Department of Computer Science and Engineering
PES University, Bangalore, India

Lecture Notes
Python for Computational Problem Solving UE25CS151A

Lecture 74,75,76,77

Introduction to Graphical User Interface with wxPython

By,
Prof. Kundhavai K R,
Assistant Professor
Dept. of CSE, PESU

Verified by,
PCPS Team –
2025

Ack – Teaching Assistants – Adithya Jeyaramsankar, Trishita Umapathi

Introduction

A **Graphical User Interface (GUI)** allows users to interact with a computer program using windows, icons, buttons, and menus, instead of typing commands in a terminal. GUIs make applications more **user-friendly and visually appealing**, especially for beginners or non-technical users.

Why GUI?

- Makes computer interaction more user-friendly and accessible to everyone.
- Provides clear visual elements such as buttons, menus, and icons to guide users.
- Reduces the need for memorizing text commands or technical procedures.
- Improves efficiency by allowing faster navigation and task execution.
- Offers a consistent interface that can be learned once and applied across multiple applications.

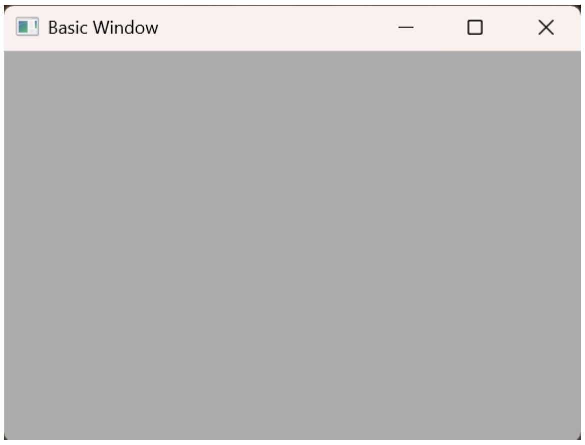
wxPython

wxPython is a Python library for creating cross-platform GUI applications. It is built on top of the **wxWidgets C++ library**, which provides **native look-and-feel** on Windows, macOS, and Linux. Unlike simpler libraries such as Tkinter, wxPython offers a **richer set of widgets** (controls) and ensures that applications look like real native apps on each operating system.

Key Points about wxPython

1. Open-source Python GUI framework.
2. Provides **native look and feel** on all operating systems.
3. Requires installation (pip install wxPython).
4. Every wxPython program has:
    - **Application(wx.App)** - starts the GUI.
    - **Frame(wx.Frame)** - the main window.
    - **Panel(wx.Panel)** - a container for widgets.
    - **Controls(widgets)** - e.g., Button, TextBox, CheckBox.
    - **Button (wx.Button)** - a clickable control used to trigger actions.
    - **Event Loop (MainLoop)** - keeps the app running.

Example 1 : Basic Window with title and geometry

import wx
 	
app = wx.App()
 	
frame = wx.Frame(None, title="Basic Window", size=(400, 300))
 	
frame.Show()
 	
app.MainLoop()
	

wx.App()

- The application object that starts every wxPython program.
- Initializes the GUI toolkit and prepares it to run.
- Manages all windows and events in the application.
- Without App, no window can be created or displayed.

wx.Frame()

- The main window of the application.
- None as first argument represents, this frame window is top level and has no parent window
- Can directly hold controls, but commonly used with a Panel for better layout
- By default it is hidden; use .Show() to make it visible.

MainLoop()

- A function that continuously loops and displays the window until it is closed.
- Waits for events (mouse clicks, key presses, etc.) and responds to them.
- Keeps the program alive; without it the window will close immediately.
- Ends only when the user closes the window or exits the program.

wx.Panel

- A panel is a window container usually placed inside a wx.Frame.
- Used to hold and group other controls like buttons, text boxes, etc.
- Helps manage layout, focus, and tab navigation within a window.
- Handles background painting automatically.
- Recommended instead of placing controls directly on the frame.

