wxPython
GUI toolkit

tutorialspoint
SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia
https://twitter.com/tutorialspoint
About the Tutorial

wxPython is a blend of wxWidgets and Python programming library. This introductory tutorial provides the basics of GUI programming and helps you create desktop GUI applications.

Audience

This tutorial is designed for software programmers who are keen on learning how to develop GUI applications for the desktop.

Prerequisites

You should have a basic understanding of computer programming terminologies. A basic understanding of Python and any of the programming languages is a plus.

Disclaimer & Copyright

© Copyright 2015 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com.
# Table of Contents

About the Tutorial......................................................................................................................................................... i
Audience .......................................................................................................................................................................... i
Prerequisites ................................................................................................................................................................. i
Disclaimer & Copyright................................................................................................................................................... i
Table of Contents.......................................................................................................................................................... ii

1. WXPYTHON – INTRODUCTION ................................................................................................................................. 1

2. WXPYTHON – ENVIRONMENT .................................................................................................................................. 2
   Windows ........................................................................................................................................................................ 2
   Linux ........................................................................................................................................................................... 2
   MacOS ....................................................................................................................................................................... 2

3. WXPYTHON – HELLO WORLD ................................................................................................................................... 3

4. WXPYTHON – FRAME CLASS ..................................................................................................................................... 5
   Window Style Constants ........................................................................................................................................... 5
   wx.Frame Class Member Functions .......................................................................................................................... 6
   wx.Frame event binders ........................................................................................................................................... 6

5. WXPYTHON – PANEL CLASS ..................................................................................................................................... 7

6. WXPYTHON – GUI BUILDER TOOLS ........................................................................................................................ 8

7. WXPYTHON – MAJOR CLASSES ................................................................................................................................. 12

8. WXPYTHON – EVENT HANDLING .......................................................................................................................... 15

9. WXPYTHON – LAYOUT MANAGEMENT .................................................................................................................. 20

10. WXPYTHON – BOXSIZER ............................................................................................................................................ 21
11. WXPYTHON – GRIDSIZER .............................................................26
12. WXPYTHON – FLEXIGRIDSIZER ..................................................29
13. WXPYTHON – GRIDBAGSIZER ......................................................32
14. WXPYTHON – STATICBOXSIZER .................................................35
15. WXPYTHON – BUTTONS ..............................................................38
16. WXPYTHON – STATICTEXT CLASS .............................................43
17. WXPYTHON – TEXTCTRL CLASS .................................................47
18. WXPYTHON – RADIOBUTTON & RADIOBOX ................................51
19. WXPYTHON – CHECKBOX CLASS ................................................55
20. WXPYTHON – COMBOBOX & CHOICE CLASS ..............................57
21. WXPYTHON – GAUGE CLASS .......................................................61
22. WXPYTHON – SLIDER CLASS ......................................................64
23. WXPYTHON – MENU ITEM, MENU & MENUBAR ............................67
24. WXPYTHON – TOOLBAR CLASS ................................................72
25. WXPYTHON – DIALOG CLASS .....................................................76
   MessageDialog ...........................................................................77
   wx.TextEntryDialog ..................................................................79
   wx.FileDialog Class .................................................................82
   wx.FontDialog Class .................................................................86
26. WXPYTHON – NOTEBOOK CLASS ..............................................89
27. WXPython – Dockable Windows ................................................................. 93

28. WXPython – Multiple Document Interface .............................................. 96

29. WXPython – SplitterWindow Class .......................................................... 98

30. WXPython – Drawing API ................................................................. 101
   wx.Colour Class ....................................................................................... 101
   wx.Pen Class .......................................................................................... 102
   wx.Brush Class ....................................................................................... 102

31. WXPython – HTMLWindow Class ............................................................ 105

32. WXPython – ListBox & ListCtrl Class ....................................................... 107

33. WXPython – Drag and Drop ................................................................. 113
wxPython is a Python wrapper for wxWidgets (which is written in C++), a popular cross-platform GUI toolkit. Developed by Robin Dunn along with Harri Pasanen, wxPython is implemented as a Python extension module.

