Experiment No:4

Aim: To create a GUI with python containing different widgets.

Theory:

Python Libraries for GUI Programming

We can use any of the following toolkits in Python for GUI

programming. 1. Tkinter:

Tkinter is a standard package used for GUI programming in Python. This is built on top of the Tk interface.

1. PyQt:

PyQt is a Python toolkit binding of the Qt toolkit. Qt is a C++ framework that is used by Python to implement a cross-platform PyQt toolkit as a plug-in.

2. wxPython:

wxPython is also a cross-platform GUI toolkit. It is a wrapper for the API

wxWidgets. **Python Tkinter Module**

Tkinter is a standard Python library used for GUI programming. It provides an object-oriented interface to build the Tk GUI toolkit. It is a faster and easier way to build a GUI in Python.

An empty Tkinter top-level window can be created by using the following steps.

1. import the Tkinter module.
2. Create the main application window.
3. Add the widgets like labels, buttons, frames, etc. to the window.
4. Call the main event loop so that the actions can take place on the user's computer screen.

```
from tkinter import *
#creating the application main window.
top = Tk()
#Entering the event main loop
top.mainloop()
```

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method

syntax

1. widget.pack(options)

A list of possible options that can be passed in pack() is given below.

- **expand:** If the expand is set to true, the widget expands to fill any space.
- **Fill:** By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.
- **size:** it represents the side of the parent to which the widget is to be placed on the window.

2. The grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax

1. widget.grid(options)

A list of possible options that can be passed inside the grid() method is given below.

- **Column**

  The column number in which the widget is to be placed. The leftmost column is represented by 0.

- **Columnspan**
  The width of the widget. It represents the number of columns up to which, the column is expanded.
- **ipadx, ipady**
  It represents the number of pixels to pad the widget inside the widget's border.

3. The place() method

The place() geometry manager organizes the widgets to the specific x and y

coordinates. Syntax

1. widget.place(options)

A list of possible options is given below.

- **Anchor:** It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)
- **bordermode:** The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.
- **height, width:** It refers to the height and width in pixels.
- **relheight, relwidth:** It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.
- **relx, rely:** It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

Widget						
	Button	Frame	Menu		r indow	
	Canvas	Label	Message	Text	LabelFrame	
	Checkbutton	ListBox	Radiob			
	Entry	Menubutton	Scale	Spinbo x		

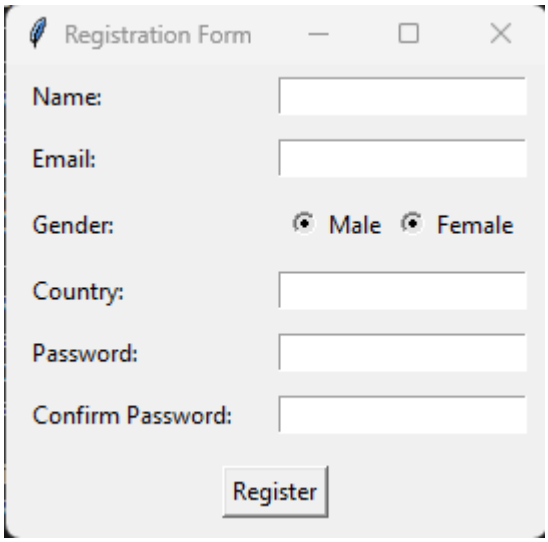
PROGRAM

Program: To create registration form using tkniter module

```
from tkinter import *

def register():
    name = name_entry.get()
    email = email_entry.get()
    gender = gender_var.get()
    country = country_var.get()
    password = password_entry.get()
    confirm_password = confirm_password_entry.get()

    print("Name:", name)
    print("Email:", email)
    print("Gender:", gender)
    print("Country:", country)
    print("Password:", password)
    print("Confirm Password:", confirm_password)

    # You can add your validation logic and registration process here

# Creating Tkinter window
root = Tk()
root.title("Registration Form")
```

```
# Variables for storing user input
name_var = StringVar()
email_var = StringVar()
gender_var = StringVar()
country_var = StringVar()
password_var = StringVar()
confirm_password_var = StringVar()

# Labels
Label(root, text="Name:").grid(row=0, column=0, sticky=W, padx=10, pady=5)
Label(root, text="Email:").grid(row=1, column=0, sticky=W, padx=10, pady=5)
Label(root, text="Gender:").grid(row=2, column=0, sticky=W, padx=10, pady=5)
Label(root, text="Country:").grid(row=3, column=0, sticky=W, padx=10, pady=5)
Label(root, text="Password:").grid(row=4, column=0, sticky=W, padx=10, pady=5)
Label(root, text="Confirm Password:").grid(row=5, column=0, sticky=W, padx=10, pady=5)

# Entries
name_entry = Entry(root, textvariable=name_var)
name_entry.grid(row=0, column=1, padx=10, pady=5)
email_entry = Entry(root, textvariable=email_var)
email_entry.grid(row=1, column=1, padx=10, pady=5)
gender_frame = Frame(root)
gender_frame.grid(row=2, column=1, padx=10, pady=5)
Radiobutton(gender_frame, text="Male", variable=gender_var, value="Male").pack(side=LEFT)
Radiobutton(gender_frame, text="Female", variable=gender_var, value="Female").pack(side=LEFT)
country_entry = Entry(root, textvariable=country_var)
country_entry.grid(row=3, column=1, padx=10, pady=5)
password_entry = Entry(root, show="*", textvariable=password_var)
password_entry.grid(row=4, column=1, padx=10, pady=5)
confirm_password_entry = Entry(root, show="*", textvariable=confirm_password_var)
confirm_password_entry.grid(row=5, column=1, padx=10, pady=5)

# Submit button
Button(root, text="Register", command=register).grid(row=6, column=0, columnspan=2, pady=10)

root.mainloop()
```

OUTPUT:



Conclusion:

Through the implementation of various widgets in Python's GUI, we have successfully crafted an interactive user interface. This experiment underscores the versatility andfunctionality of Python for developing intuitive graphical applications