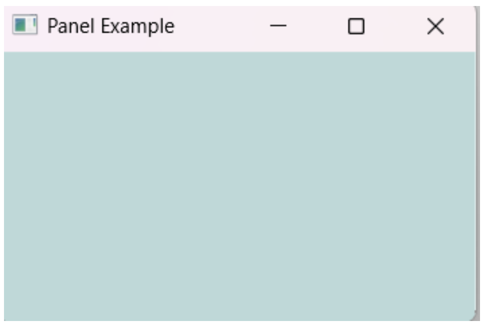
Syntax:

W = wx.Panel(parent, options)

- parent – parent window (usually a wx.Frame)
- options – used to set position, size, style, or other properties of the panel, written as comma-separated key-value pairs

Example 2 : Simple Panel Example

```
import wx

app = wx.App(False)

frame = wx.Frame(None, title="Panel Example", size=(300,
200))

panel = wx.Panel(frame)

panel.SetBackgroundColour("light blue")

frame.Show()

app.MainLoop()
```



Difference between wx.Frame and wx.Panel:

Component	Description
wx.Frame	- Top-level window of your application.
	- Has a **title bar**, **minimize**, **maximize**, and **close** buttons.
	- Cannot be placed inside another frame or panel — it stands on its own.
	- Acts as the **main shell** of your application.
	- Usually holds a **wx.Panel** inside it.
wx.Panel	- A **container widget** placed **inside a wx.Frame** (or another panel).
	- Not a top-level window — it has **no title bar or system buttons**.
	- Used to **hold other widgets** like wx.Button, wx.StaticText, or wx.TextCtrl.
	- Often used for **layout management** and **background color customization**.

wx.Button

- A button widget that users can click to perform an action.
- Created inside a frame or panel.
- Needs a label (text on the button).
- You can bind events to the button, so when it's clicked, a function is called.

Syntax:

wx.Button(parent, id, label, pos, size, style)

- parent → the window or panel it belongs to
- label → text displayed on the button
- pos → (x, y) position
- size → width and height

Types of buttons

- Normal Button (wx.Button) → Standard text button.
- Toggle Button (wx.ToggleButton) → Two-state button (On/Off).
- Bitmap Button (wx.BitmapButton) → Button with an image/icon.

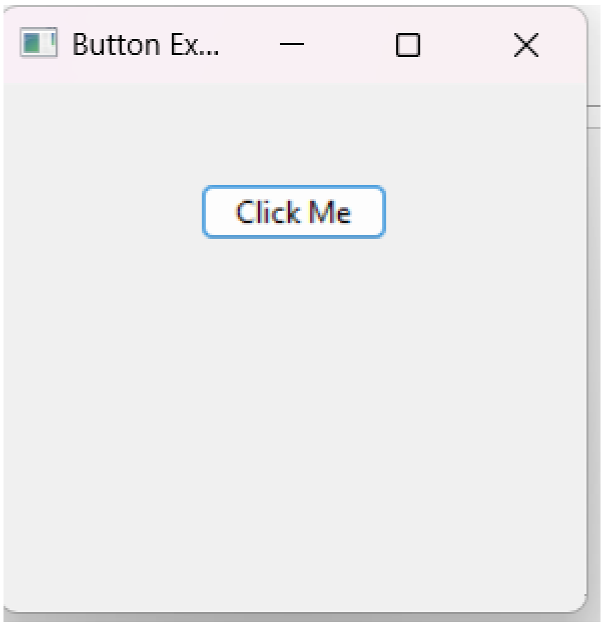
Syntax:

- Normal Button **: w**x.Button(parent, id, label, pos, size, style)
- Toggle button : wx.ToggleButton(parent, id, label, pos, size, style)
- Bitmap button : wx.BitmapButton(parent, id, bitmap, pos, size, style)

Some important methods

Class	**Method**	**Description**
wx.Button	SetLabel()	Change button text
wx.Button	GetLabel()	Get current text
wx.Button	SetDefault()	Makes button default (Enter key triggers it)
wx.ToggleButton	GetValue()	Returns toggle state (True/False)
wx.ToggleButton	SetValue()	Set state programmatically

