

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

wx.Window

wx.Control

wx.ControlWithItem

Top Level
Widget

Containers

wx.PopupWindow

wx.ScrolledWindow

wx.Frame

wx.MDIParentFrame

wx.MDIChildFrame

wx.Dialog

wx.ScrolledWindow

wx.Panel

wx.SplitterWindow

wx.Notebook

Dynamic Widget

Static Widget

wx.ToggleButton

wx.CheckBox

wx.TextCtrl

wx.SpinCtrl

wx.ComboBox

wx.BitmapButton

wx.Slider

wx.Choice

wx.RadioButton

wx.Button

wx.ScrollBar

wx.Grid

wx.RadioButton

wx.SpinButton

wx.ListBox

wx.StaticBitmap

wx.StaticBox

wx.Gauge

wx.StaticText

wx.StaticLine

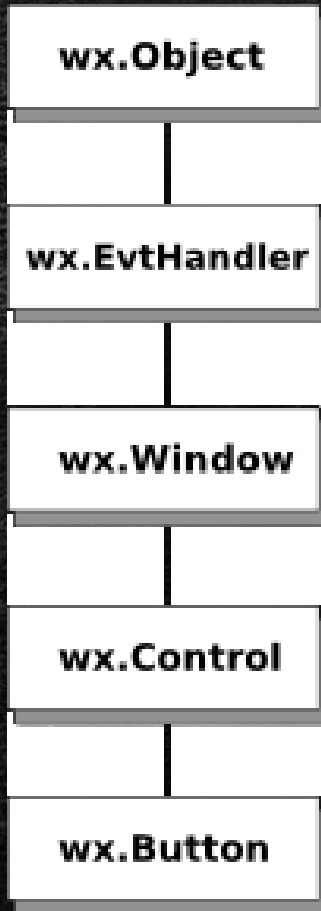
Other Widget

wx.ToolBar

wx.MenuBar

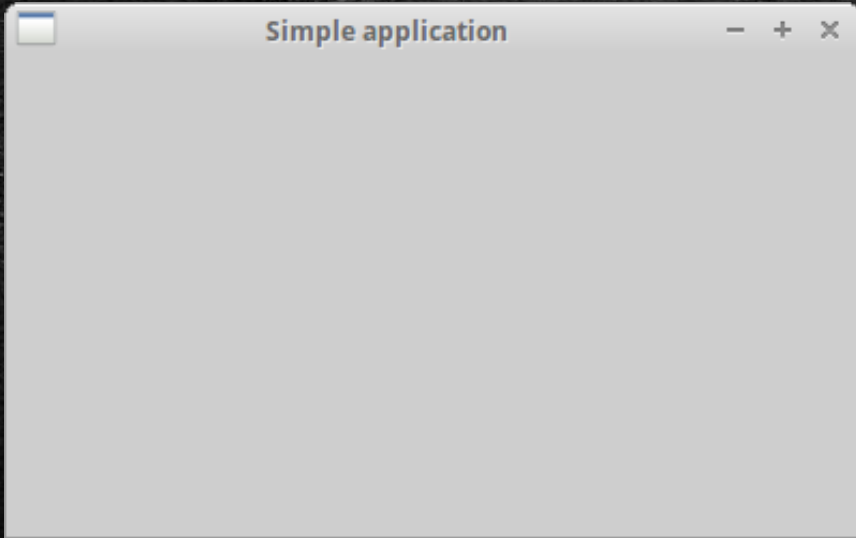
wx.StatusBar

wxPython Introduction - Inheritance



- Widgets can inherit functionality from other widgets
 - Existing classes are called base classes, parents, or ancestors.
 - The widgets that inherit are called derived widgets, child widgets or descendants.
- Button inherits 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")`

