# wxPython 101

A quick walkthrough on GUI building for your program

- Minchang (Carson) Zhang

# An Introduction to wxPython

- wxPython Architecture
- Widgets, Menus, Status Bars
- Layout
- Event and multi-threading
- How to design your GUI
- Summary and more

# wxPython Architecture

A brief introduction on the wxPython architecture and how it works

# wxPython Introduction - Architecture

Base Widget

Top Level Widget

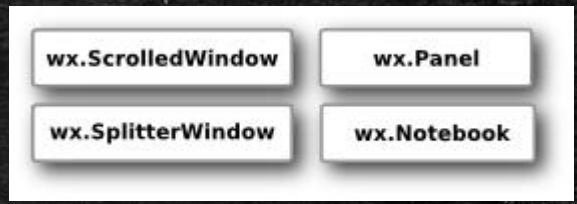
Containers

Dynamic Widget

Static Widget

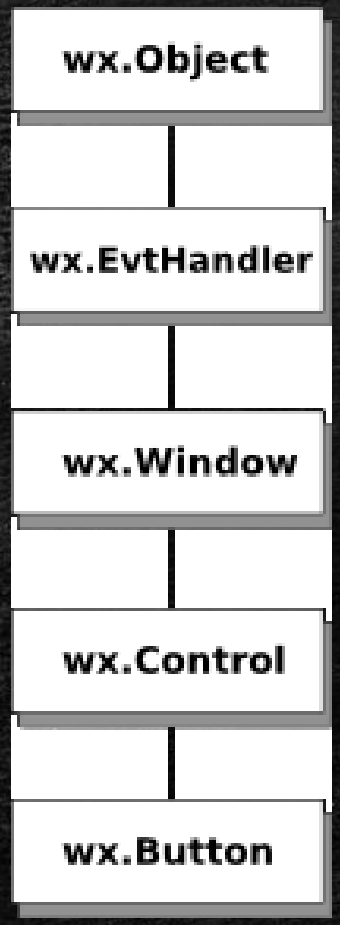
Other Widget





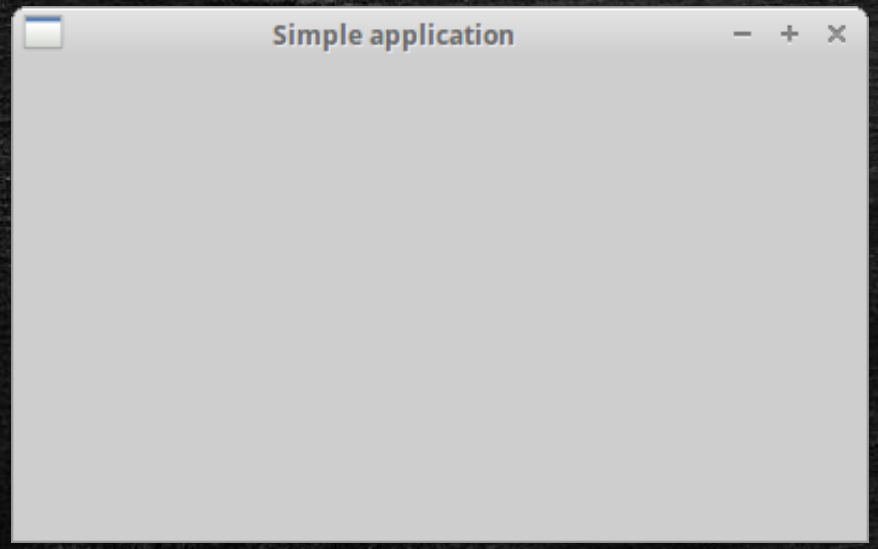



# wxPython Introduction - Inheritance




- Widgets can inherit functionality from other widgets
  - Existing classes are called base classes, parents, or ancestors.
  - The widgets that inherits called derived widgets, child widgets or descendants.
- Button inherit from 4 base classes, the closest is wx.Control
- Button and window react to events
- Finally all objects inherit from wx.Object