Example 3 : Button labeled Click Me

import wx
 	
app = wx.App()
 	
frame = wx.Frame(None, title="Button Example", size=(250,150))
 	
panel = wx.Panel(frame)
 	
button = wx.Button(panel, label="Click Me", pos=(80,40))
 	
frame.Show()
 	
app.MainLoop()
	

Simple event handling with wx.Button

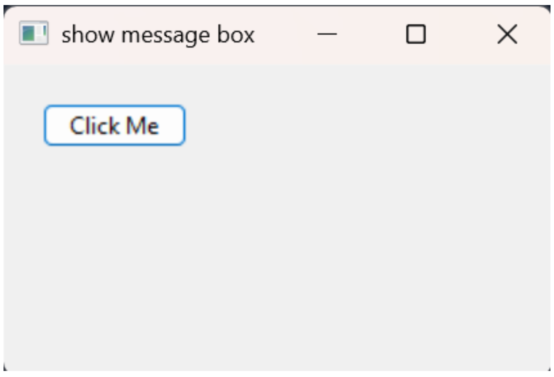
- Event handling is the process of making a program respond to user actions, such as clicking a button, pressing a key, or moving the mouse.
- In wxPython, events are signals sent when something happens in the GUI.
- To respond to an event, you bind it to an event handler function.
- The event handler contains the code that runs when the event occurs.
- For buttons, the most common event is wx.EVT_BUTTON.

  Syntax:

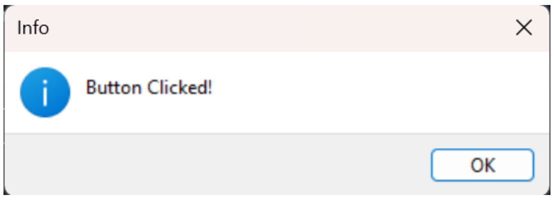
  ```
  button.Bind(event, handler_function)
  ```
- button – the wx.Button control that will trigger the event
- event – the type of event to handle (e.g., wx.EVT_BUTTON for button clicks)
- handler_function – the function to run when the event occurs

Example 4 : Display message box on click

```
import wx

def on_click(event):

    wx.MessageBox("Button Clicked!", "Info")

app = wx.App()

frame = wx.Frame(None, title="show message box",
size=(300, 200))

panel = wx.Panel(frame)

btn = wx.Button(panel, label="Click Me", pos=(20, 20))

btn.Bind(wx.EVT_BUTTON, on_click)

frame.Show()

app.MainLoop()
```



After clicking the button, a message box appears


Info

Button Clicked!

Drawing on a Panel

To draw in wxPython, we first need a **surface** — typically a **wx.Panel**. The actual drawing is done using a special object called **wx.PaintDC**, which acts as our **drawing context** or **"paintbrush"**. wx.PaintDC is used when you want to **draw shapes, lines, or text** on a window or panel. It gives you access to special **drawing tools** like **pens** and **brushes**.

Tools for wx.PaintDC

wx.Pen

- A pen is used to draw lines or the borders of shapes.
- You can set the color and thickness of the pen.

wx.Brush

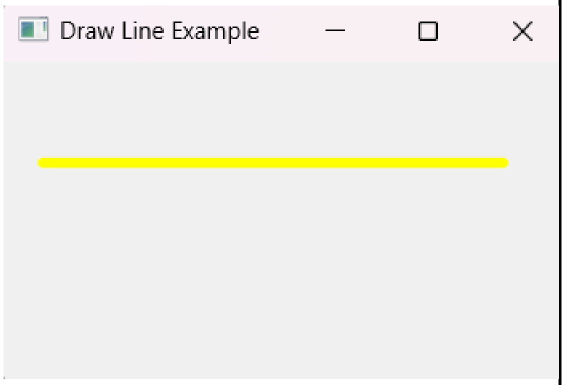
- A **brush** is used to fill the **inside of shapes** with a color.