Just like wxWidgets, wxPython is also a free software. It can be downloaded from the official website http://wxpython.org. Binaries and source code for many operating system platforms are available for download on this site.

Principal modules in wxPython API include a core module. It consists of wxObject class, which is the base for all classes in the API. Control module contains all the widgets used in GUI application development. For example, wx.Button, wx.StaticText (analogous to a label), wx.TextCtrl (editable text control), etc.

wxPython API has GDI (Graphics Device Interface) module. It is a set of classes used for drawing on widgets. Classes like font, color, brush, etc. are a part of it. All the container window classes are defined in Windows module.

Official website of wxPython also hosts Project Phoenix – a new implementation of wxPython for Python 3.*. It focuses on improving speed, maintainability, and extensibility. The project began in 2012 and is still in beta stage.
2. wxPython – Environment

Windows

Prebuilt binaries for Windows OS (both 32 bit and 64 bit) are available on http://www.wxpython.org/download.php page. Latest versions of installers available are:

wxPython3.0-win32-3.0.2.0-py27.exe for 32-bit Python 2.7
wxPython3.0-win64-3.0.2.0-py27.exe for 64-bit Python 2.7

wxPython demo, samples and wxWidgets documentation is also available for download on the same page.

wxPython3.0-win32-docs-demos.exe

Linux

wxPython binaries for many Linux distros can be found in their respective repositories. Corresponding package managers will have to be used to download and install. For instance on Debian Linux, following command should be able to install wxPython.

```
sudo apt-get install python-wxgtk3.0
```

MacOS

Prebuilt binaries for MacOS in the form of disk images are available on the download page of the official website.
A simple GUI application displaying Hello World message is built using the following steps:

- Import wx module.
- Define an object of Application class.
- Create a top level window as object of wx.Frame class. Caption and size parameters are given in constructor.
- Although other controls can be added in Frame object, their layout cannot be managed. Hence, put a Panel object into the Frame.
- Add a StaticText object to display ‘Hello World’ at a desired position inside the window.
- Activate the frame window by show() method.
- Enter the main event loop of Application object.

```python
import wx

app = wx.App()
window = wx.Frame(None, title="wxPython Frame", size=(300, 200))
panel = wx.Panel(window)
label = wx.StaticText(panel, label="Hello World", pos=(100, 50))
window.Show(True)
app.MainLoop()
```

The above code produces the following output:
**wxFrame object** is the most commonly employed top level window. It is derived from **wxWindow class**. A frame is a window whose size and position can be changed by the user. It has a title bar and control buttons. If required, other components like menu bar, toolbar and status bar can be enabled. A wxFrame window can contain any frame that is not a dialog or another frame.
**wx.Frame Class** has a default constructor with no arguments. It also has an overloaded constructor with the following parameters:

```
Wx.Frame (parent, id, title, pos, size, style, name)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parent</td>
<td>Window parent. If ‘None’ is selected the object is at the top level window. If ‘None’ is not selected, the frame appears on top of the parent window</td>
</tr>
<tr>
<td>id</td>
<td>Window identifier. Usually -1 to let the identifier be generated automatically</td>
</tr>
<tr>
<td>Title</td>
<td>Caption to appear in the title bar</td>
</tr>
<tr>
<td>Pos</td>
<td>The starting position of the frame. If not given, wxDefaultPosition is as decided by OS</td>
</tr>
<tr>
<td>Size</td>
<td>Dimensions of the window. wxDefaultSize is decided by OS</td>
</tr>
<tr>
<td>style</td>
<td>Appearance of the window controlled by style constants</td>
</tr>
<tr>
<td>name</td>
<td>The internal name of object</td>
</tr>
</tbody>
</table>

### Window Style Constants