Raw Data

Plot

	Random Date	PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	2016-01-06	1	0	3	male	22.0	1	0	A/5 21171	7.25	nan	S
1	2016-02-14	2	1	1	female	38.0	1	0	PC 17599	71.2933	C85	C
2	2016-01-31	3	1	3	female	26.0	0	0	STON/O2. 3101282	7.925	nan	S
3	2016-01-17	4	1	1	female	35.0	1	0	113803	53.1	C123	S
4	2016-01-30	5	0	3	male	35.0	0	0	373450	8.05	nan	S
5	2016-02-08	6	0	3	male	nan	0	0	330877	8.4583	nan	Q
6	2016-02-15	7	0	1	male	54.0	0	0	17463	51.8625	E46	S
7	2016-01-04	8	0	3	male	2.0	3	1	349909	21.075	nan	S
8	2016-02-06	9	1	3	female	27.0	0	2	347742	11.1333	nan	S
9	2016-01-13	10	1	2	female	14.0	1	0	237736	30.0708	nan	C
10	2016-01-30	11	1	3	female	4.0	1	1	PP 9549	16.7	G6	S
11	2016-01-22	12	1	1	female	58.0	0	0	113783	26.55	C103	S
12	2016-01-05	13	0	3	male	20.0	0	0	A/5. 2151	8.05	nan	S
13	2016-01-17	14	0	3	male	39.0	1	5	347082	31.275	nan	S
14	2016-02-11	15	0	3	female	14.0	0	0	350406	7.8542	nan	S
15	2016-01-21	16	1	2	female	55.0	0	0	248706	16.0	nan	S
16	2016-01-11	17	0	3	male	2.0	4	1	382652	29.125	nan	Q
17	2016-01-11	18	1	2	male	nan	0	0	244273	13.0	nan	S
18	2016-01-12	19	0	3	female	31.0	1	0	345763	18.0	nan	S
19	2016-01-24	20	1	3	female	nan	0	0	2649	7.225	nan	C
20	2016-01-06	21	0	2	male	35.0	0	0	239855	26.0	nan	S
21	2016-02-19	22	1	2	male	34.0	0	0	248699	13.0	D56	S
22	2016-02-14	23	1	3	female	15.0	0	0	330923	8.0292	nan	Q
23	2016-01-20	24	1	1	male	28.0	0	0	113788	35.5	A6	S
24	2016-01-17	25	0	3	female	8.0	3	1	349909	21.075	nan	S
25	2016-02-19	26	1	3	female	38.0	1	5	347077	31.3875	nan	S
26	2016-02-06	27	0	3	male	nan	0	0	2631	7.225	nan	C
27	2016-01-18	28	0	1	male	19.0	3	2	19950	263.0	C23 C25 C27	S
28	2016-01-24	29	1	3	female	nan	0	0	330959	7.8792	nan	Q
29	2016-01-27	30	0	3	male	nan	0	0	349216	7.8958	nan	S
30	2016-02-04	31	0	1	male	40.0	0	0	PC 17601	27.7208	nan	C
31	2016-01-04	32	1	1	female	nan	1	0	PC 17569	146.5208	B78	C
32	2016-01-08	33	1	3	female	nan	0	0	335677	7.75	nan	Q
33	2016-02-02	34	0	2	male	66.0	0	0	C.A. 24579	10.5	nan	S
34	2016-02-15	35	0	1	male	28.0	1	0	PC 17604	82.1708	nan	C
35	2016-02-18	36	0	1	male	42.0	1	0	113789	52.0	nan	S
36	2016-02-15	37	1	3	male	nan	0	0	2677	7.2292	nan	C

Column Log

Name	Type	Non Null	Null	Non Null Percentage
Random Date	object	891	0	100.00%
PassengerId	int64	891	0	100.00%
Survived	int64	891	0	100.00%
Pclass	int64	891	0	100.00%
Name	object	891	0	100.00%
Sex	object	891	0	100.00%
Age	float64	714	177	80.13%
SibSp	int64	891	0	100.00%
Parch	int64	891	0	100.00%
Ticket	object	891	0	100.00%
Fare	float64	891	0	100.00%
Cabin	object	204	687	22.90%
Embarked	object	889	2	99.78%