Common Drawing Methods

- dc.DrawLine(x1, y1, x2, y2)
  → Draws a **line** between two points.
- dc.DrawRectangle(x, y, width, height)
  → Draws a **rectangle** at (x, y) with given width and height.

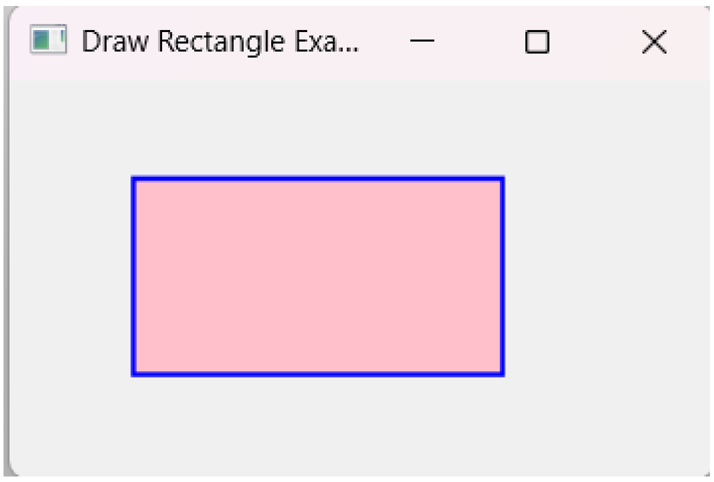
Example 5 : Drawing a line

```
import wx
app = wx.App(False)
frame = wx.Frame(None, title="Draw Line Example", size=(300, 200))
panel = wx.Panel(frame)
def on_paint(event):
   dc = wx.PaintDC(panel)
   dc.SetPen(wx.Pen("yellow", width=5))
   dc.DrawLine(20, 50, 250, 50)   # Draw a horizontal line
panel.Bind(wx.EVT_PAINT, on_paint)
frame.Show()
app.MainLoop()
```



Example 6 : Drawing a Rectangle

```
import wx
app = wx.App(False)
frame = wx.Frame(None, title="Draw Rectangle Example", size=(300, 200))
panel = wx.Panel(frame)
def on_paint(event):
   dc = wx.PaintDC(panel)
   dc.SetPen(wx.Pen("blue", width=2))
   dc.SetBrush(wx.Brush("pink"))         # Fill color
   dc.DrawRectangle(50, 40, 150, 80)
panel.Bind(wx.EVT_PAINT, on_paint)
frame.Show()
app.MainLoop()frame.Show()
app.MainLoop()
```



WxCheckBox:

- A **CheckBox** is a small square box that users can **check** or **uncheck**.
- Used when you want to let users **select one or more options**.
- In wxPython, it is created using the **wx.CheckBox()** widget.

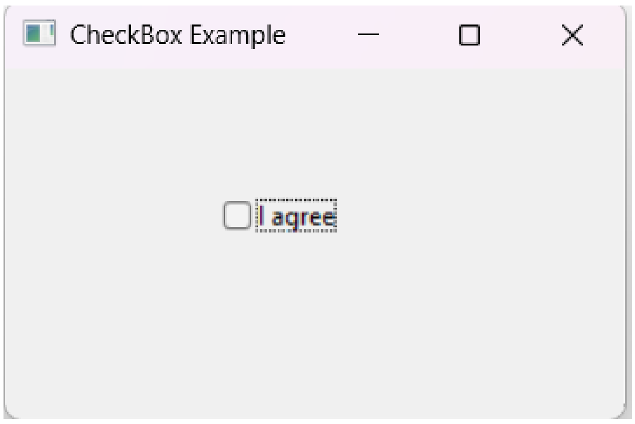
Syntax:

- wx.CheckBox(parent, id=wx.ID_ANY, label="", pos=(x, y))

- **Parameters:**
  - **Parent** - The window or panel where the checkbox appears.
  - **Id** - Widget ID (use wx.ID_ANY if not needed).
  - **Label** - The text shown next to the checkbox.
  - **Pos** - Position of the checkbox (x, y) in pixels.

