WXPYTHON GUI – LAYOUT MANAGEMENT

LAYOUT MANAGEMENT

- Layouts are containers which are **used to arrange the UI widgets** in the GUI application
- In wxpython, UI widgets will be arranged either using absolute position or sizers (layout manager)
- In wxpython, layout manager is called as **sizer**
- Wx.Sizer is a base class for all the sizer sub classes
- Examples:
  - BoxSizer
  - Grid Sizer
  - FlexiGridSizer
  - GridBagSizer
  - StaticBoxSizer, etc,…

PROBLEMS OF ABSOLUTE POSITIONING

- The position of the widget does not change **if the window is resized**
- The UI appearance might not be uniform on different display devices with different resolutions
- Modification in the layout is difficult as it may need redesigning the entire form.

DESCRIPTION OF LAYOUT MANAGER

S.N	Layout	Description
1.	BoxSizer	It arranges the widgets either in horizontal or vertical side
2.	GridSizer	It arranges the widgets in 2D grid in the left to right and top to bottom order
3.	FlexiGridSizer	• It also has a two dimensional grid. • However it provides little more flexibility in arranging the widgets
4.	GridBagSizer	• It also has a two dimensional grid. • It offers more enhancements than FlexiGridSizer. • Widgets can be added in a specific cell within a grid
5.	StaticBoxSizer	Grouping of widgets with caption (puts a BoxSizer into a static box)

Common Methods of Sizer

1. Add()

- It is an instance method of sizer
- This method is **used to add one or more number of widgets to a particular sizer** (layout manager)
- This method takes 4 arguments, where 1st argument is widget and remaining arguments are optional
- Return type: Any

1. BoxSizer

- It is most widely used sizer in wxpython
- Here UI widgets are arranged in horizontal (row side) or vertical (column side)
- It's layout is identified by the orientation value wither wx.VERTICAL or wx.HORIZONTAL

Creation

```
bx=wx.BoxSizer(integer orientation)

bx.Add(widget, integer proportion=0, integer flag=0, integer
border=0)
```

Where,

1. Proportion

- This is the **optional property**
- This is **0** by default.
- **0** means the **size of the widget is unchangeable**
- Any value except 0 will change the size of the widget relative to value in other widget.

2. Flag

- This is the **optional property**
- It is set of flags which are used to change certain things about the sizer.

Alignment Flags

- wx.ALIGN_TOP
- wx.ALIGN_BOTTOM
- wx.ALIGN_LEFT
- wx.ALIGN_RIGHT
- wx.ALIGN_CENTER_VERTICAL
- wx.ALIGN_CENTER_HORIZONTAL

Border Flags

- wx.TOP
- wx.BOTTOM
- wx.LEFT
- wx.RIGHT
- wx.ALL

Behavior Flags

wx.Expand

- This will expand to fill the space provided to it.

wx.Shaped

- It is similar to wx.Expand but it maintains the item's aspect ratio

wx.Fixed_MINSIZE

- This won't allow the item become smaller than its initial minimum size

wx.RESERVE_SPACE_EVEN_IF_HIDDEN

- It does not allow the sizer to reclaim an item's space, when it is hidden

3. Border

- This is the **optional property**
- The amount of the padding to be used between widgets
- For Example. **wx.ALL** will apply padding on all sides.

