Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Building a Basic User-Interactive GUI Application using Kivy in Python	
Experiment No: 16	**Date:**	**Enrollment No: 92400133110**

Aim: Building a Basic User-Interactive GUI Application using Kivy in Python

IDE:

A comparative analysis of Tkinter and Kivy, two popular Python GUI frameworks:

Criteria	Tkinter	Kivy
Origin/Integration	Built-in standard GUI toolkit for Python	Third-party library, must be installed separately
Platform Support	Cross-platform (Windows, macOS, Linux)	Cross-platform (Windows, macOS, Linux, Android, iOS)
Mobile App Support	Not natively supported	Yes, designed for mobile apps (Android/iOS)
Look and Feel	Native look (uses OS elements; sometimes outdated)	Custom UI (same look on all platforms)
Ease of Use (Beginner Friendly)	Easier for beginners, simple widgets and layout	Slightly steeper learning curve due to different approach
Custom Widgets	Limited custom widgets	Highly customizable, supports multi-touch, gestures
Performance	Lightweight, fast for basic applications	Better for graphics-rich or touch-based applications
Layout Management	Pack, Grid, Place layout managers	Uses relative positioning and advanced layout controls
Graphics and Animation	Basic support	Rich support for OpenGL, animations, and gestures
Community and Support	Long-standing, extensive community	Newer but active open-source community
Event Handling	Traditional event binding using command and bind	Event-driven, uses Clock, on_touch_*, properties
Development Use Case	Desktop apps, simple tools, admin panels	Mobile apps, multimedia apps, dashboards, games

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Building a Basic User-Interactive GUI Application using Kivy in Python	
Experiment No: 16	**Date:**	**Enrollment No: 92400133110**

Use Tkinter:
You are developing a simple desktop application, teaching basic GUI programming, or need something lightweight and native-looking on desktops.

Use Kivy:
You are targeting mobile platforms, want touch support, need consistent UI across devices, or are building multimedia-rich or gesture-based apps.

Library	Purpose / UI Type	Installation	Import Syntax	Best Use Case
Tkinter	Native Desktop GUI	Built-in (python3-tk on Linux)	import tkinter as tk	Basic desktop apps, learning GUI concepts
Kivy	Multi-touch apps for desktop & mobile	pip install kivy	from kivy.app import App	Mobile-like UIs, gesture support, kiosk apps
Textual	Terminal UI with app-like look	pip install textual	from textual.app import App	Terminal dashboards, TUI-based dev tools
Remi	Web UI from pure Python (no HTML)	pip install remi	import remi.gui as gui	Turn Python scripts into web apps easily
NiceGUI	Fast web UI with Vue3 + Python	pip install nicegui	from nicegui import ui	Reactive dashboards, IoT UI, admin panels
Flet	Flutter-style UI in pure Python	pip install flet	import flet as ft	Mobile/web-style apps, no need for Dart
Eel	HTML/JS frontend + Python backend	pip install eel	import eel	Convert HTML+JS UI into desktop apps with Python
Dear PyGui	GPU-accelerated desktop GUI	pip install dearpygui	import dearpygui.dearpygui as dpg	High-perf apps, dashboards, tools with fast UI
pywebview	Native desktop app with embedded web UI	pip install pywebview	import webview	Build web UI as desktop apps with native look

Toga	Native UI for desktop/mobile (BeeWare)	pip install toga	import toga	Native look across macOS, Windows, Linux
JustPy	Server-side reactive web UI (no JS needed)	pip install justpy	import justpy as jp	Dashboards, education tools, reactive forms
Gooey	Turn CLI apps into GUI instantly	pip install gooey	from gooey import Gooey	Beautify CLI tools, Python scripts for non-coders

Example Syntax Comparison:

Tkinter Button Example:

```
import tkinter as tk

def say_hello():
    print("Hello, Tkinter!")

root = tk.Tk()
btn = tk.Button(root, text="Click Me", command=say_hello)
btn.pack()
root.mainloop()
```

Kivy Button Example:

```
from kivy.app import App
from kivy.uix.button import Button

class MyApp(App):
    def build(self):
        return Button(text='Click Me', on_press=lambda x: print("Hello, Kivy!"))

MyApp().run()
```

