PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Introduction to GUI with WxPython

UE25CS151A

Department of Computer Science and Engineering

Prof. Kundhavai K R, CSE Department

GUI - WxPython

PES UNIVERSITY

Why GUI:

- GUI stands for Graphical User Interface
- Allows users to interact with computers visually using windows, buttons, icons, and menus
- Enables even non-technical users to use applications easily
- Provides an intuitive and user-friendly experience
- Reduces the learning curve and simplifies workflows
- No need to remember or type complex commands

GUI - wxPython

PES UNIVERSITY

Popular Python GUI frameworks

1. Tkinter/ttkbootstrap
2. Qt for Python: PySide2 / Qt5
3. PySimpleGUI
4. PyGUI
5. Kivy
6. **wxPython**
7. Libavg
8. PyForms
9. Wax
10. PyGTK

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

WxPython

Topics to be Covered:

1. GUI with wxPython
2. wx.Panel
3. wx.Frame
4. wx.Button
5. wx.PaintDC
6. wx.CheckBox
7. wx.StaticText
8. wx.TextCtrl
9. wx.MessageDialog
10. wx.TextEntryDialog
11. wx.SetFont()
12. wx.SetSize
13. wx.SetBackgroundColour
14. wx.BoxSizer
15. wx.GridSizer

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

PES UNIVERSITY
CELEBRATING 50 YEARS

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.

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING
GUI - wxPython

wxPython

- Open-source Python GUI toolkit based on the wxWidgets C++ library
- Cross-platform: works on Windows, macOS, and Linux
- Uses native widgets for a real native look and feel
- Simple, easy to write, and easy to understand for Python programmers

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

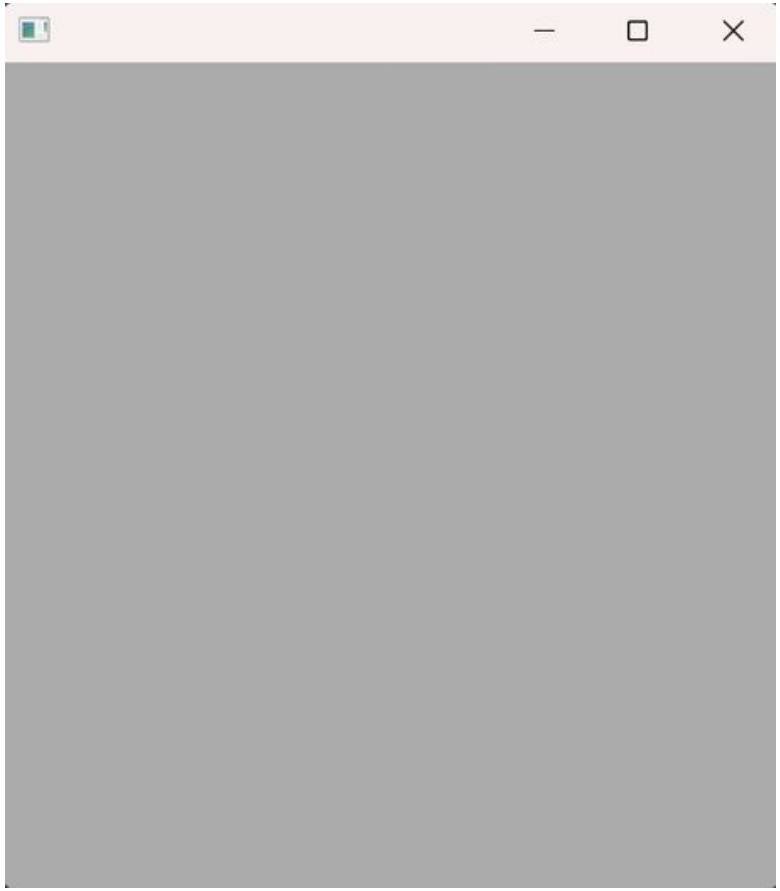
Installation command

pip install -U wxPython

Basic Window

```
import wx
app = wx.App()
frame = wx.Frame(None)
frame.Show()
app.MainLoop()
```

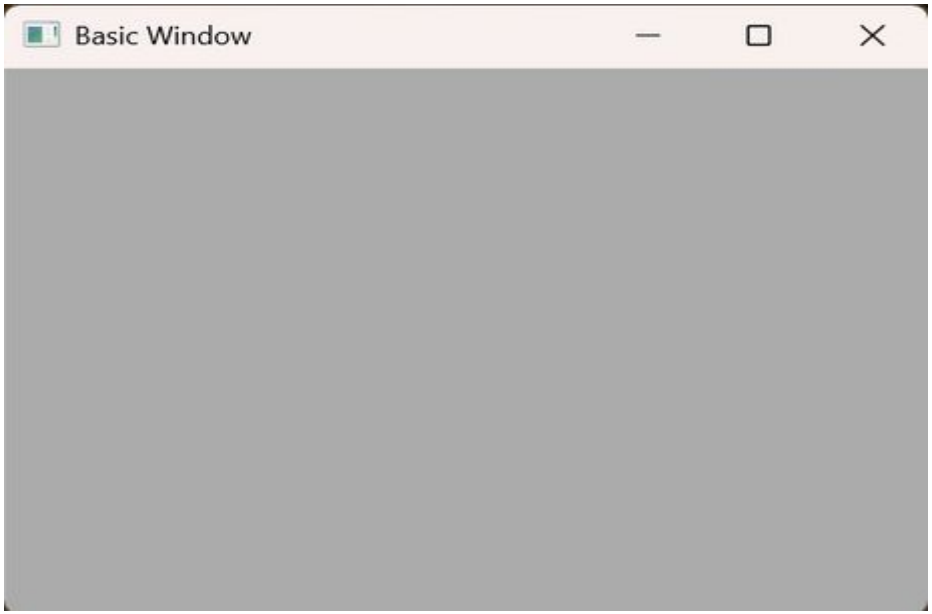
Output :



