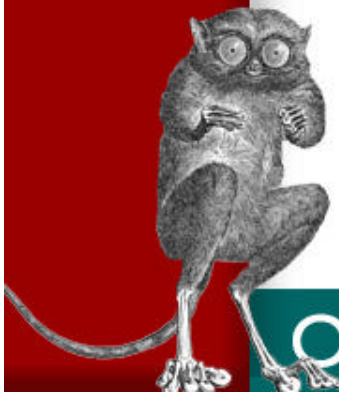
- Sizers
  - Similar to LayoutManagers in Java.
  - Not as flexible as LayoutContraints, but much simpler, once you get over the hump.
  - Relationships defined by containment within sizers or nested sizers.
  - All items (windows or nested sizers) added to a Sizer are laid out by a specific algorithm determined by the class of sizer.
  - An item's position within its allotted space is also controllable.



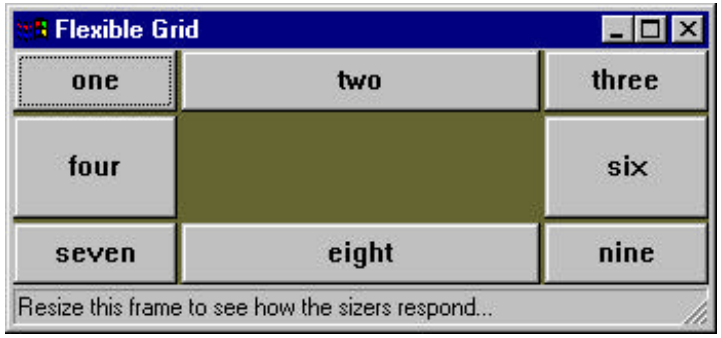
wx.BoxSizer

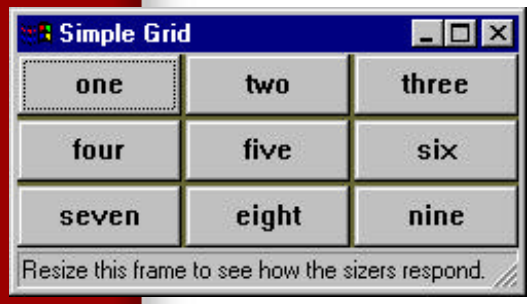


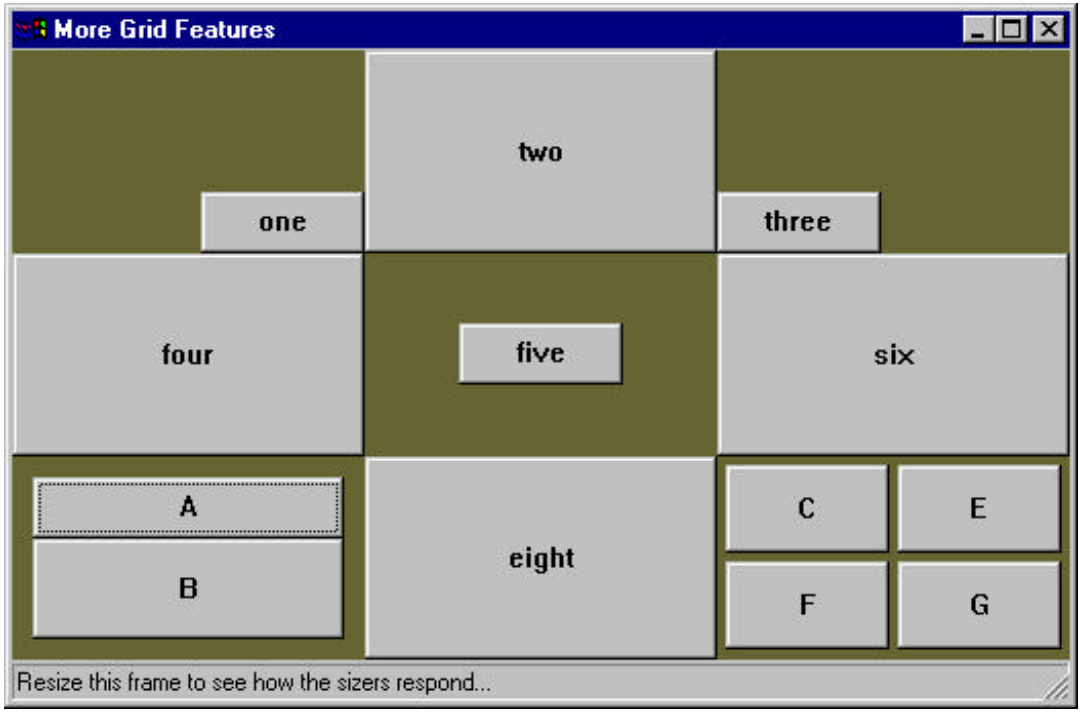

```
box = wx.BoxSizer(wx.VERTICAL)
box.Add(wx.Button(win, 1010, "one"), 0, wx.EXPAND)
box2 = wx.BoxSizer(wx.HORIZONTAL)
box2.Add(wx.Button(win, 1010, "two"),   0, wx.EXPAND)
box2.Add(wx.Button(win, 1010, "three"), 0, wx.EXPAND)
box2.Add(wx.Button(win, 1010, "four"),  0, wx.EXPAND)
box2.Add(wx.Button(win, 1010, "five"),  0, wx.EXPAND)

box3 = wx.BoxSizer(wx.VERTICAL)
box3.Add(wx.Button(win, 1010, "six"),   0, wx.EXPAND)
box3.Add(wx.Button(win, 1010, "seven"), 2, wx.EXPAND)
box3.Add(wx.Button(win, 1010, "eight"), 1, wx.EXPAND)
box3.Add(wx.Button(win, 1010, "nine"),  1, wx.EXPAND)

box2.Add(box3, 1, wx.EXPAND)
box.Add(box2, 1, wx.EXPAND)
box.Add(wx.Button(win, 1010, "ten"), 0, wx.EXPAND)
```