	Marwadi University **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Building a Basic User-Interactive GUI Application using Kivy in Python	
Experiment No: 16	**Date:**	**Enrollment No: 92400133110**

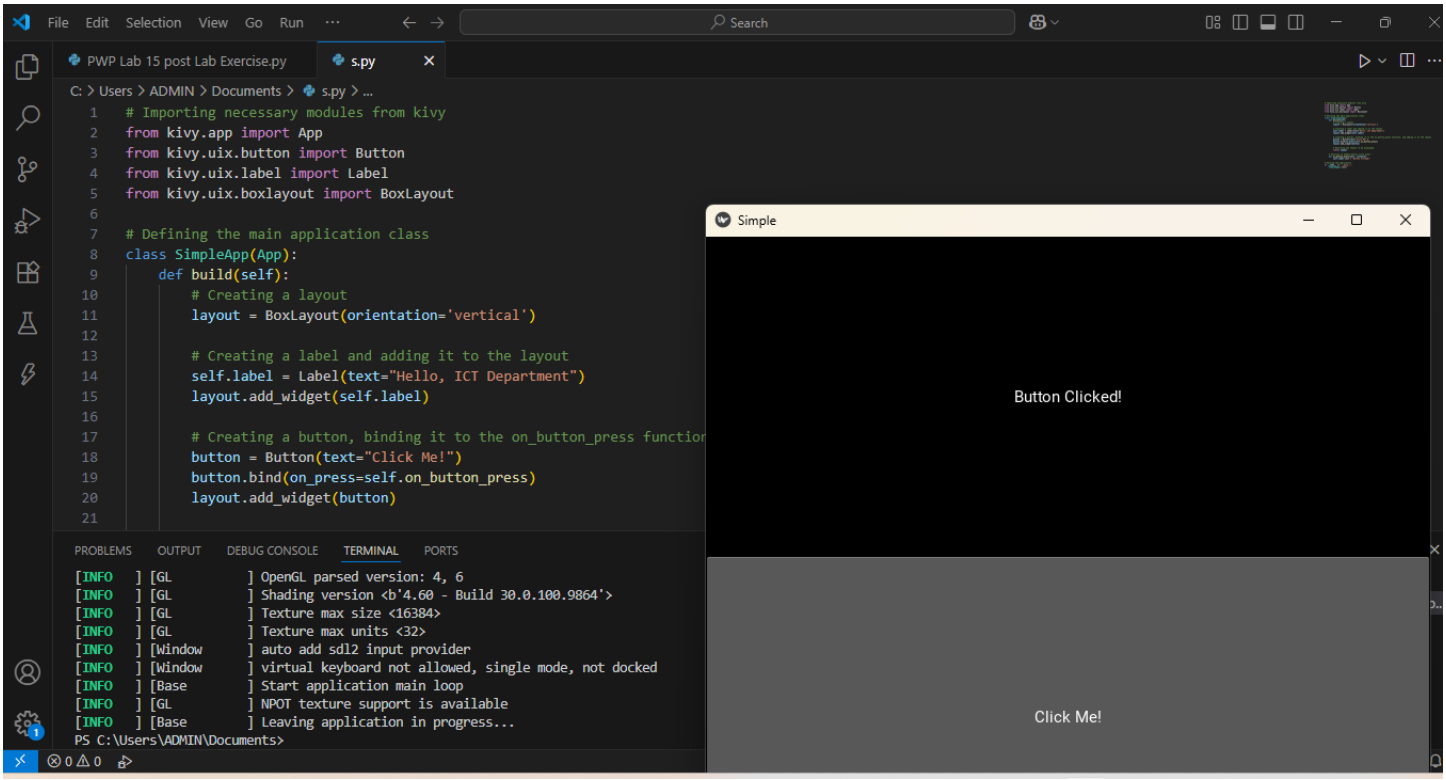
Kivy was first released in early 2011. This cross-platform Python framework can be deployed to Windows, Mac, Linux, and Raspberry Pi. It supports multitouch events in addition to regular keyboard and mouse inputs. Kivy even supports GPU acceleration of its graphics, since they're built using OpenGL ES2.