Adding title and geometry to the window

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

Output :



PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

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.
- Can directly hold controls, but commonly used with a Panel for better layout
- By default it is hidden; use .Show() to make it visible.

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

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.

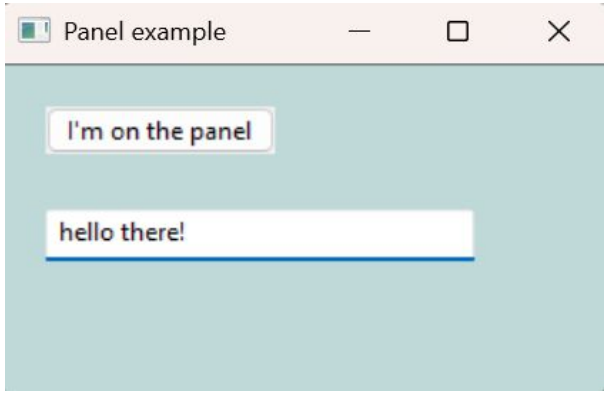
- **Syntax:**

  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

Panel example 1 - adding button and textbox on the panel:

```
import wx
app = wx.App()

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

panel = wx.Panel(frame, style=wx.SIMPLE_BORDER)
panel.SetBackgroundColour("light blue")  # Or wx.Colour(R,G,B)

btn = wx.Button(panel, label="I'm on the panel", pos=(20, 20))

# Add a text box on the panel
text = wx.TextCtrl(panel, value="", pos=(20, 70), size=(200, 25))
frame.Show()
app.MainLoop()
```



PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

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

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

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)

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

some important methods

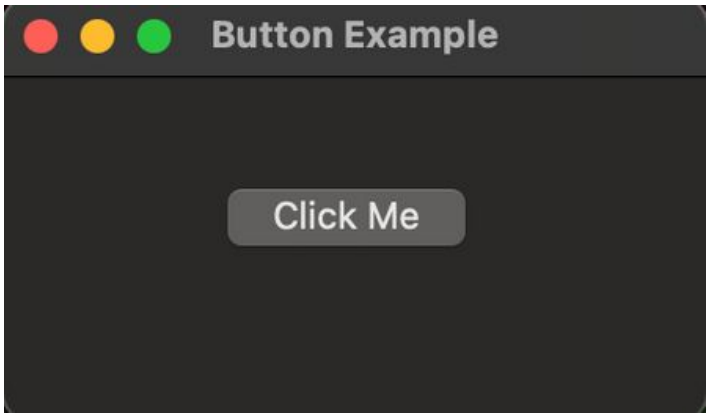
Class	Method	Description
wx.Button	SetLabel()	Change button text
wx.Button	GetLabel()	Get current text
wx.Button	SetDefault()	Makes button default (Enter key triggers it)
wx.ToggleButton	GetValue()	Returns toggle state (True/False)
wx.ToggleButton	SetValue()	Set state programmatically

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

Button example 1: 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

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

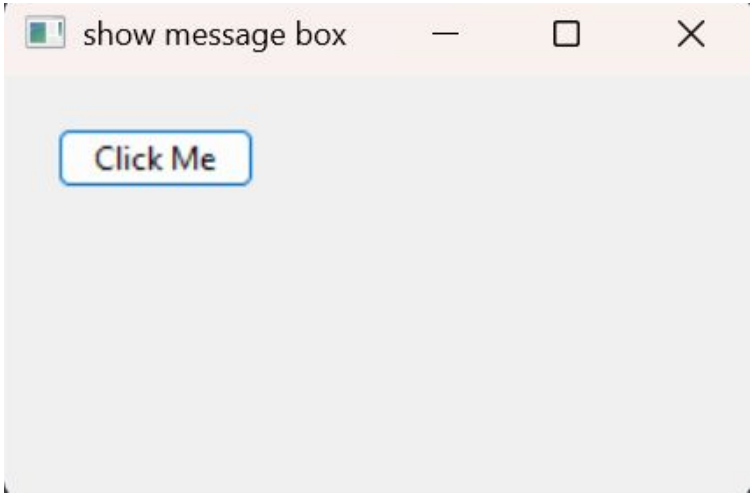
Event Handling example 1 - Showing message box on click

```
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()
```