I. EXAMPLE OF BOX SIZER – VERTICAL ARRANGEMENTS

Language	:	Python 3
Editor	:	VSC Editor
OS	:	Windows 10
GUI Framework	:	wxPython

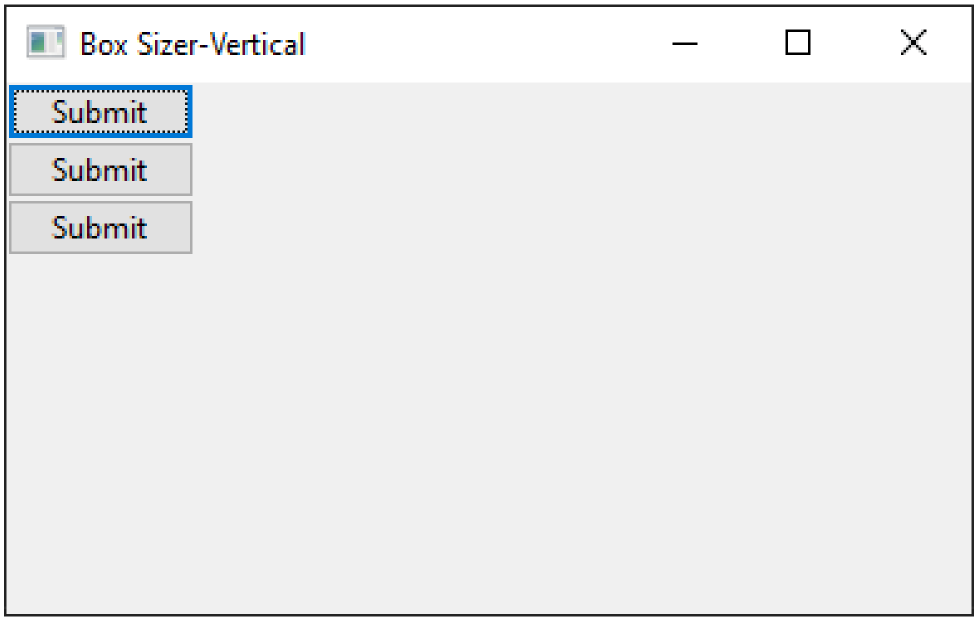
SOURCE CODE

```
import wx
# create an object for application class
obj=wx.App()
r=wx.Frame(None, title="Box Sizer-Vertical")
# create a panel
pl=wx.Panel(r)
# CREATE A BOX SIZER (LAYOUT MANAGER)
bx=wx.BoxSizer(wx.VERTICAL)
# create a button 1
b1=wx.Button(pl, label="Submit")
# create a button 2
b2=wx.Button(pl, label="Submit")
# create a button 3
b3=wx.Button(pl, label="Submit")
# add buttons 1,2 3
bx.Add(b1)
bx.Add(b2)
bx.Add(b3)
# add sizer to panel
pl.SetSizer(bx)
```

Here widgets are arranged **in vertical side (column side)**

```
# display the window
r.Show()
# run the application
obj.MainLoop()
```

2. OUTPUT



II. EXAMPLE OF BOX SIZER – HORIZONTAL ARRANGEMENTS

Language	:	Python 3
Editor	:	VSC Editor
OS	:	Windows 10
GUI Framework	:	wxPython

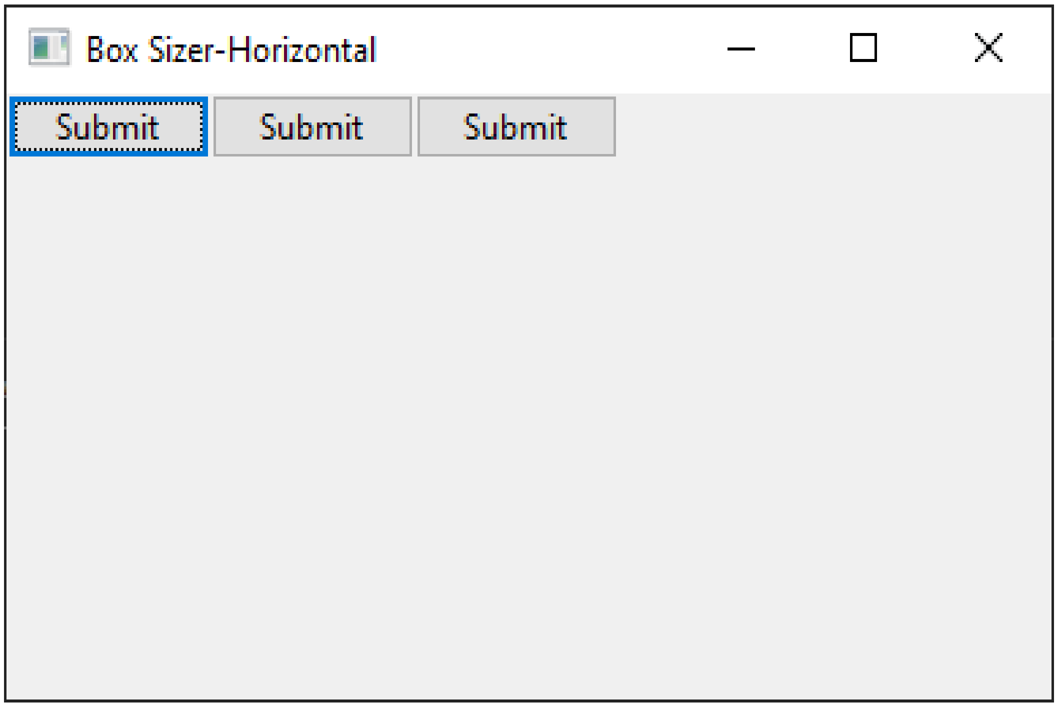
SOURCE CODE

```
import wx
# create an object for application class
obj=wx.App()
r=wx.Frame(None, title="Box Sizer-Horizontal")
# create a panel
pl=wx.Panel(r)
# CREATE A BOX SIZER (LAYOUT MANAGER)
bx=wx.BoxSizer(wx.HORIZONTAL)
# create a button 1
b1=wx.Button(pl, label="Submit")
# create a button 2
b2=wx.Button(pl, label="Submit")
# create a button 3
b3=wx.Button(pl, label="Submit")
# add buttons 1,2 3
bx.Add(b1)
bx.Add(b2)
bx.Add(b3)
# add sizer to panel
pl.SetSizer(bx)
```