# A simple application




```
import wx


app = wx.App()


frame = wx.Frame(None, title='Simple
application')
frame.Show()


app.MainLoop()
```

# Widgets, Menus, Status Bars

Brief illustrations of how they work

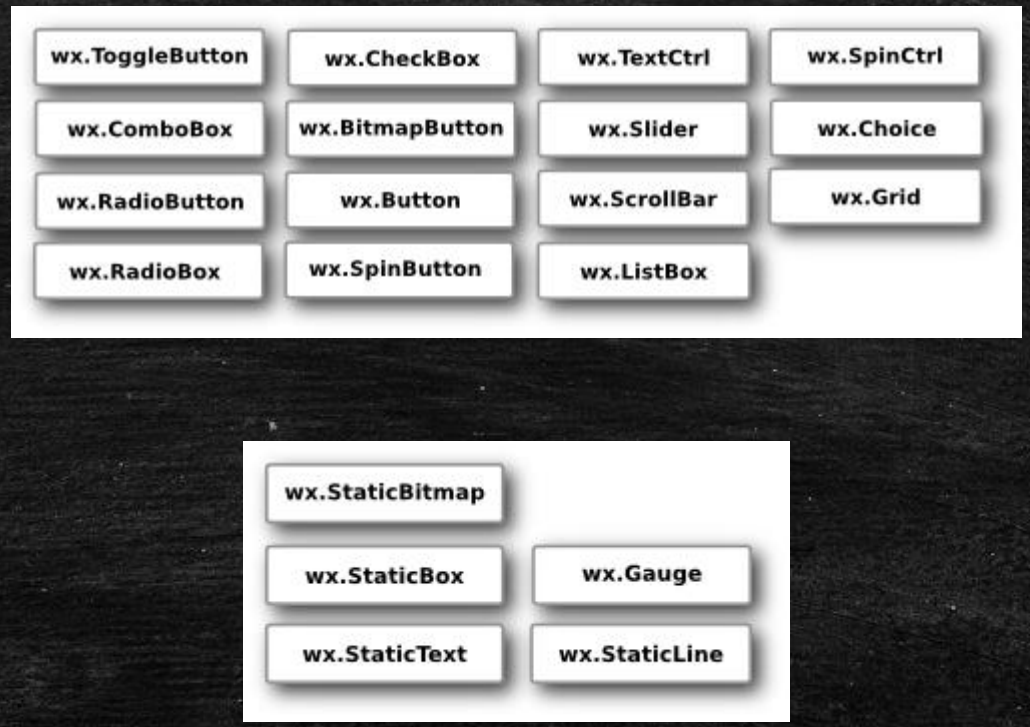
# Widgets

Widgets are used across the GUI for user interaction

Each widget has its own property

Bind widget with response function

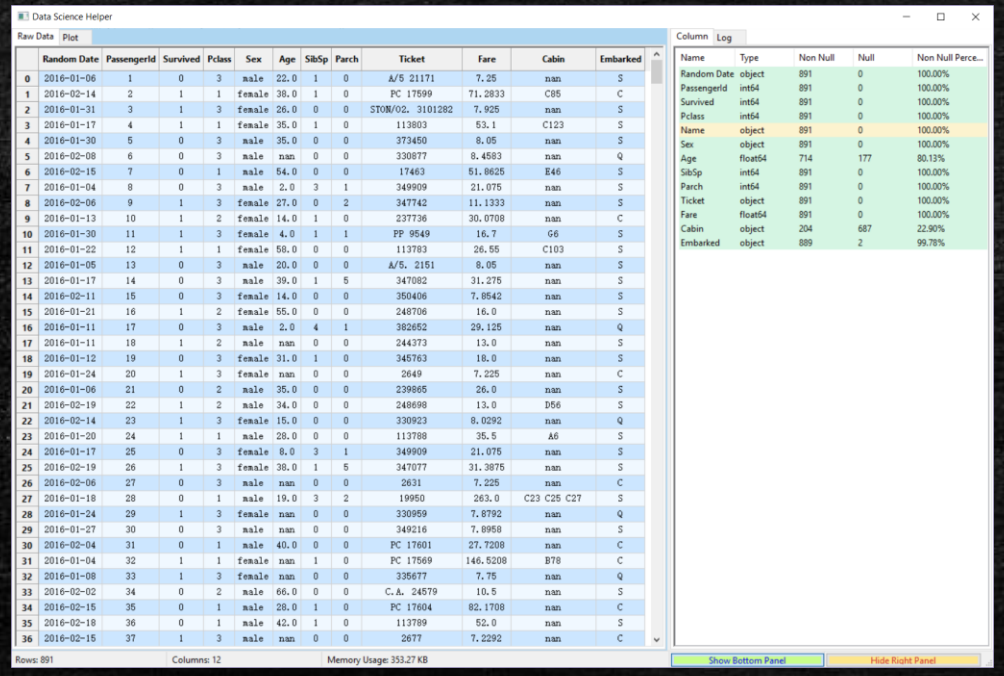
Ref: https://docs.wxwidgets.org/trunk/page_screenshots.html