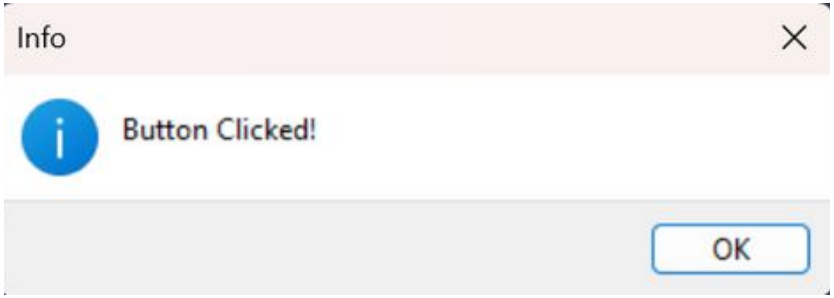
PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

GUI - wxPython

Event Handling example 1 (contd.)
Output :



After clicking the button,
a message box appears



Event Handling example 2 - changing button label on click

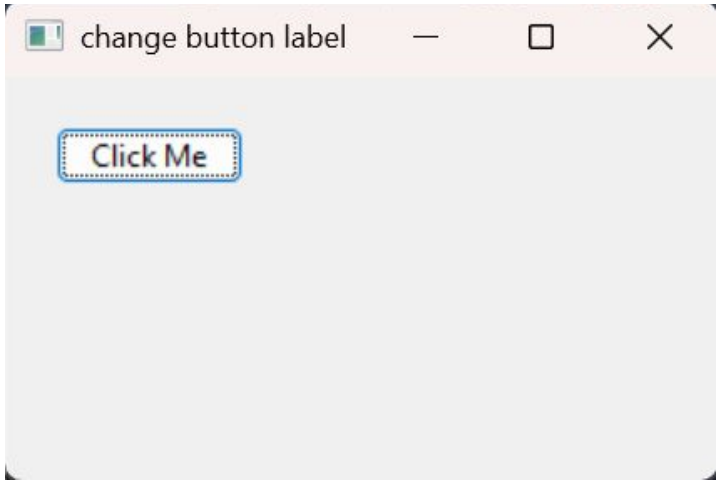
```
import wx

def change_label(event):
    btn.SetLabel("Clicked!")

app = wx.App()

frame = wx.Frame(None, title="change button label", size=(300, 200))
panel = wx.Panel(frame)

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

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

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

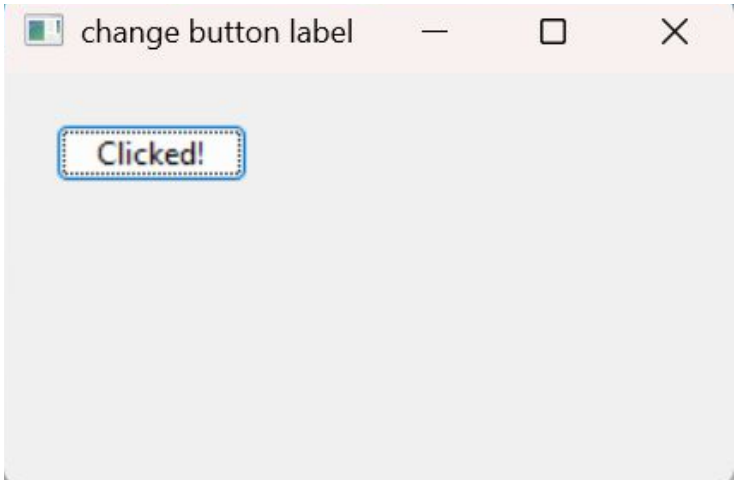
GUI - wxPython

Event Handling example 2 (contd.)

Output :



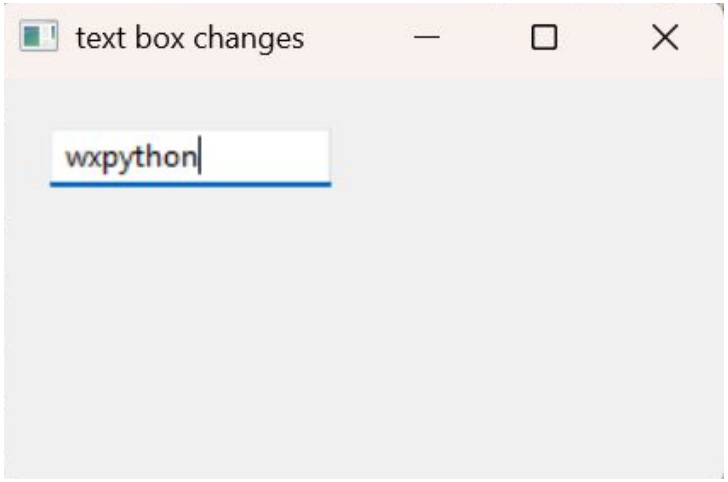
after clicking, the text on the button is changed



Event Handling example 3 - print changes entered in the text box

```
import wx

def on_text_change(event):
    print("Text changed to:", event.GetString())

app = wx.App()

frame = wx.Frame(None, title="text box changes", size=(300, 200))
panel = wx.Panel(frame)

text_box = wx.TextCtrl(panel, pos=(20, 20))
text_box.Bind(wx.EVT_TEXT, on_text_change)