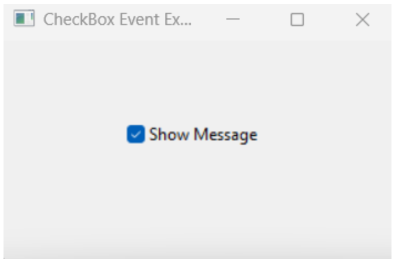
Example 7: Simple checkbox creation

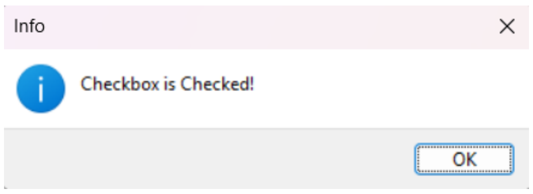
```
import wx
app = wx.App(False)
frame = wx.Frame(None, title="CheckBox Example",
size=(300, 200))
panel = wx.Panel(frame)
# Create a CheckBox
check = wx.CheckBox(panel, label="I agree",
pos=(100, 60))
frame.Show()
app.MainLoop()
```



Example 8: Simple event handling with checkbox

```
import wx
app = wx.App(False)
frame = wx.Frame(None, title="CheckBox Event Example",
size=(300, 200))
panel = wx.Panel(frame)
check = wx.CheckBox(panel, label="Show Message", pos=(90,
60))
```



```
def on_check(event):
    if check.GetValue():  # True if checked
        wx.MessageBox("Checkbox is Checked!", "Info")
    else:
        wx.MessageBox("Checkbox is Unchecked!", "Info")
check.Bind(wx.EVT_CHECKBOX, on_check)
frame.Show()
app.MainLoop()
```
	 Info
	Checkbox is Checked!

Widgets

Widgets are **GUI components** such as buttons, labels, and text boxes.

WxPython provides a wide range of widgets to **display and collect information**

Common examples:

- **wx.StaticText** → Display static text
- **wx.TextCtrl** → Take user input
- **wx.MessageDialog** → Show messages
- **wx.TextEntryDialog** → Ask for text input

wx.StaticText

Used to display text that cannot be edited by the user.

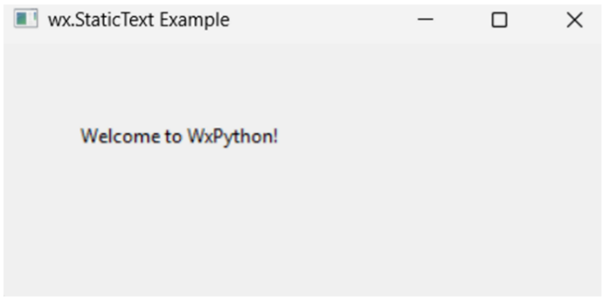
Syntax:

```
wx.StaticText(parent, id=wx.ID_ANY, label="", pos=(x, y))
```

Parameters:

- Parent - The container (like a wx.Panel or wx.Frame) where the text will appear. Used to specify which window the widget belongs to.
- Id - A unique identifier for the widget. Usually set as wx.ID_ANY if you don't need a specific ID.
- Label - The actual text displayed on the screen.
- Pos - The position of the text in pixels on the window. Written as (x, y) where (0, 0) is the top-left corner.

Example 9: StaticText Example

import wx
 	
app = wx.App(False)
 	
frame = wx.Frame(None, title="wx.StaticText Example", size=(400, 200))
 	
panel = wx.Panel(frame)
 	
text = wx.StaticText(panel, label="Welcome to WxPython!", pos=(50, 50))
 	
frame.Show()
 	
app.MainLoop()
	 wx.StaticText Example
	Welcome to WxPython!

wx.TextCtrl

Used to allow the user to enter or edit text.

