	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:92301733025.**

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

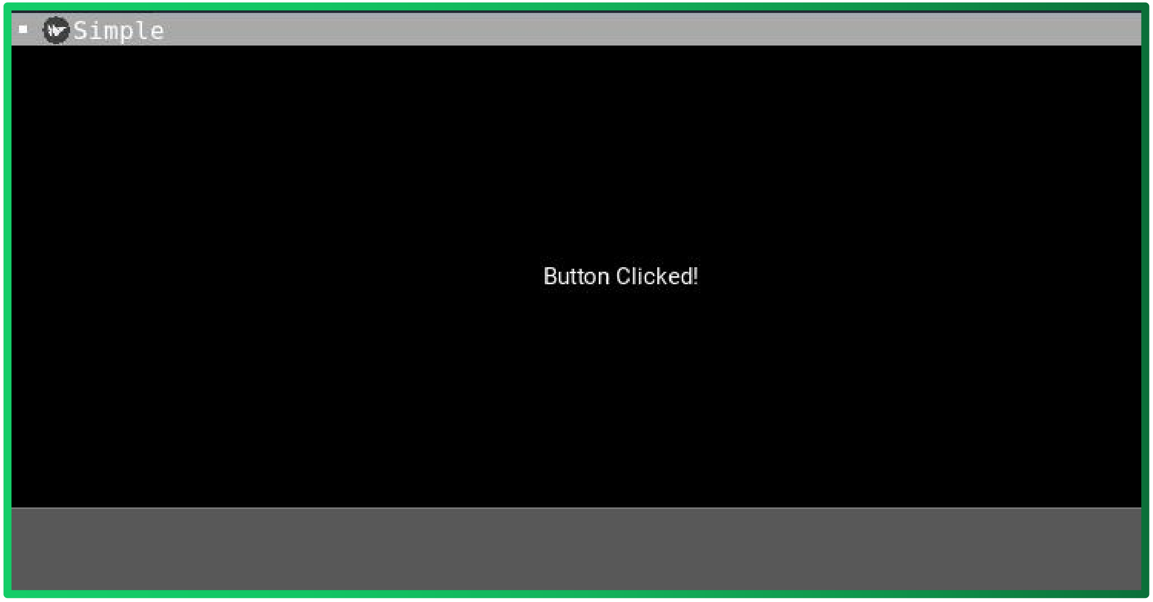
```
pip install kivy
```

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()
```

Output :

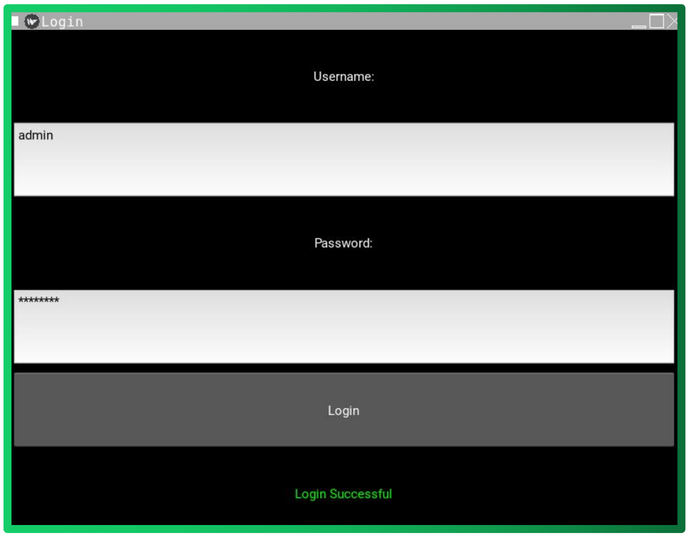


Button Clicked!

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()
```

Output :



Username:

admin

Password:

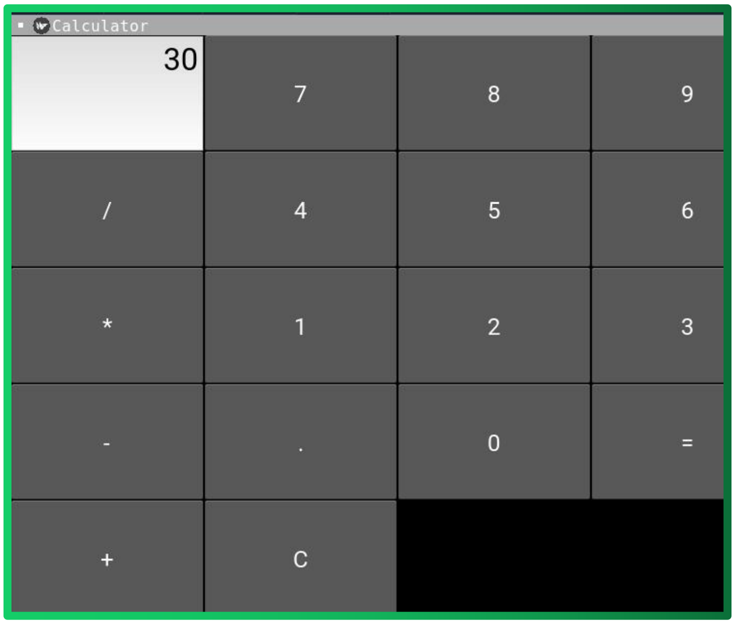
****

Login Successful

Calculator App Using Kivy

```
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()
```

Output :



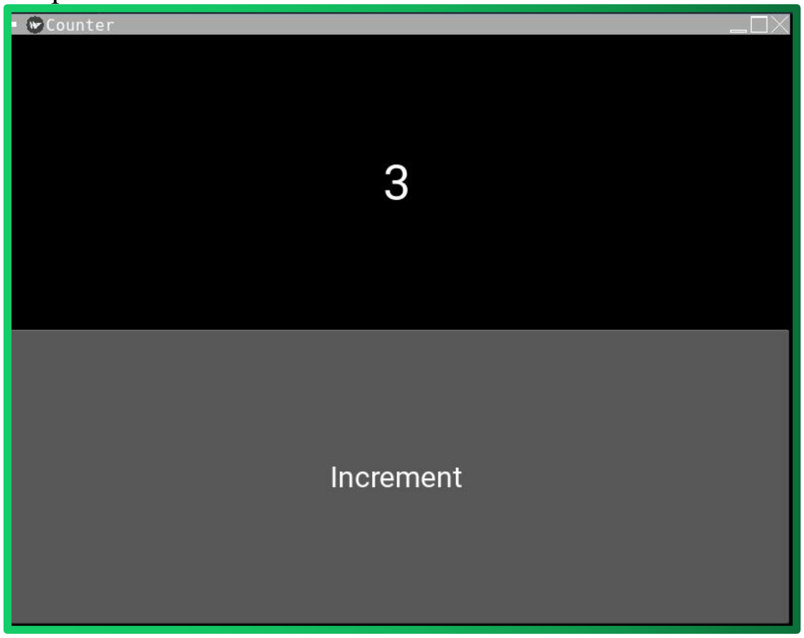
Post Lab Exercise:

- Design Counter App (This app has a button that increments a counter displayed on the screen every time the button is clicked)

Code :

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.button import Button
from kivy.uix.label import Label