Here widgets are arranged **in horizontal side (row side)**

```
# display the window
r.Show()
# run the application
obj.MainLoop()
```

2. OUTPUT



III. EXAMPLE OF BOX SIZER WITH OPTIONS

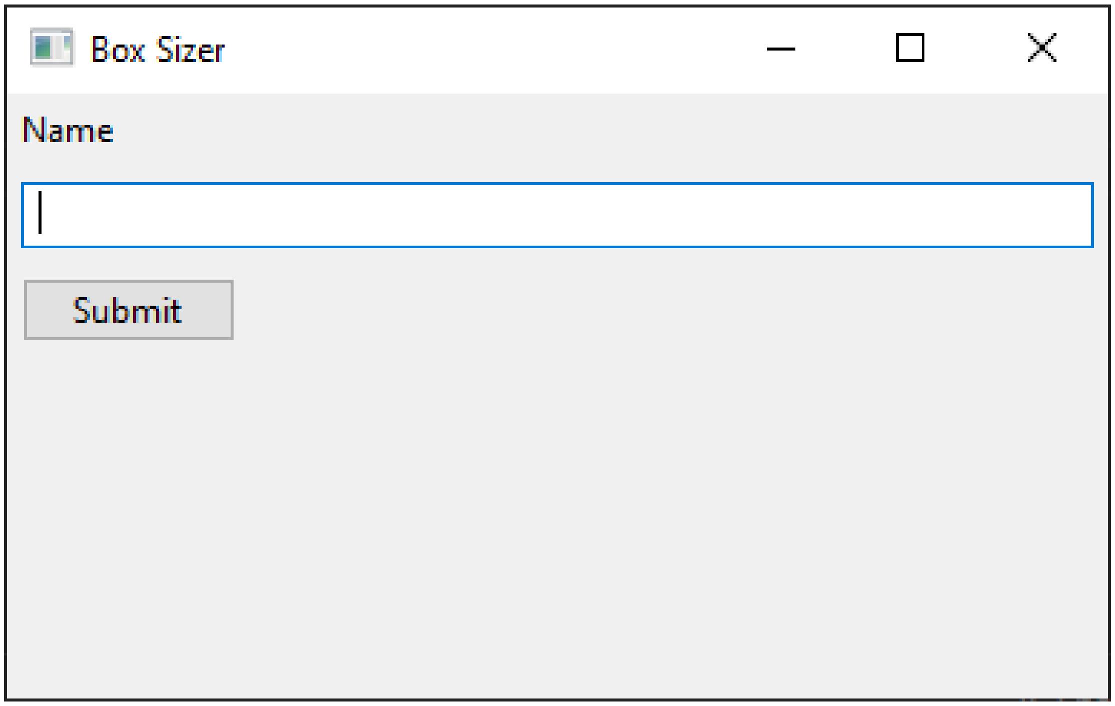
Language	:	Python 3
Editor	:	VSC Editor
OS	:	Windows 10
GUI Framework	:	wxPython

SOURCE CODE

```
import wx
# create an object for application class
obj=wx.App()
r=wx.Frame(None, title="Box Sizer")
# create a panel
pl=wx.Panel(r)
# create a box sizer with vertical
bx=wx.BoxSizer(wx.VERTICAL)
# create a label
lb=wx.StaticText(pl, label="Name")
# create a text box
tt=wx.TextCtrl(pl)
# create a button
bt=wx.Button(pl, label="Submit")
# add label, text box and button to box sizer
bx.Add(lb,0,flag=wx.ALL, border=5)
bx.Add(tt,0,flag=wx.ALL | wx.EXPAND, border=5)
bx.Add(bt,0,flag=wx.ALL, border=5)
# add sizer to panel
pl.SetSizer(bx)
```

```
# display the window
r.Show()
# run the application
obj.MainLoop()
```

2. OUTPUT


Box Sizer
Name
Submit