Rows: 891

Columns: 12

Memory Usage: 353.27 KB

Show Bottom Panel

Hide Right Panel

Menu Bar, Toolbar, Status Bar

How does it 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)`

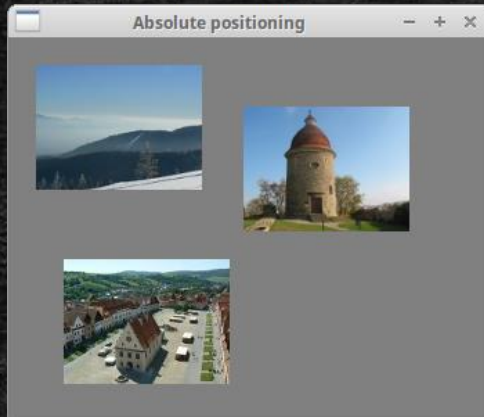
Layout

How your GUI presents

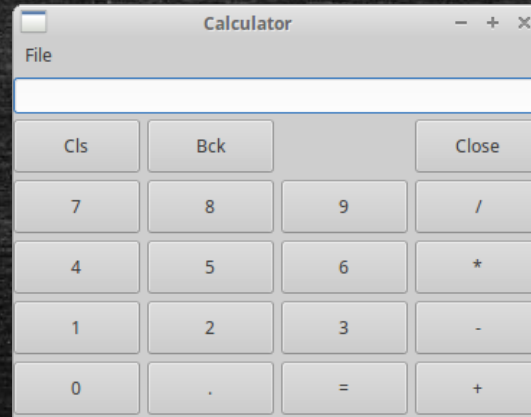
Layout

- 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

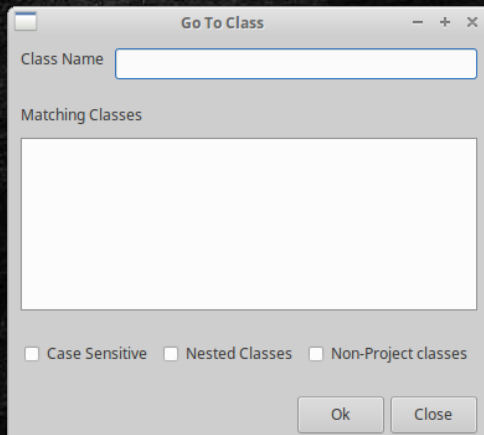
Layout - Sizers



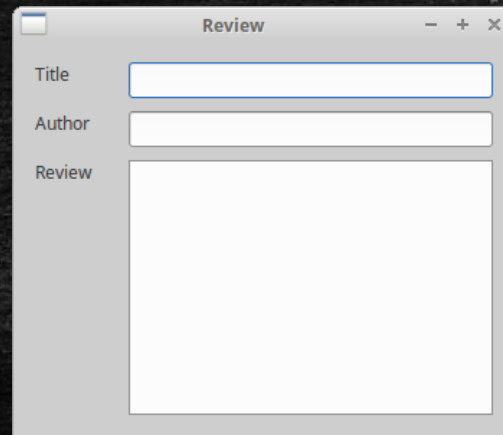
Absolute Positioning



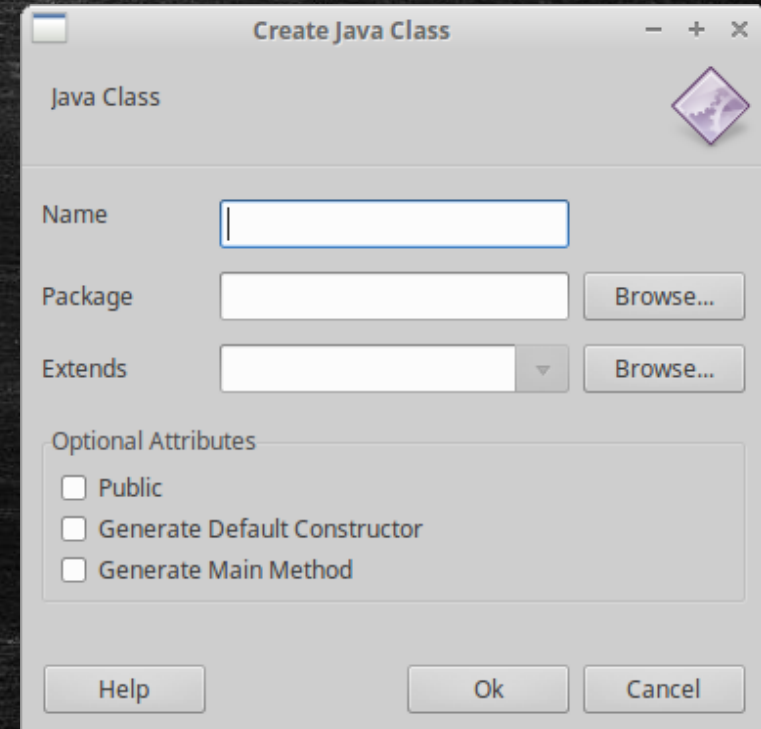
GridSizer



Box Sizers

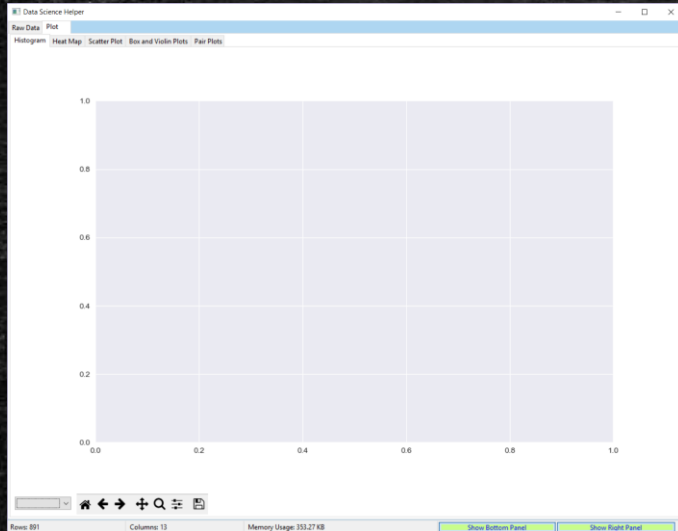


FlexGridSizer



GridBagSizer
StaticBoxSizer

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)
```

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


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>