```
wx.DEFAULT_FRAME_STYLE
wx.CAPTION
wx.MINIMIZE_BOX
wx.MAXIMIZE_BOX
wx.CLOSE_BOX
wx.SYSTEM_MENU
wx.RESIZE_BORDER
```
wx.DEFAULT_FRAME_STYLE is defined as:

wx.MINIMIZE_BOX | wx.MAXIMIZE_BOX | wx.RESIZE_BORDER | wx.SYSTEM_MENU | wx.CAPTION | wx.CLOSE_BOX | wx.CLIP_CHILDREN

Example

```python
window=wx.Frame(None, -1, "Hello", pos=(10,10), size=(300,200), style= wx.DEFAULT_FRAME_STYLE, name="frame")
```

**wx.Frame Class Member Functions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CreateStatusBar()</td>
<td>Creates the status bar at bottom of the window</td>
</tr>
<tr>
<td>CreateToolBar()</td>
<td>Creates the toolbar at the top or left of the window</td>
</tr>
<tr>
<td>GetMenuBar()</td>
<td>Gets reference to menu bar</td>
</tr>
<tr>
<td>GetStatusBar()</td>
<td>Gets reference to statusbar</td>
</tr>
<tr>
<td>SetMenuBar()</td>
<td>Displays the menu bar object in the frame</td>
</tr>
<tr>
<td>setStatusBar()</td>
<td>Associates the status bar object to the frame</td>
</tr>
<tr>
<td>SetToolBar()</td>
<td>Associates a toolbar object to the frame</td>
</tr>
<tr>
<td>SetStatusText()</td>
<td>Displays text on the status bar</td>
</tr>
<tr>
<td>Create()</td>
<td>Creates a frame with provided parameters</td>
</tr>
<tr>
<td>Centre()</td>
<td>Places the frame at the center of display</td>
</tr>
<tr>
<td>SetPosition()</td>
<td>Places the frame at given screen coordinates</td>
</tr>
<tr>
<td>SetSize()</td>
<td>Resizes the frame to given dimensions</td>
</tr>
<tr>
<td>SetTitle()</td>
<td>Inserts the given text in the title bar</td>
</tr>
</tbody>
</table>
**wx.Frame event binders**

<table>
<thead>
<tr>
<th>Event</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVT_CLOSE</td>
<td>When the frame is being closed by the user clicking the close button or programmatically.</td>
</tr>
<tr>
<td>EVT_MENU_OPEN</td>
<td>When a menu is about to be opened.</td>
</tr>
<tr>
<td>EVT_MENU_CLOSE</td>
<td>When a menu has just been closed.</td>
</tr>
<tr>
<td>EVT_MENU_HIGHLIGHT</td>
<td>When the menu item with the specified id has been highlighted.</td>
</tr>
</tbody>
</table>
Widgets such as button, text box, etc. are placed on a panel window. **wx.Panel class** is usually put inside a wxFrame object. This class is also inherited from wxWindow class.

Although controls can be manually placed on panel by specifying the position in screen coordinates, it is recommended to use a suitable layout scheme, called **sizer** in wxPython, to have better control over the placement and address the resizing issue.

In **wxPanel constructor**, the parent parameter is the wx.Frame object in which the panel is to be placed. Default value of id parameter is wx.ID_ANY, whereas the default style parameter is wxTAB_TRAVERSAL.

wxPython API has the following sizers, using which controls are added into a panel object:

<table>
<thead>
<tr>
<th>Sizer Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wx.BoxSizer</td>
<td>Widgets are arranged in a vertical or horizontal box</td>
</tr>
<tr>
<td>wx.StaticBoxSizer</td>
<td>Adds a staticbox around the sizer</td>
</tr>
<tr>
<td>wx.GridSizer</td>
<td>One control each added in equal sized cells of a grid</td>
</tr>
<tr>
<td>wx.FlexGridSizer</td>
<td>Control added in cell grid can occupy more than one cell</td>
</tr>
<tr>
<td>wx.GridBagSizer</td>
<td>Controls explicitly positioned in a grid and spanning over more than one row and/or column</td>
</tr>
</tbody>
</table>

Sizer object is applied as the layout manager of the panel using SetSizer() method of wxPanel class.