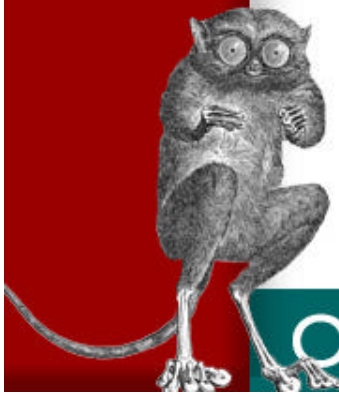


wx.GridSizer

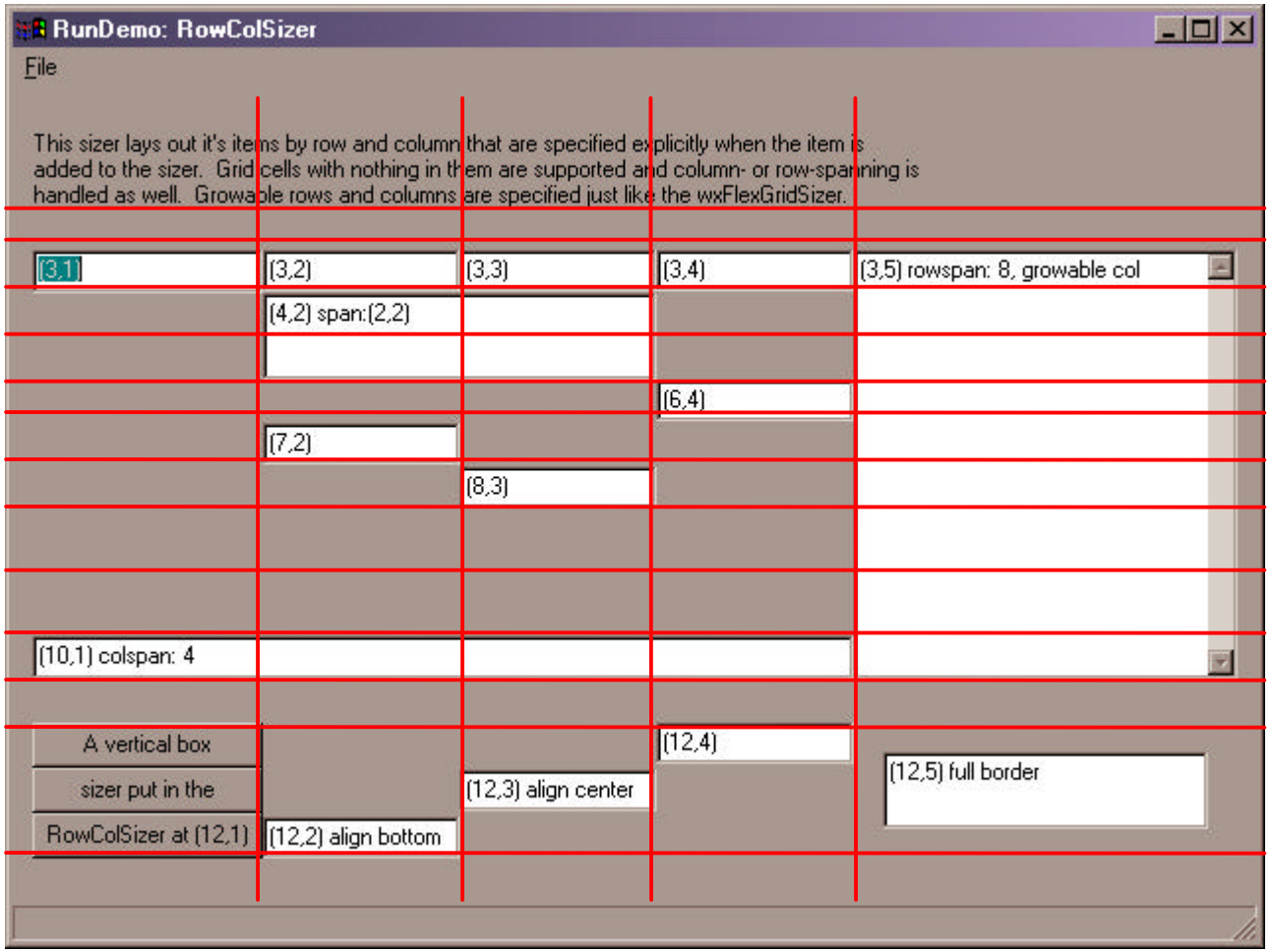


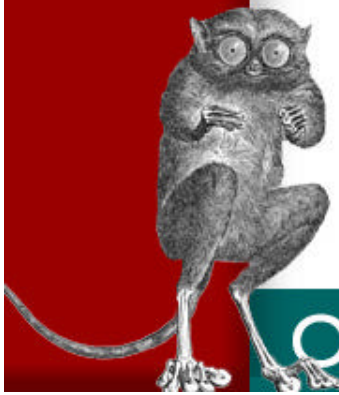






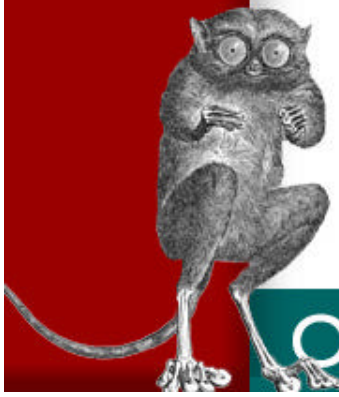
RowColSizer





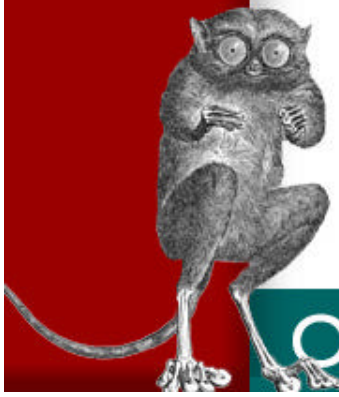
Drawing

- A wx.DC is a *device context* onto which graphics and text can be drawn.
- Represents a number of output devices in a generic way:
  - windows
  - printers
  - bitmaps
  - the whole screen
- The same piece of code may be used to draw on different devices.



Drawing

- DC's have many drawing primitives:
  - DrawArc, DrawBitmap, DrawElipse, DrawLine, DrawLines, DrawPoint, DrawPolygon, DrawRectangle, DrawRoundedRectangle, DrawSpline, DrawText
- And work with GDI objects:
  - wx.Font, wx.Bitmap, wx.Brush, wx.Pen, wx.Mask, wx.Icon, etc.