Syntax:

wx.TextCtrl(parent, id=wx.ID_ANY, value="", pos=(x, y), size=(width, height), style=0)

Common Styles:

wx.TE_MULTILINE → Multiple lines of text

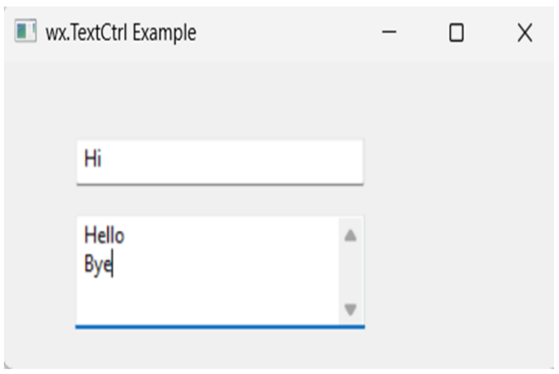
wx.TE_PASSWORD → Hide characters (password field)

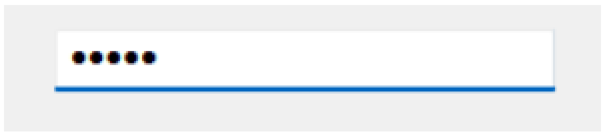
wx.TE_READONLY → Display only, not editable

Example 10 : TextCtrl Example

import wx
app = wx.App(False)
frame = wx.Frame(None, title="wx.TextCtrl Example", size=(400, 200))
panel = wx.Panel(frame)	

```
# Single-line
txt1 = wx.TextCtrl(panel, pos=(50, 40), size=(200, 25))

# Multi-line
txt2 = wx.TextCtrl(panel, pos=(50, 80), size=(200, 60),
style=wx.TE_MULTILINE)

# Create a password-style text box
password_box = wx.TextCtrl(panel, pos=(50, 60),
size=(200, 25), style=wx.TE_PASSWORD)

frame.Show()
app.MainLoop()
```





wx.MessageDialog

Used to display a message box to the user — for information, warnings, or questions.

Syntax:

wx.MessageDialog(parent, message, caption="", style=0)

Parameters:

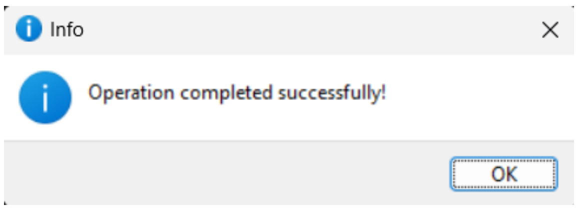
- **parent** – The window or panel that owns the dialog box.
  It decides where the dialog will appear (usually centered on this parent window).
- **message** – The main text or information you want to show to the user.
  Example: "File saved successfully!"
- **caption** – The title displayed on the top bar of the dialog box.
  Example: "Information", "Warning", "Error".
- **style** – Defines the type of buttons and icon shown in the dialog.

Common styles include:

- wx.OK → OK button only
- wx.OK | wx.CANCEL → OK and Cancel buttons
- wx.YES_NO → Yes and No buttons
- wx.ICON_INFORMATION, - Adds an information icon

- wx.ICON_WARNING - Adds a warning icon
- wx.ICON_ERROR → Adds an error icon

Example 11 : MessageBox Example

import wx
app = wx.App(False)
frame = wx.Frame(None, title="MessageDialog Example")
dlg = wx.MessageDialog(frame, "Operation completed successfully!", "Info", wx.OK | wx.ICON_INFORMATION)
dlg.ShowModal()
dlg.Destroy()
frame.Show()
app.MainLoop()
	
	
	Operation completed successfully!