# Widgets – how do they talk to each other

## In same container

- Use oop rules, just call them
- self.text.SetBackgroundColor(“Red”)




## In different containers

- Follow the family-relationship of the containers
  - GetParent or GetChildren
- PubSub
  - pub.subscribe(self.PrintMessage, "LOG_MESSAGE")
  - def PrintMessage(self, log_message)
  - pub.sendMessage("LOG_MESSAGE", log_message="something")

# Menu Bar, Toolbar, Status Bar

## How does is work

- create wx menubar/toolbar/statusbar object first
- create menubar/toolbar/statusbar
- append items into menubar/toolbar/statusbar
- set menubar/toolbar/statusbar into frame
- bind responding functions

## Code

- menubar = wx.MenuBar()
- fileMenu = wx.Menu()
- fileItem = fileMenu.Append(wx.ID_EXIT, 'Quit', 'Quit application')
- menubar.Append(fileMenu, '&File')
- self.SetMenuBar(menubar)
- self.Bind(wx.EVT_MENU, self.OnQuit, fileItem)

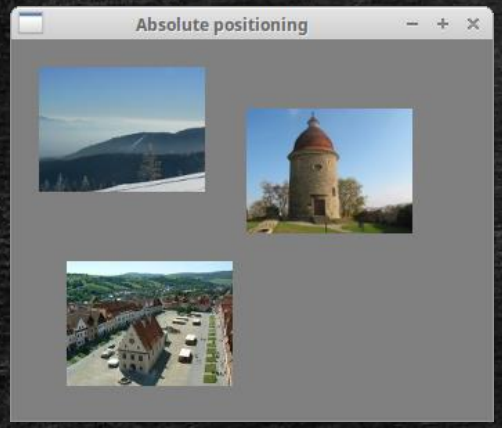
# Layout

How your GUI presents

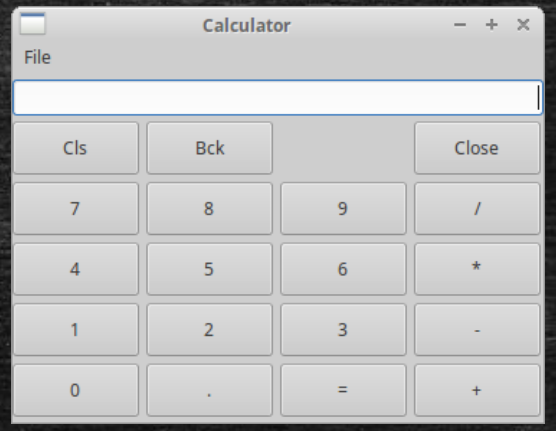
# Layout

- Absolute positioning using pixels
  - Size & position will not change
  - NOT recommended
- Sizers: handle all the positions
  - wx.BoxSizer: put several widgets into a row or a column
  - wx.StaticBoxSizer: similar to box sizer but with a static box around the sizer
  - wx.GridSizer: lays out widgets in 2D table where cells have identical sizes
  - wx.FlexGridSizer: similar to grid sizers but with flexible size cells
  - wx.GridBagSizer: the most flexible sizers where item can span