```
x.Panel.SetSizer(wx.???Sizer())
```

Panel object in turn is added to the top level frame.
Creating a good looking GUI by manual coding can be tedious. A visual GUI designer tool is always handy. Many GUI development IDEs targeted at wxPython are available. Following are some of them:

- wxFormBuilder
- wxDesigner
- wxGlade
- BoaConstructor
- gui2py

wxFormBuilder is an open source, cross-platform WYSIWYG GUI builder that can translate the wxWidget GUI design into C++, Python, PHP or XML format. A brief introduction to usage of wxFormBuilder is given here.

First of all the latest version of wxFormBuilder needs to be downloaded and installed from http://sourceforge.net/projects/wxformbuilder/. On opening the application, a new project with blank grey area at the center appears.

Give a suitable name to the project and choose Python as code generation language. This is done in the Object properties window as shown in the following image:
Then from ‘Forms’ tab of components palette, choose Frame.

Add a vertical wxBoxSizer from ‘Layouts’ tab.
Add necessary controls in the Box with suitable captions. Here, a StaticText (label), two TextCtrl objects (text boxes) and a wxButton object are added. The frame looks like the following image:

Enable Expand and Stretch on these three controls. In the object properties for wxButton object, assign a function findsquare() to OnButtonClick event.
Save the project and press F8 to generate Python code for developed GUI. Let the generated file be named as Demo.py

In the executable Python script, import demo.py and define FindSquare() function. Declare Application object and start a main event loop. Following is the executable code:

```python
import wx

# import the newly created GUI file
import demo
class CalcFrame(demo.MyFrame1):
    def __init__(self, parent):
        demo.MyFrame1.__init__(self, parent)

    def FindSquare(self, event):
        num = int(self.m_textCtrl1.GetValue())
        self.m_textCtrl2.SetValue(str(num*num))

app = wx.App(False)
frame = CalcFrame(None)
frame.Show(True)
```
#start the applications  
app.MainLoop()  

The above code produces the following output:
Original wxWidgets (written in C++) is a huge class library. GUI classes from this library are ported to Python with wxPython module, which tries to mirror the original wxWidgets library as close as possible. So, wx.Frame class in wxPython acts much in the same way as wxFrame class in its C++ version.

wxObject is the base for most of the classes. An object of wxApp (wx.App in wxPython) represents the application itself. After generating the GUI, application enters in an event loop by MainLoop() method. Following diagrams depict the class hierarchy of most commonly used GUI classes included in wxPython.
Unlike a console mode application, which is executed in a sequential manner, a GUI based application is event driven. Functions or methods are executed in response to user’s actions like clicking a button, selecting an item from collection or mouse click, etc., called events.

Data pertaining to an event which takes place during the application’s runtime is stored as object of a subclass derived from `wx.Event`. A display control (such as Button) is the source of event of a particular type and produces an object of Event class associated to it. For instance, click of a button emits a `wx.CommandEvent`. This event data is dispatched to event handler method in the program. wxPython has many predefined event binders. An **Event binder** encapsulates relationship between a specific widget (control), its associated event type and the event handler method.

For example, to call **OnClick() method** of the program on a button’s click event, the following statement is required:

```
self.b1.Bind(EVT_BUTTON, OnClick)
```

**Bind() method** is inherited by all display objects from `wx.EVTHandler` class. `EVT_.BUTTON` here is the binder, which associates button click event to OnClick() method.

**Example**

In the following example, the MoveEvent, caused by dragging the top level window – a `wx.Frame` object in this case – is connected to **OnMove() method** using `wx.EVT_MOVE` binder. The code displays a window. If it is moved using mouse, its instantaneous coordinates are displayed on the console.