Before using Kivy, you need to install it. You can install it using pip:
pip install kivy

Create a Simple Kivy Application
Let's start by building a basic app with a label and a button.

```
# Importing necessary modules from kivy
from kivy.app import App
from kivy.uix.button import Button
from kivy.uix.label import Label
from kivy.uix.boxlayout import BoxLayout

# Defining the main application class
class SimpleApp(App):
    def build(self):
        # Creating a layout
        layout = BoxLayout(orientation='vertical')

        # Creating a label and adding it to the layout
        self.label = Label(text="Hello, ICT Department")
        layout.add_widget(self.label)

        # Creating a button, binding it to the on_button_press function, and adding it to the layout
        button = Button(text="Click Me!")
        button.bind(on_press=self.on_button_press)
        layout.add_widget(button)
```

```
        # Returning the layout to be displayed
        return layout

    # Function to handle button click event
    def on_button_press(self, instance):
        self.label.text = "Button Clicked!"

# Running the application
if __name__ == '__main__':
    SimpleApp().run()
```

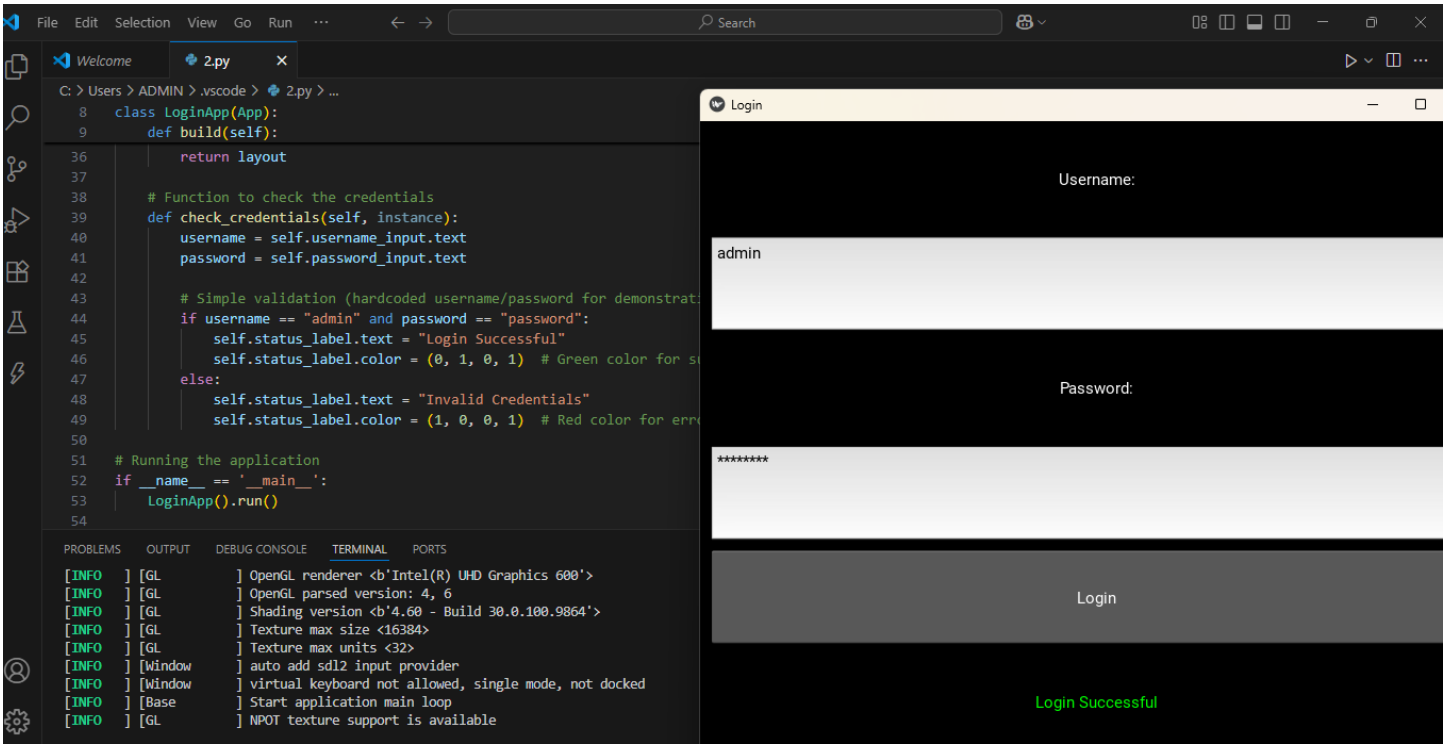


Kivy Login Page Example

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.label import Label
from kivy.uix.textinput import TextInput
```

```
from kivy.uix.button import Button

# Defining the main application class
class LoginApp(App):
    def build(self):
        # Main layout
        layout = BoxLayout(orientation='vertical', padding=10, spacing=10)

        # Username label and input
        self.username_label = Label(text="Username:")