Explanation:

- wx.MessageDialog → Creates a dialog box with a message and buttons.
- style=wx.YES_NO | wx.ICON_QUESTION → Adds **Yes/No buttons** and a **question icon**.
- ShowModal() → Displays the dialog and waits for the user's response.
- Based on the response, another message box is shown.

wx.TextEntryDialog

Used to get a single line of text input from the user — for example, entering a name, age, or any short response

Syntax:

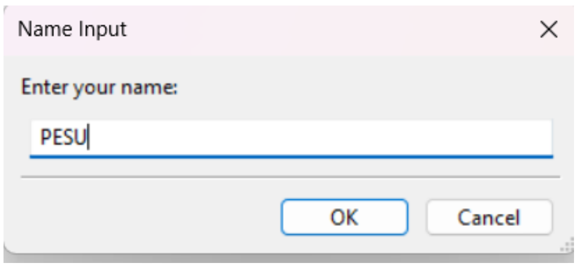
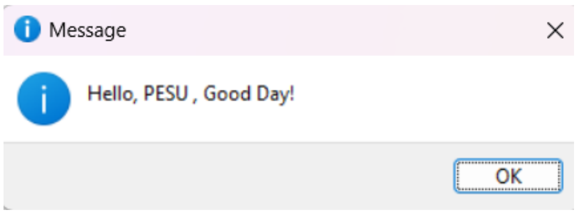
wx.TextEntryDialog(parent, message, caption, defaultValue="")

Parameters:

- parent – The window (frame or panel) on which the dialog will appear.
- message – The prompt or question shown to the user (e.g., "Enter your name:").
- caption – The title displayed on the dialog window.
- defaultValue – (Optional) The text shown by default in the input box.

Example 12 : TextEntry Example

import wx
app = wx.App()
frame = wx.Frame(None, title="Text Entry Dialog Example",	

```
size=(300, 200))
dialog = wx.TextEntryDialog(frame, "Enter your name:",
"Name Input")
if dialog.ShowModal() == wx.ID_OK:
    name = dialog.GetValue()
    wx.MessageBox(f"Hello, {name} , Good Day!")
dialog.Destroy()
frame.Show()
app.MainLoop()
```
	 Name Input
	Enter your name:
	PESU
	 Message
	Hello, PESU , Good Day!

Customizing Widgets

SetFont()

Used to change the font style, size, and weight of text in a widget (like wx.StaticText).

Syntax:

widget.SetFont(wx.Font(pointSize, family, style, weight))

Parameters:

- pointSize: Font size in points.
- family: Font family
- style: Normal / Italic / Slant
- weight: Normal / Bold / Light

SetSize()

Used to **set or change the size and position** of a widget.

Syntax:

widget.SetSize(x, y, width, height)

Parameters:

(x, y) → position on the window (top-left corner)
(width, height) → size in pixels

SetBackgroundColour()

Used to set the background color of a widget.

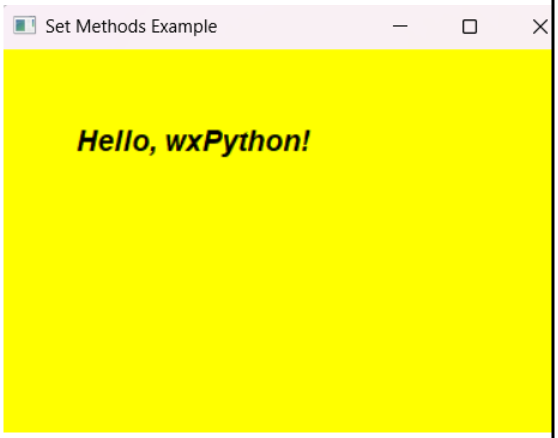
Syntax:

widget.SetBackgroundColour("light blue")

Colors:

- Named Colors (“red” , “green” etc)
- RGB tuple ((255,0,0))

Example 13 :

import wx
app = wx.App()
frame = wx.Frame(None, title="Set Methods Example", size=(400, 300))
panel = wx.Panel(frame)
text = wx.StaticText(panel, label="Hello, wxPython!", pos=(50, 50))
text.SetFont(wx.Font(14, wx.FONTFAMILY_DEFAULT, wx.FONTSTYLE_ITALIC, wx.FONTWEIGHT_BOLD))
text.SetSize(200, 40)
panel.SetBackgroundColour("yellow")
frame.Show()
app.MainLoop()
	 Hello, wxPython!