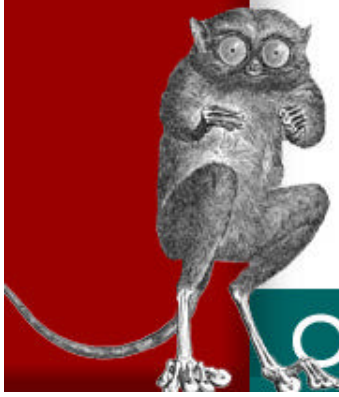
Code break...

O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR

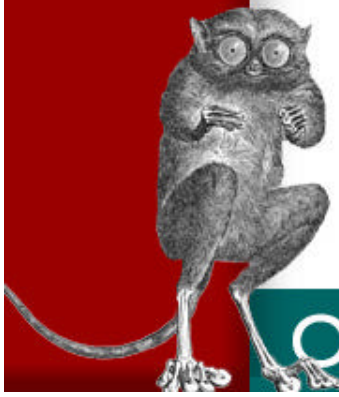
Debugging with PyCrust

- Interactive Python Shell
- 100% Python
- Part of wxPython
- Standalone App
- Embeddable Components



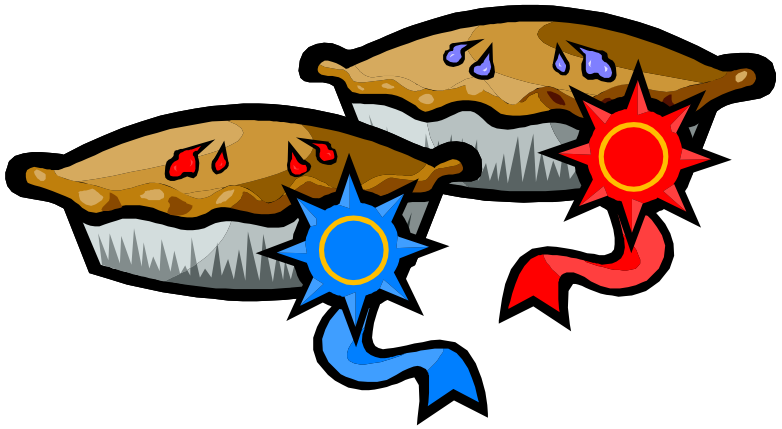
PyCrust Embeddable Components

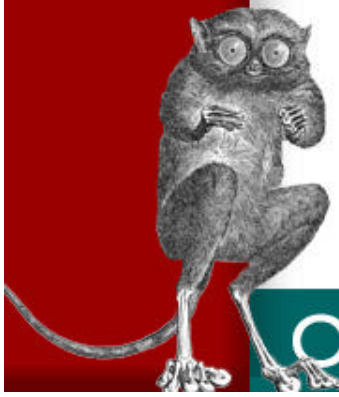
- Interactive Shell: py.shell
- Namespace Viewer: py.filling
- Integrated Combo: py.crust



PyCrust Features

- Colorized Python Code
- Attribute/Method Auto-Completion
- Function/Method Calltips
- Multiline Command Editing
- Command History/Recall





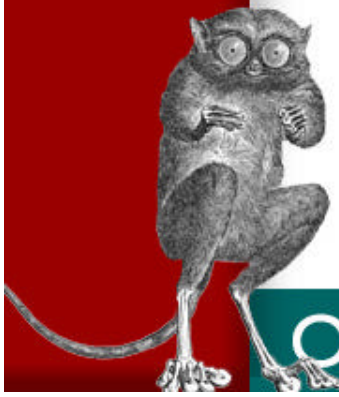
PyCrust demo...

O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR

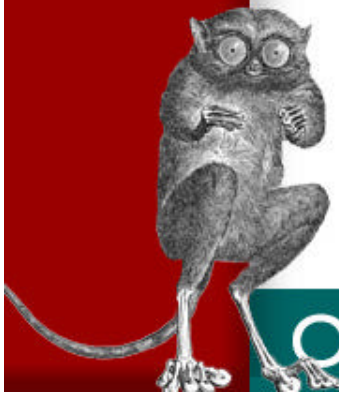
Other tools

- wxDesigner
- Boa Constructor
- wxGlade
- WingIDE
- PythonCard
- Chandler



Last minute additions

- Slides and sample code available at http://wxPython.org/OSCON2003/
- wxPython T-shirts and other gear available at http://www.cafepress.com/wxPython/



Questions?

O'REILLY
OPEN
SOURCE
CONVENTION™

July 7-11, 2003 • PORTLAND, OR