        layout.add_widget(self.username_label)

        self.username_input = TextInput(multiline=False)
        layout.add_widget(self.username_input)

        # Password label and input
        self.password_label = Label(text="Password:")
        layout.add_widget(self.password_label)

        self.password_input = TextInput(password=True, multiline=False)
        layout.add_widget(self.password_input)

        # Login button
        self.login_button = Button(text="Login")
        self.login_button.bind(on_press=self.check_credentials)
        layout.add_widget(self.login_button)

        # Label to display the login status
        self.status_label = Label(text="")
        layout.add_widget(self.status_label)

        return layout

    # Function to check the credentials
```

```
    def check_credentials(self, instance):
        username = self.username_input.text
        password = self.password_input.text

        # Simple validation (hardcoded username/password for demonstration)
        if username == "admin" and password == "password":
            self.status_label.text = "Login Successful"
            self.status_label.color = (0, 1, 0, 1)  # Green color for success
        else:
            self.status_label.text = "Invalid Credentials"
            self.status_label.color = (1, 0, 0, 1)  # Red color for error

# Running the application
if __name__ == '__main__':
    LoginApp().run()
```



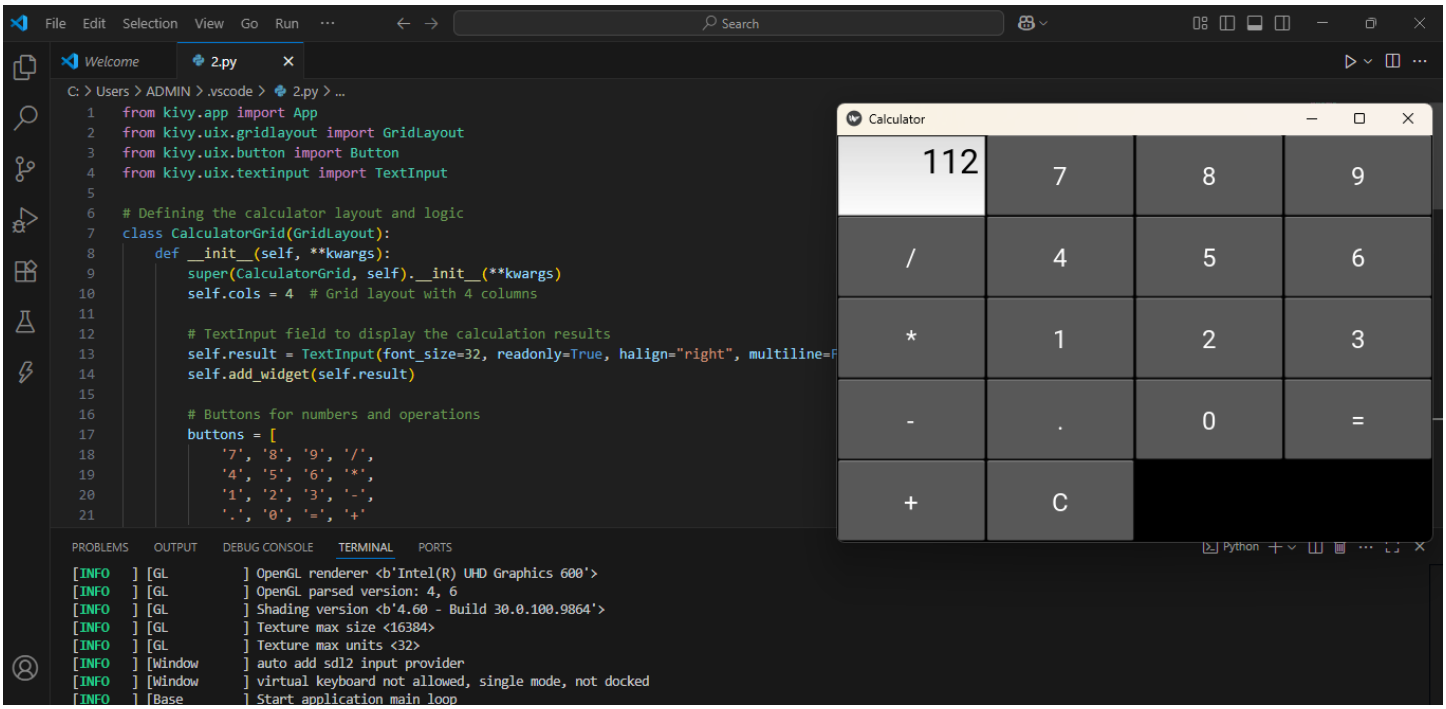
Calculator App Using Kivy

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Building a Basic User-Interactive GUI Application using Kivy in Python	
Experiment No: 16	**Date:**	**Enrollment No: 92400133110**