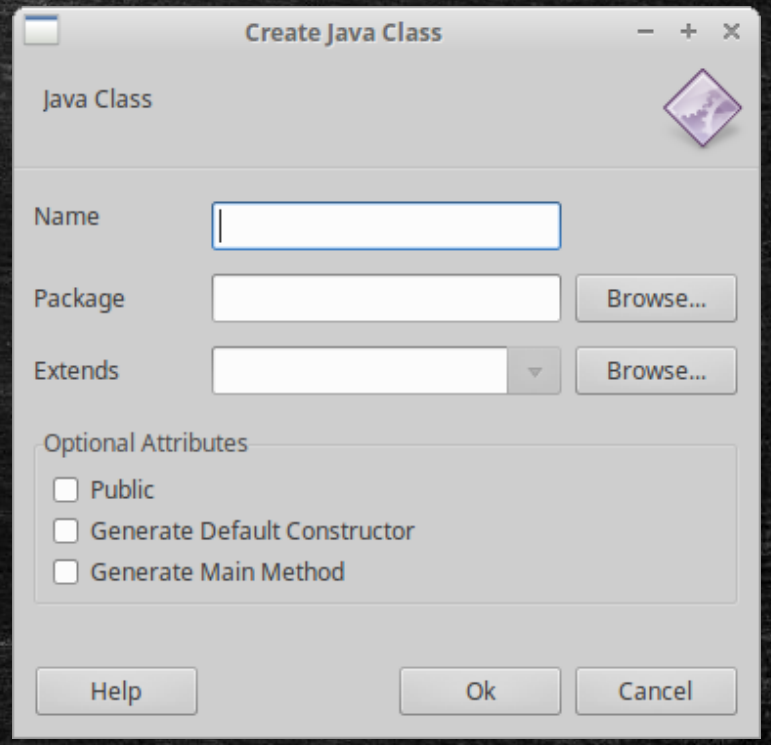
# Layout - Sizers




Absolute Positioning

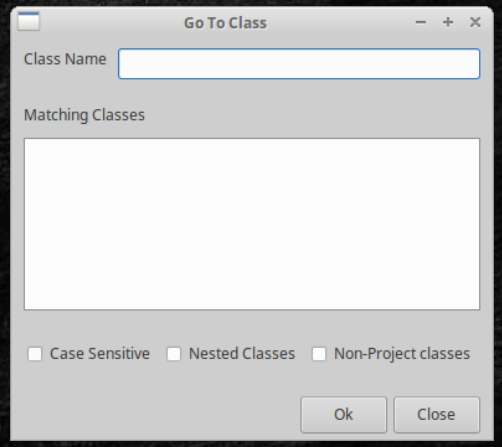



GridSizer

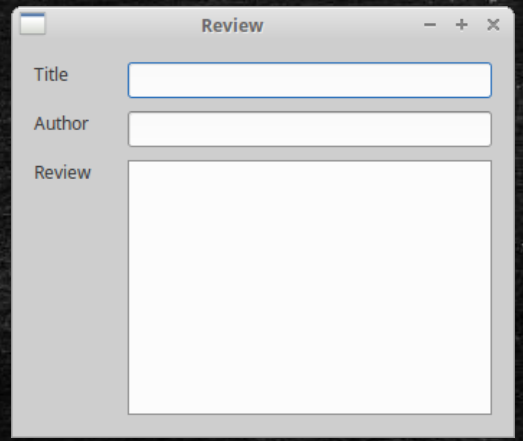



GridBagSizer

StaticBoxSizer



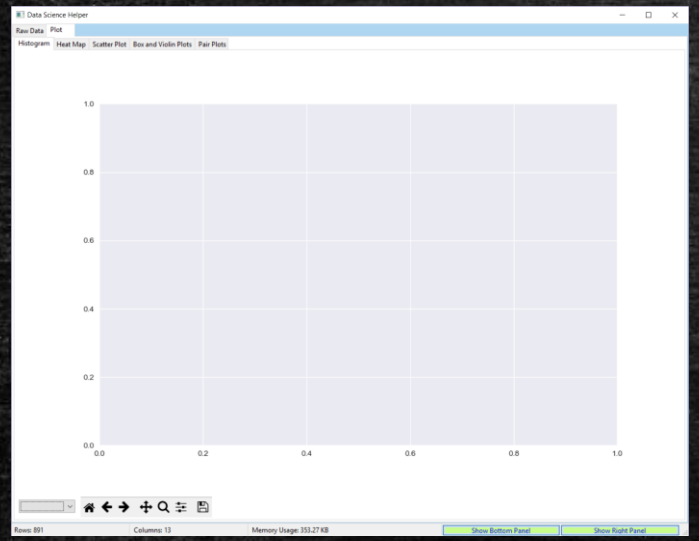

Box Sizers




FlexGridSizer

Ref: http://zetcode.com/wxpython/layout/

# Layout - Sizers




How it works:

- create sizer object first
- append items into sizer
- Adjust layouts
- set sizer into frame/panel

```
self.figure = Figure()
self.axes = self.figure.add_subplot(111)
self.canvas = FigureCanvas(self, -1, self.figure)

self.toolbar = NavigationToolbar(self.canvas)

self.dropdown_menu = wx.ComboBox(
    self, choices=self.available_columns, style=wx.CB_READONLY
)
self.Bind(wx.EVT_COMBOBOX, self.column_selected)

toolbar_sizer = wx.BoxSizer(wx.HORIZONTAL)
toolbar_sizer.Add(self.dropdown_menu, 0, wx.ALL | wx.ALIGN_CENTER, 5)
toolbar_sizer.Add(self.toolbar, 0, wx.ALL, 5)

sizer = wx.BoxSizer(wx.VERTICAL)
sizer.Add(self.canvas, 1, wx.LEFT | wx.TOP | wx.GROW)
sizer.Add(toolbar_sizer)
self.SetSizer(sizer)
```

# Event and multi-threading

Advanced knowledge but you have to know when to use them

# Event

- Events: All GUI applications are event-driven.
  - Identify event binder name
  - Create Event handler
  - Bind an event to an event handler
  - For example:
    - self.Bind(wx.EVT_CLOSE, self.OnCloseWindow)
    - def OnCloseWindow(self, event)
- Some most common evets:
  - System events: create, delete, cancel, etc
  - Paint Event: wx.EVT_PAINT
  - Focus Event: wx.FocusEvent