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

Event Handling example 3(contd.)
Output :



detects changes in the text box

```
Text changed to: w
Text changed to: wx
Text changed to: wxp
Text changed to: wxpy
Text changed to: wxpyt
Text changed to: wxpyth
Text changed to: wxpytho
Text changed to: wxpython
```

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Tools for wx.PaintDC

- 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**.

wx.Pen

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

Example:

```
python

dc.SetPen(wx.Pen("blue", 3))
```

Creates a blue pen with 3-pixel thickness and tells the drawing context (`dc`) to use it for all outlines and lines.

wx.Brush

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

Example:

```
python

dc.SetBrush(wx.Brush("yellow"))
```

Creates a yellow brush and tells the drawing context (`dc`) to use it for filling any shapes drawn (like rectangles or circles).

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Tools for wx.PaintDC

- 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**.

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.

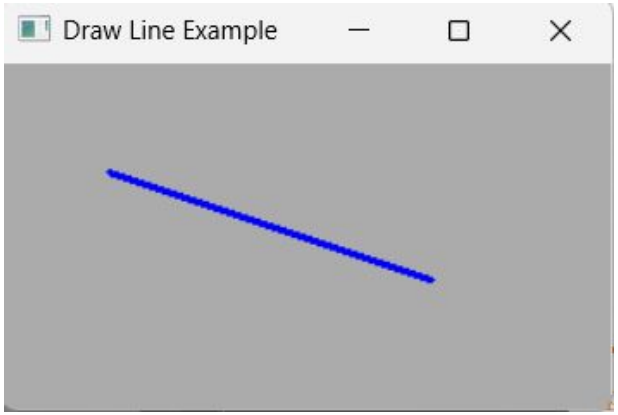
PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.PaintDC - Code example

Example 1:

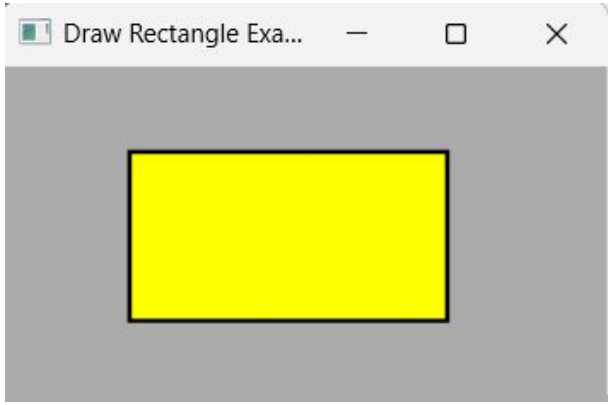
```
import wx

def OnPaint(event):
    dc = wx.PaintDC(frame)              # Create a drawing context
    dc.SetPen(wx.Pen("blue", 3))        # Blue color, 3-pixel thickness
    dc.DrawLine(50, 50, 200, 100)       # Draw a line from (50,50) to (200,100)

app = wx.App(False)
frame = wx.Frame(None, title="Draw Line Example", size=(300, 200))
frame.Bind(wx.EVT_PAINT, OnPaint)       # Bind paint event
frame.Show()
app.MainLoop()
```

Output:



Example 2:

```
import wx

def OnPaint(event):
    dc = wx.PaintDC(frame)
    dc.SetPen(wx.Pen("black", 2))           # Black border, 2 pixels thick
    dc.SetBrush(wx.Brush("yellow"))         # Yellow fill color
    dc.DrawRectangle(60, 40, 150, 80)       # Rectangle at (x=60, y=40)

app = wx.App(False)
frame = wx.Frame(None, title="Draw Rectangle Example", size=(300, 200))
frame.Bind(wx.EVT_PAINT, OnPaint)
frame.Show()
app.MainLoop()
```



WxCheckBox: A Quick Overview

- A **CheckBox** is a small square box that users can **check or uncheck**.
- It is 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:

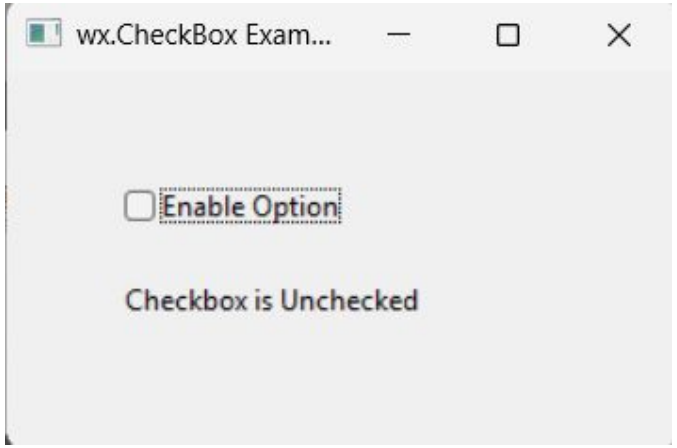
- **Parent** – The window or panel where the checkbox appears.
- **Id** – Widget ID (use `wx.ID_ANY` if not needed).
- **Label** – The text shown next to the checkbox.
- **pos** – Position of the checkbox (x, y) in pixels.