```
from kivy.app import App
from kivy.uix.gridlayout import GridLayout
from kivy.uix.button import Button
from kivy.uix.textinput import TextInput

# Defining the calculator layout and logic
class CalculatorGrid(GridLayout):
    def __init__(self, **kwargs):
        super(CalculatorGrid, self).__init__(**kwargs)
        self.cols = 4  # Grid layout with 4 columns

        # TextInput field to display the calculation results
        self.result = TextInput(font_size=32, readonly=True, halign="right", multiline=False)
        self.add_widget(self.result)

        # Buttons for numbers and operations
        buttons = [
            '7', '8', '9', '/',
            '4', '5', '6', '*',
            '1', '2', '3', '-',
            '.', '0', '=', '+'
        ]

        # Adding buttons to the layout
        for button in buttons:
            self.add_widget(Button(text=button, font_size=24, on_press=self.on_button_press))

        # Clear button to reset the calculator
        self.add_widget(Button(text="C", font_size=24, on_press=self.clear_result))

    # Function to handle button press events
    def on_button_press(self, instance):
        current_text = self.result.text
        button_text = instance.text
```

```
        # If the equals sign is pressed, evaluate the expression
        if button_text == "=":
            try:
                self.result.text = str(eval(current_text))
            except Exception:
                self.result.text = "Error"
        else:
            # Otherwise, append the pressed button's text to the current expression
            if current_text == "Error":
                self.result.text = button_text  # Reset the result if there's an error
            else:
                self.result.text += button_text

    # Function to clear the result field
    def clear_result(self, instance):
        self.result.text = ""

# Main App class
class CalculatorApp(App):
    def build(self):
        return CalculatorGrid()

# Running the application
if __name__ == '__main__':
    CalculatorApp().run()
```

Marwadi University Marwadi Chandarana Group NAAC A+	**Marwadi University** **Faculty of Engineering & Technology** **Department of Information and Communication Technology**	
Subject: Programming With Python (01CT1309)	**Aim:** Building a Basic User-Interactive GUI Application using Kivy in Python	
Experiment No: 16	**Date:**	**Enrollment No: 92400133110**



Post Lab Exercise:

- Design Counter App (This app has a button that increments a counter displayed on the screen every time the button is clicked)
- Git hub
- https://github.com/hemanth-singampalli/-hemanth.git
- Text Input App (This app allows users to type in a text field and display the typed text on the screen when a button is pressed.)
- Git hub
- https://github.com/hemanth-singampalli/hemanth.git