class CounterApp(App):
    def build(self):
        self.counter = 0
        layout = BoxLayout(orientation='vertical')
        self.label = Label(text=str(self.counter), font_size=50)
        increment_button = Button(text="Increment", font_size=30)
        increment_button.bind(on_press=self.increment_counter)
        layout.add_widget(self.label)
        layout.add_widget(increment_button)
        return layout
    def increment_counter(self, instance):
        self.counter += 1
        self.label.text = str(self.counter)
if __name__ == '__main__':
    CounterApp().run()
```

Output :

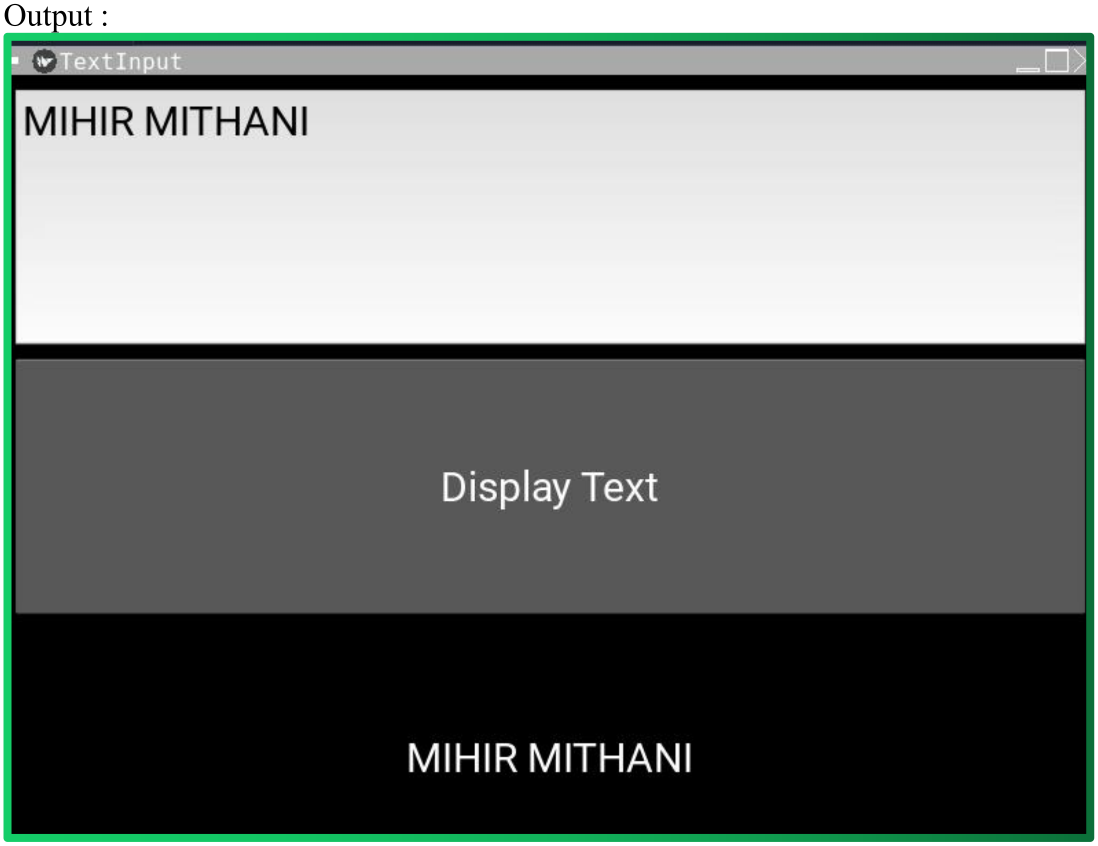


- Text Input App (This app allows users to type in a text field and display the typed text on the screen when a button is pressed.)

  Code :

```
from kivy.app import App
from kivy.uix.boxlayout import BoxLayout
from kivy.uix.textinput import TextInput
from kivy.uix.button import Button
from kivy.uix.label import Label
class TextInputApp(App):
    def build(self):
        layout = BoxLayout(orientation='vertical', padding=10, spacing=10)
        self.text_input = TextInput(hint_text="Type something here", font_size=30)
        self.label = Label(text="Your text will appear here", font_size=30)
        submit_button = Button(text="Display Text", font_size=30)
        submit_button.bind(on_press=self.display_text)
        layout.add_widget(self.text_input)
        layout.add_widget(submit_button)
        layout.add_widget(self.label)
        return layout
    def display_text(self, instance):
        entered_text = self.text_input.text
        self.label.text = entered_text
if __name__ == '__main__':
    TextInputApp().run()
```

Output :



MIHIR MITHANI

MIHIR MITHANI