  - Key Event: wx.KeyEvent

# Threading

- Multi-threading is commonly used for long-running processes
- GUI operations must take place in the main thread
- A few different implementation:
  - wx.CallAfter()
    - Allows the thread to call a function on a different thread
  - wx.EVT_IDLE and wx.WakeUpIdle()
    - Manage thread communications
  - Custom: use python threading functionalities
    - import threading
    - thread = threading.Thread(target=self.do_work)
    - thread.setDaemon(True)
    - thread.start() and thread.stop()

# How to design your GUI

Design your GUI in a few steps

# How to design your GUI

- Target your audience
- Design your functions
- Design and draw your layouts
- Structure your code - OOP

# Summary

- wxPython Architecture
  - Top Level: Window/Frame
  - Container: Panels, Scroller windows, splitter windows, notebooks
  - Widgets: Add your widgets and their responsive functions
  - Layout: Sizers
  - Tools: menus, status bars, tool bars
- Design your GUI
  - Design your functions first
  - Draw your layouts
  - Use threads for long-running functions
- Refs
  - http://zetcode.com/wxpython/
  - https://docs.wxwidgets.org
  - https://wxpython.org/Phoenix/docs/html/index.html
  - https://wiki.wxpython.org/FrontPage

# More to Discover

- File system: tree control, file import/export
- Grid control: dealing with data in tabular format
- Drag and Drop: clipboard, data transfer
- Graphics: images, painting, etc
- User choices: dialogs, wizard
- HTML Integration

# Questions ?

Slides available at: https://github.com/zmcddn/Presentations