Layout Managers in wxPython

They automatically arrange widgets inside a window or panel — instead of manually setting pixel positions (pos=(x, y)).

wx.BoxSizer

Used to arrange widgets **in a single row or column**.

Purpose:

Organize items **horizontally or vertically** in a clean, flexible layout.

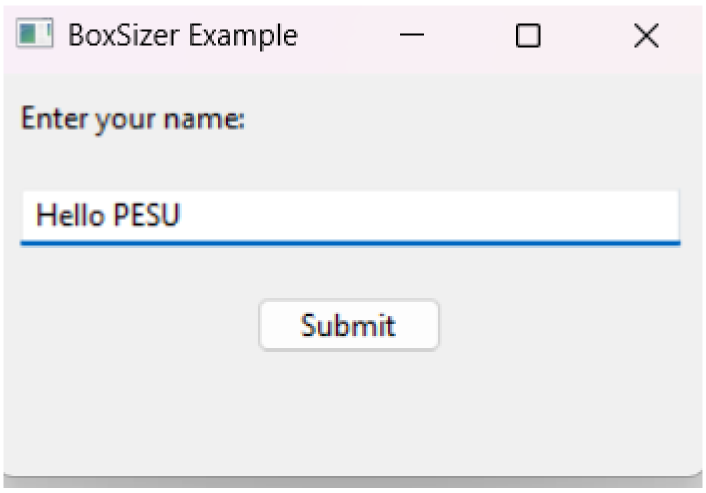
Syntax:

```
sizer = wx.BoxSizer(wx.HORIZONTAL)                    # or wx.VERTICAL
sizer.Add(widget, proportion, flag, border)
```

proportion → how much space it takes compared to others (0 = fixed size)
flag → alignment or border (e.g., wx.ALL, wx.CENTER)
border → space around widget (in pixels)

Example:

```
import wx
app = wx.App()
frame = wx.Frame(None, title="BoxSizer Example", size=(300, 200))
panel = wx.Panel(frame)
sizer = wx.BoxSizer(wx.VERTICAL)
text = wx.StaticText(panel, label="Enter your name:")
text2 = wx.TextCtrl(panel)
btn = wx.Button(panel, label="Submit")
sizer.Add(text, 0, wx.ALL, 10)
sizer.Add(text2, 0, wx.ALL | wx.EXPAND, 10)
sizer.Add(btn, 0, wx.ALL | wx.CENTER, 10)
panel.SetSizer(sizer)
frame.Show()
app.MainLoop()
```



wx.GridSizer

Used to arrange widgets in a **table-like grid** (rows × columns).

Purpose:
Organize items evenly in a **grid structure**, each cell having equal size.

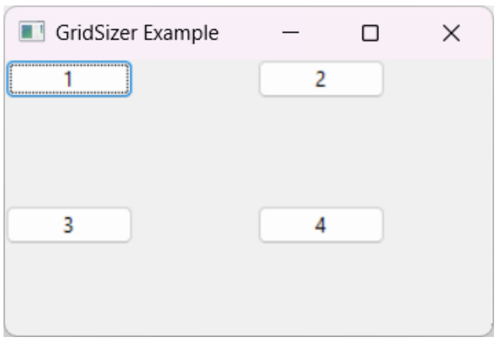
Syntax:

```
sizer = wx.GridSizer(rows, cols, vgap, hgap)
sizer.Add(widget, flag, border)
```

rows, cols → number of rows and columns
vgap, hgap → vertical & horizontal space between cells

Example:

```
import wx

app = wx.App()
frame = wx.Frame(None, title="GridSizer Example", size=(300, 200))
panel = wx.Panel(frame)
sizer = wx.GridSizer(2, 2, 10, 10)
sizer.Add(wx.Button(panel, label="1"))
sizer.Add(wx.Button(panel, label="2"))
sizer.Add(wx.Button(panel, label="3"))
sizer.Add(wx.Button(panel, label="4"))
panel.SetSizer(sizer)
frame.Show()
app.MainLoop()
```


GridSizer Example
1
2
3
4