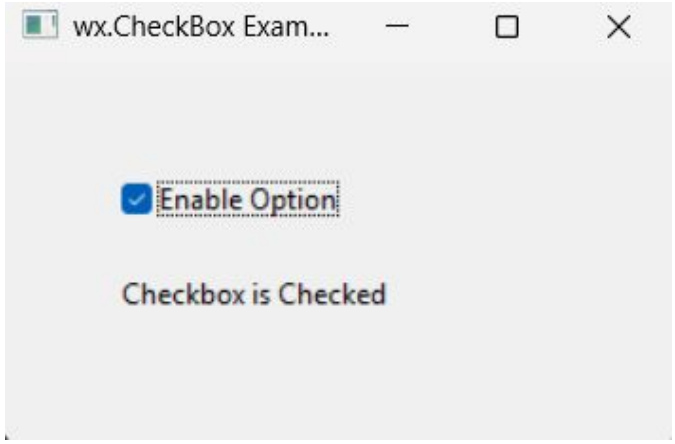
PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.CheckBox example

```
import wx

def OnCheck(event):
    if checkbox.GetValue():
        label.SetLabel("Checkbox is Checked")
    else:
        label.SetLabel("Checkbox is Unchecked")

app = wx.App(False)
frame = wx.Frame(None, title="wx.CheckBox Example", size=(300, 200))
panel = wx.Panel(frame)

checkbox = wx.CheckBox(panel, label="Enable Option", pos=(50, 50))
label = wx.StaticText(panel, label="Checkbox is Unchecked", pos=(50, 90))

checkbox.Bind(wx.EVT_CHECKBOX, OnCheck)

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

Output:





What are Widgets in WxPython

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

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.StaticText

Purpose:

Display text that cannot be edited by the user.

Syntax:

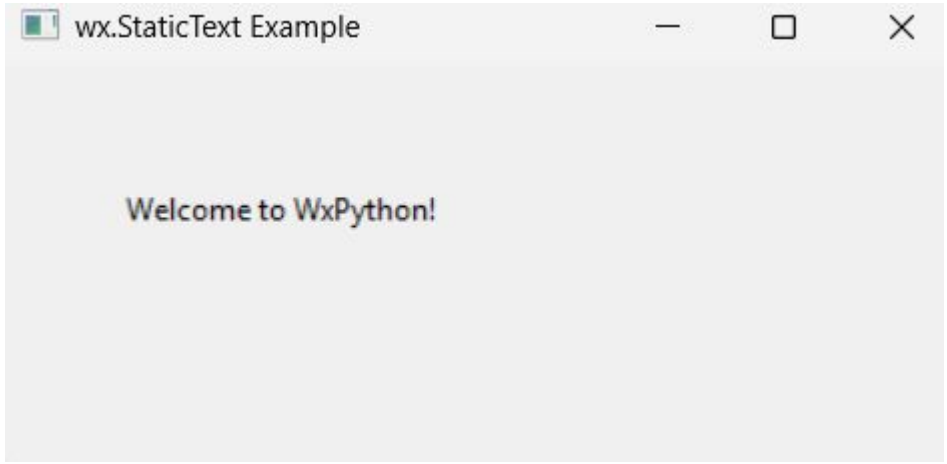
```
wx.StaticText(parent, id=wx.ID_ANY, label="", pos=(x, y))
```

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.

wx.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.TextCtrl

Purpose:

Allow the user to **enter or edit text**.

Syntax:

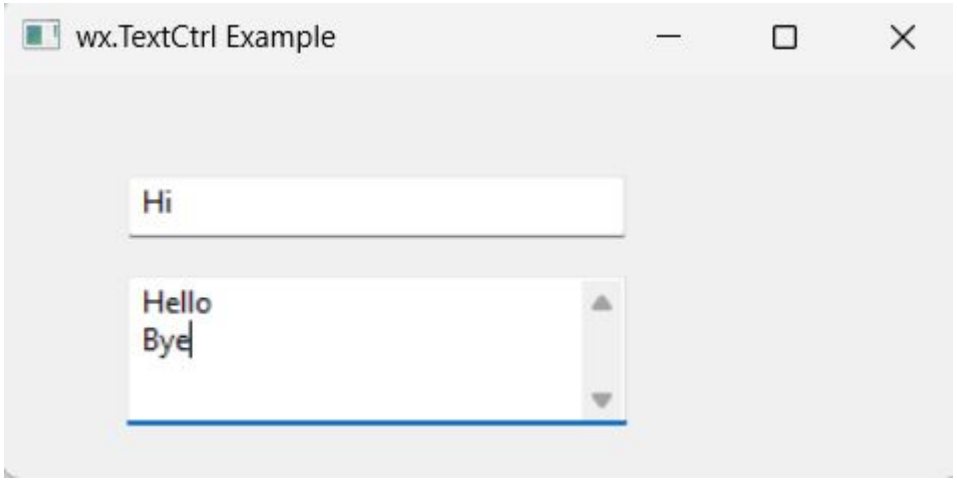
```
wx.TextCtrl(parent, id=wx.ID_ANY, value="", pos=(x, y), size=(w, h), style=0)
```

Common Styles:

wx.TE_MULTILINE ⟶ Multiple lines of text
wx.TE_PASSWORD ⟶ Hide characters (password field)
wx.TE_READONLY ⟶ Display only, not editable

wx.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)

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



PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.MessageDialog

Purpose:

Display a **message box** to the user (e.g., info, warning, question).

Syntax:

```
dlg = wx.MessageDialog(parent, message, caption, style)
```

Parameters:

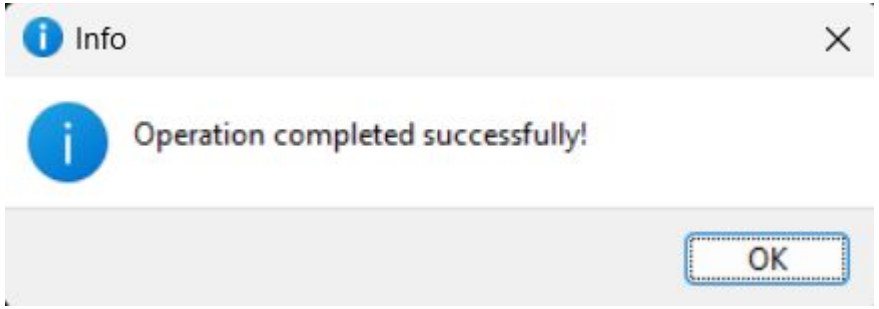
parent – The window or panel that owns the dialog box.
It decides where the dialog will appear (usually centered on this parent window).
message – The main text or information you want to show to the user.
Example: "File saved successfully!"
caption – The title displayed on the top bar of the dialog 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, wx.ICON_WARNING, wx.ICON_ERROR → Add icons

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.MessageDialog - 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()
```



wx.TextEntryDialog

Purpose:

Ask the user for text input in a popup box.

Syntax:

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

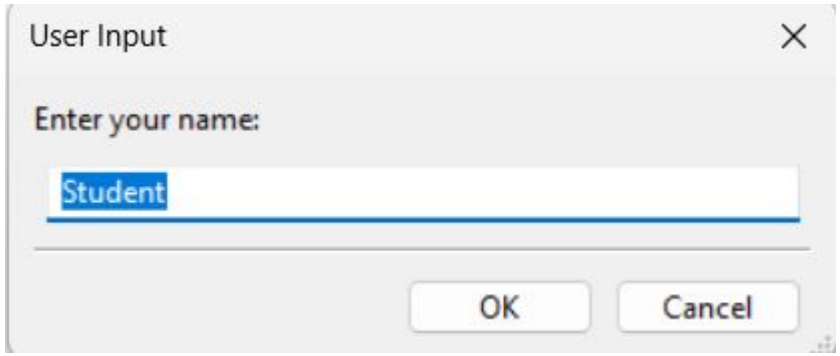
Parameters:

- **parent** – The window (frame or panel) on which the dialog will appear.
- **message** – The main text or information displayed inside the dialog box.
- **caption** – The title shown on the dialog window's title bar.
- **style** – Defines the buttons and icon type (e.g., OK, Cancel, Yes/No, Info, Warning).

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.TextEntryDialog - Example

```
import wx
app = wx.App(False)
frame = wx.Frame(None, title="TextEntryDialog Example")

dlg = wx.TextEntryDialog(frame, "Enter your name:", "User Input", "Student")
if dlg.ShowModal() == wx.ID_OK:
    name = dlg.GetValue()
    wx.MessageBox(f"Hello, {name}!", "Greeting")
dlg.Destroy()

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



SUMMARY

Method	Purpose	Example Use
wx.StaticText	Display static, non-editable text	Labels, headings
wx.TextCtrl	Get text input from user	Input boxes, notes
wx.MessageDialog	Show messages or alerts	Confirmation boxes
wx.TextEntryDialog	Ask user to type something	Name, email prompts

SetFont()

Purpose:

Change the font style, size, and weight of a widget's text.

Syntax:

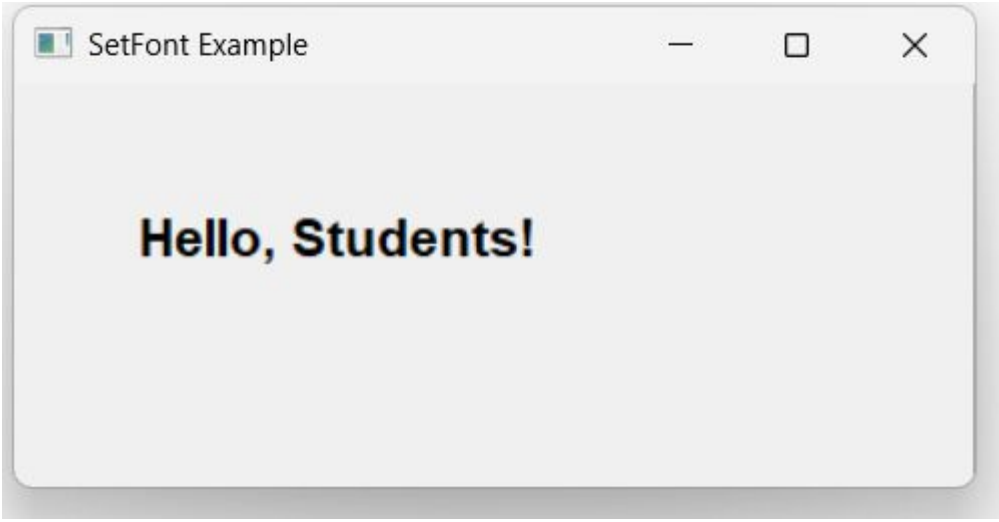
```
widget.SetFont(wx.Font(pointSize, family, style, weight))
```

Parameters:

- pointSize - font size in points
- family - font family (e.g., wx.FONTFAMILY_SWISS)
- style - normal, italic, slant
- weight - normal, bold, light

SetFont() - Example