```
import wx

class Example(wx.Frame):

    def __init__(self, *args, **kw):
        super(Example, self).__init__(*args, **kw)
        self.InitUI()

    def InitUI(self):
        self.Bind(wx.EVT_MOVE, self.OnMove)
        self.SetSize((250, 180))

    def OnMove(self):
        print(f"CURRENT POSITION: ({self.GetPosition()} in mm)"

if __name__ == '__main__':
    app = wx.App()
    frame = Example(parent=None, size=(300, 220))
    frame.Show()
    app.MainLoop()
```
self.SetTitle('Move event')
self.Centre()
self.Show(True)

def OnMove(self, e):
    x, y = e.GetPosition()
    print "current window position x=%d, y=%d" % (x, y)

ex = wx.App()
Example(None)
ex.MainLoop()

The above code produces the following output:

current window position x= 562  y= 309
current window position x= 562  y= 309
current window position x= 326  y= 304
current window position x= 384  y= 240
current window position x= 173  y= 408
current window position x= 226  y= 30
current window position x= 481  y= 80
Some of the subclasses inherited from `wx.Event` are listed in the following table:

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>wxKeyEvent</code></td>
<td>Occurs when a key is pressed or released</td>
</tr>
<tr>
<td><code>wxPaintEvent</code></td>
<td>Is generated whenever contents of the window needs to be redrawn</td>
</tr>
<tr>
<td><code>wxMouseEvent</code></td>
<td>Contains data about any event due to mouse activity like mouse button pressed or dragged</td>
</tr>
<tr>
<td><code>wxScrollEvent</code></td>
<td>Associated with scrollable controls like <code>wxScollbar</code> and <code>wxSlider</code></td>
</tr>
<tr>
<td><code>wxCommandEvent</code></td>
<td>Contains event data originating from many widgets such as button, dialogs, clipboard, etc.</td>
</tr>
<tr>
<td><code>wxMenuEvent</code></td>
<td>Different menu-related events excluding menu command button click</td>
</tr>
<tr>
<td><code>wxColourPickerEvent</code></td>
<td><code>wxColourPickerCtrl</code> generated events</td>
</tr>
<tr>
<td><code>wxDirFileickerEvent</code></td>
<td>Events generated by <code>FileDialog</code> and <code>DirDialog</code></td>
</tr>
</tbody>
</table>

Events in `wxPython` are of two types. Basic events and Command events. A basic event stays local to the window in which it originates. Most of the `wxWidgets` generate command events. A **command event** can be propagated to window or windows, which are above the source window in class hierarchy.

**Example**

Following is a simple example of event propagation. The complete code is:

```python
import wx

class MyPanel(wx.Panel):
    def __init__(self, parent):
        super(MyPanel, self).__init__(parent)

        b=wx.Button(self, label='Btn', pos=(100,100))
        b.Bind(wx.EVT_BUTTON, self.btnclk)
        self.Bind(wx.EVT_BUTTON, self.OnButtonClicked)
```

24
def OnButtonClicked(self, e):
    print 'Panel received click event. propagated to Frame class'
    e.Skip()

def btnclk(self,e):
    print "Button received click event. propagated to Panel class"
    e.Skip()

class Example(wx.Frame):
    def __init__(self,parent):
        super(Example, self).__init__(parent)
        self.InitUI()

    def InitUI(self):
        mpnl = MyPanel(self)
        self.Bind(wx.EVT_BUTTON, self.OnButtonClicked)
        selfSetTitle('Event propagation demo')
        self.Centre()
        self.Show(True)

    def OnButtonClicked(self, e):
        print 'click event received by frame class'
        e.Skip()

ex = wx.App()
In the above code, there are two classes. **MyPanel**, a wx.Panel subclass and Example, a wx.Frame subclass which is the top level window for the program. A button is placed in the panel.

This Button object is bound to an event handler btnclk() which propagates it to parent class (MyPanel in this case). Button click generates a **CommandEvent** which can be propagated to its parent by Skip() method.

MyPanel class object also binds the received event to another handler OnButtonClicked(). This function in turn transmits to its parent, the Example class. The above code produces the following output:

```
Example(None)
ex.MainLoop()
```

Button received click event. Propagated to Panel class.
Panel received click event. Propagated to Frame class.
Click event received by frame class.