```
import wx

app = wx.App(False)

frame = wx.Frame(None, title="SetFont Example", size=(400, 200))
panel = wx.Panel(frame)

# Create a label
label = wx.StaticText(panel, label="Hello, Students!", pos=(50, 50))

# Change the font: 16pt size, Swiss family, Normal style, Bold weight
label.SetFont(wx.Font(16, wx.FONTFAMILY_SWISS, wx.FONTSTYLE_NORMAL, wx.FONTWEIGHT_BOLD))

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



SetSize()

Purpose:

Set the width and height of the widget.

Syntax:

```
widget.SetSize(width, height)
```

Parameters:

- width - width of widget
- height - height of widget

SetSize() - Example

```
import wx

app = wx.App(False)

frame = wx.Frame(None, title="SetSize Example", size=(400, 200))
panel = wx.Panel(frame)

# Create a button
button = wx.Button(panel, label="Click Me", pos=(50, 50))

# Change button size (width = 200, height = 60)
button.SetSize(200, 60)

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

SetBackgroundColour()

Purpose:

Change background color of a widget

Syntax:

```
widget.SetBackgroundColour("color_name")
```

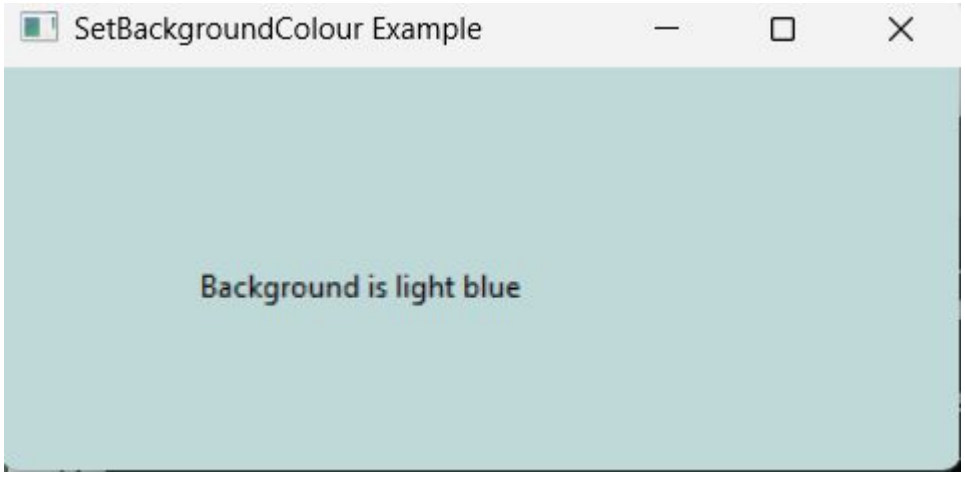
Colors:

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

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

SetBackgroundColour() - Example

```
import wx

app = wx.App(False)

frame = wx.Frame(None, title="SetBackgroundColour Example", size=(400, 200))
panel = wx.Panel(frame)

# Change background color of the panel
panel.SetBackgroundColour("light blue")

# Add a label for reference
label = wx.StaticText(panel, label="Background is light blue", pos=(80, 80))

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



PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Layout Management

Without layout management:

- Widgets can overlap or be misaligned.
- GUI may look messy on different screen sizes.

With Sizers (layout managers):

- Widgets arrange themselves automatically.
- Responsive design — works on all devices.
- Easy to adjust spacing and alignment.

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.BoxSizer

Purpose:

Arrange widgets in a straight line – horizontally or vertically

Syntax:

```
sizer = wx.BoxSizer(wx.HORIZONTAL)  # or wx.VERTICAL
```

Flags:

wx.ALL - Adds space (margin) around the widget
wx.EXPAND - Makes the widget stretch to fill space
wx.ALIGN_CENTER - Centers the widget

Proportions:

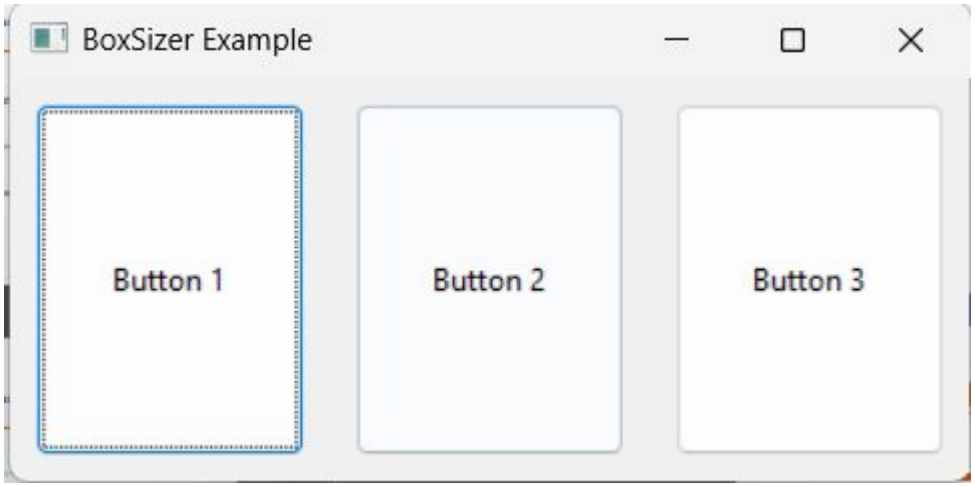
0 - Widget keeps its natural size
>0 - Widget expands proportionally

If btn1 has proportion=1 and btn2 has proportion=2 → btn2 gets twice the space of btn1.

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

wx.BoxSizer() - Example

```
import wx

app = wx.App(False)

frame = wx.Frame(None, title="BoxSizer Example", size=(400, 200))
panel = wx.Panel(frame)

# Create buttons
btn1 = wx.Button(panel, label="Button 1")
btn2 = wx.Button(panel, label="Button 2")
btn3 = wx.Button(panel, label="Button 3")

# Create a horizontal BoxSizer
sizer = wx.BoxSizer(wx.HORIZONTAL)

# Add buttons to the sizer with spacing
sizer.Add(btn1, 1, wx.ALL | wx.EXPAND, 10)
sizer.Add(btn2, 1, wx.ALL | wx.EXPAND, 10)
sizer.Add(btn3, 1, wx.ALL | wx.EXPAND, 10)

# Apply sizer to panel
panel.SetSizer(sizer)

frame.Show()
```



wx.GridSizer

Purpose:

Arrange widgets in a grid of rows and columns

Syntax:

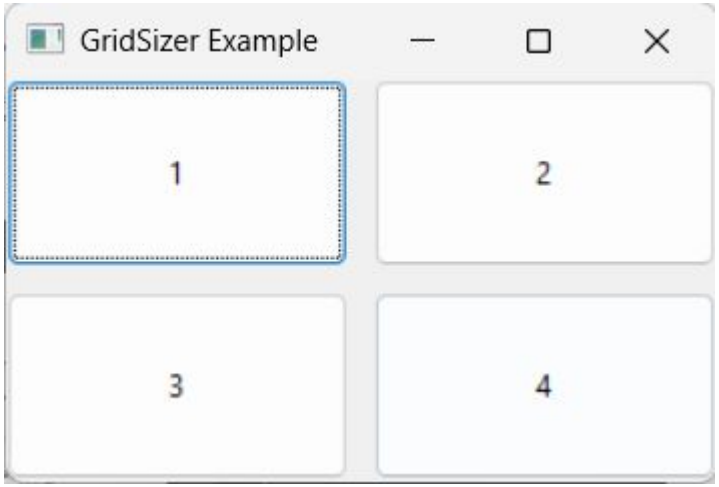
```
sizer = wx.GridSizer(rows, cols, vgap, hgap)
```

Parameters:

rows - Number of rows
cols - Number of columns
vgap - Vertical space between widgets
hgap - Horizontal space between widgets

wx.GridSizer() - Example

```
import wx

app = wx.App(False)

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

# Create 4 buttons
b1 = wx.Button(panel, label="1")
b2 = wx.Button(panel, label="2")
b3 = wx.Button(panel, label="3")
b4 = wx.Button(panel, label="4")

# Create GridSizer with 2 rows, 2 columns, 10px gaps
sizer = wx.GridSizer(2, 2, 10, 10)

# Add buttons to the sizer
sizer.Add(b1, 0, wx.EXPAND)
sizer.Add(b2, 0, wx.EXPAND)
sizer.Add(b3, 0, wx.EXPAND)
sizer.Add(b4, 0, wx.EXPAND)

# Apply sizer to panel
panel.SetSizer(sizer)

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



SUMMARY

Method	Purpose
SetFont()	Change font size, style, weight
SetSize()	Set widget width & height
SetBackgroundColour()	Change background color
wx.BoxSizer	Arrange widgets in line (H/V)
wx.GridSizer	Arrange widgets in grid

THANK YOU

Department of Computer Science and Engineering

Prof. Kundhavai K R, CSE Department

Ack: Teaching Assistant – Trishita Umapathi
PES2UG22CS631

Thank You –

Adithya Jeyaramsankar**–** PES2UG22CS029

Aditiyaa Naag-PES2UG22CS033

Akshat Bhandari -PES2UG22CS048