

PYTHON EXPERIMENT 7

Name : Gini Chacko

Roll : 8942

Class : SE Comps B

Aim:

Creating GUI (Registration Form for any Application) with python containing widgets such as labels, textbox, radio buttons, checkboxes, List Boxes, images and custom dialog boxes.

Objectives:

- 1) To create Graphical user interface in Python using Tkinter library.**
- 2) To explore variety of widgets while creating Registration form.**

CODE:

```
import tkinter as tk
from tkinter import ttk
from tkinter import *

window = Tk()
window.title("REGISTRATION FORM")
window.geometry('350x320')          #widthxheight
window.configure(background = "#dce6f5")

l1 = Label(window ,width=15,bg='#ffffff', text = "First Name")
l1.grid(row = 0,column = 0, padx=10, pady=10)
txt1 = Entry(window,width=20)
txt1.grid(column=2, row=0, padx=10, pady=10)
l2 = Label(window ,width=15,bg='#ffffff',text = "Last Name")
l2.grid(row = 5,column = 0, padx=10, pady=10)
txt2 = Entry(window,width=20)
txt2.grid(column=2, row=5, padx=10, pady=10)
l3 = Label(window ,width=15,bg='#ffffff',text = "Email Id")
l3.grid(row = 10,column = 0, padx=10, pady=10)
```

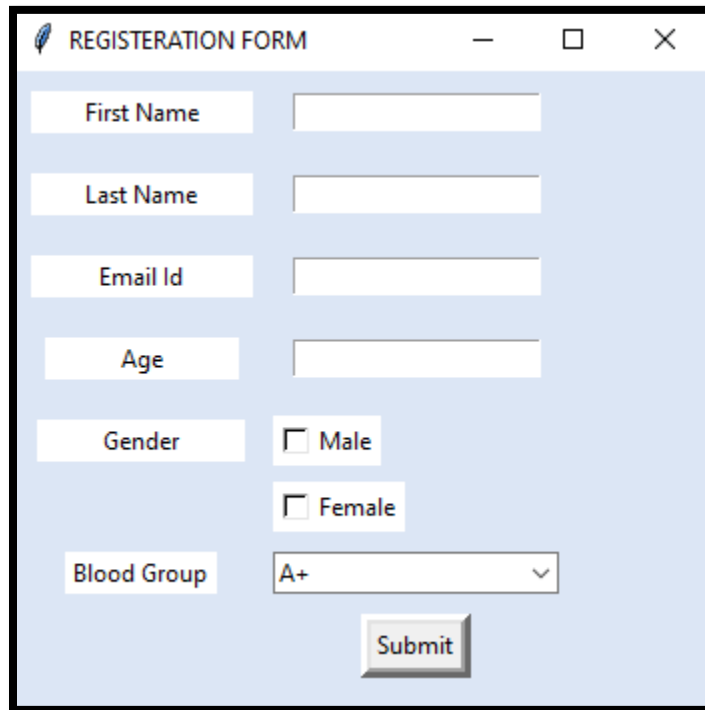
```

txt3 = Entry(window,width=20)
txt3.grid(column=2, row=10, padx=10, pady=10)
l4 = Label(window ,width=13,bg='#ffffff',text ="Age")
l4.grid(row = 15,column = 0, padx=10, pady=10)
txt4 = Entry(window,width=20)
txt4.grid(column=2, row=15, padx=10, pady=10)
l5 = Label(window ,width=14,bg='#ffffff',text = "Gender")
l5.grid(row = 20,column = 0, padx=10, pady=10)
var1 = IntVar()
Checkbutton (window,bg='#ffffff', text='Male', variable=var1).grid(column=2,row=20, sticky=W)
var2 = IntVar()
Checkbutton(window,bg='#ffffff', text='Female', variable=var2).grid(column=2,row=22, sticky=W)
l7= Label(window ,width=10,bg='#ffffff',text = "Blood Group")
l7.grid(row =25,column = 0, padx=10, pady=10)

n = tk.StringVar()
bg = ttk.Combobox(window, width = 20,textvariable = n)
# Adding combo- box drop down list
bg['values'] = ('A+',
                'A-',
                'B+',
                'B-',
                'O+',
                'O-',
                'AB+',
                'AB-')
bg.grid(column =2, row = 25)
bg.current(0)
btn = Button(window, text='Submit', bd = '5',command = window.destroy)
btn.grid(row=32, column=2)
window.mainloop()

```

OUTPUT:



REGISTRATION FORM

First Name

Last Name

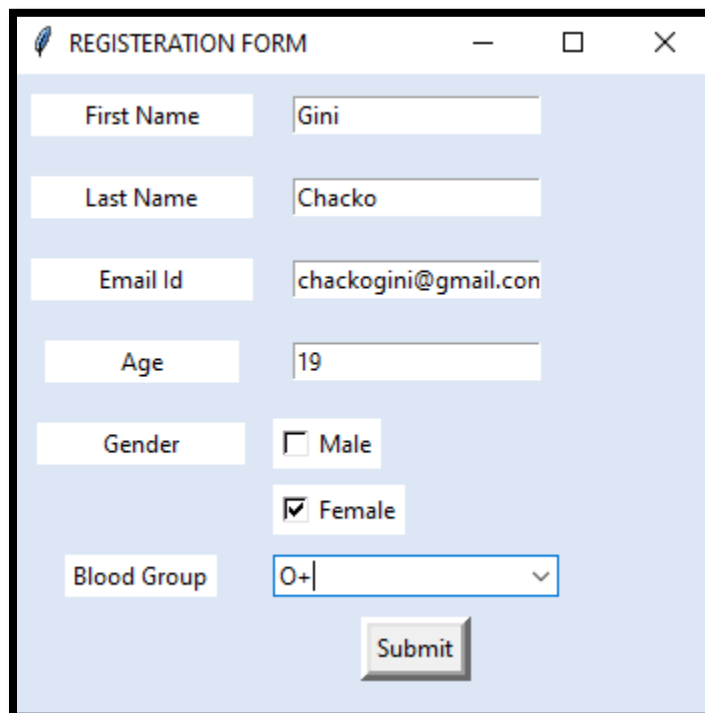
Email Id

Age

Gender ☐ Male ☐ Female

Blood Group

Submit



REGISTRATION FORM

First Name

Last Name

Email Id

Age

Gender ☐ Male ☒ Female

Blood Group

Submit

Post Lab questions:

1) What are the different libraries available in Python to create GUI?

Ans:

1. Kivy : Kivy is an open-source Python library for rapid development of applications which makes use of innovative user interfaces, such as multi-touch apps. This framework is a cross-platform and runs on Linux, Windows, OS X, Android, iOS, and Raspberry Pi. The graphics engine is built using a modern and fast graphics pipeline.

2. Libavg : Libavg is an open-source high-level development platform for media-centric applications. It uses Python as the scripting language, is written in high-speed C++ and uses modern OpenGL for display output. This framework is great for the development of modern touch UIs and supports all major touch driver models, including Windows touch, Linux XInput, and TUIO. Libavg has a number of features such as it supports the full variety of display elements which modern graphics-intensive applications needs, the layout engine supports thousands of display elements on the screen at once as well as a hardware-accelerated video output and much more.

3. PyQt : PyQt is a set of Python v2 and v3 bindings for The Qt Company's Qt application framework and runs on all platforms supported by Qt including Windows, OS X, Linux, iOS and Android. This framework brings together the Qt C++ cross-platform application framework and the cross-platform interpreted language Python. Qt is a cross-platform application development framework for desktop, embedded and mobile. Qt includes abstractions of network sockets, threads, Unicode, regular expressions, SQL databases, SVG, OpenGL, XML, a fully functional web browser, a help system, a multimedia framework, as well as a rich collection of GUI widgets.

4. PySimpleGUI : PySimpleGUI is a GUI framework for Python which supports Python 3 version. It is simple to create custom GUIs with the help of this framework. Currently, there are 4 actively developed and maintained "ports" of this framework which are tkinter which is fully complete, Qt using Pyside2 which is at alpha stage, WxPython which is at the development stage and Remi (Web browser support) which is also at the development stage.

5. Pyforms : Pyforms is a cross-environment framework for developing GUI applications. The framework offers a Python layer of desktop forms, based on PyQt, OpenGL and other libraries, allow applications to run on Desktop GUI, Web

and terminal without requiring code modifications. Pyforms promotes modular software design and code reusability with minimal effort.

6. Tkinter : Tkinter or Tk interface is Python's de-facto standard GUI (Graphical User Interface) package. This is an open-source framework and is available on platforms like Unix and Windows. It is one of the simplest and most popular ways to build a GUI-based application in Python.

7. Wax : Wax is a Python GUI framework that sits atop wxPython (wxWindows for Python). This framework removes the low-level aspects of wxPython (which is basically a direct binding to the ugly C API) and gives you simple python objects to create your GUI. It also runs on many platforms (Win32, Linux w/ GTK, and macOS/OSX w/ Carbon).

8. WxPython : WxPython is an open-source cross-platform GUI toolkit for Python. It is implemented as a set of Python extension modules that wrap GUI components of the popular wxWidgets cross-platform library, which is written in C++. With the help of this framework, developers can create native user interfaces for their Python applications that run on Windows, Macs, and Linux or other Unix-like systems. Since the programming language is Python, wxPython programs are simple, easy to write and easy to understand.

2) Explain how do you create dialog box (example Message Box or Input Box) using PyAutoGUI ?

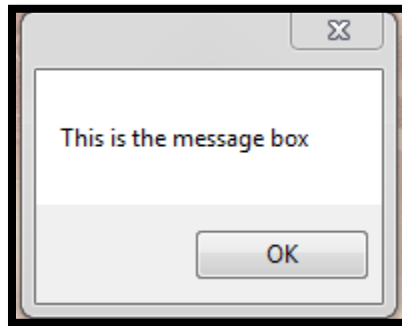
Ans:

- **PyAutoGUI** is the module of python used to automate and programmatically control the mouse and keyboard.
- Dialog boxes or message boxes are one of the key features of GUI in python.
- It prompts the user for taking an action or to give a feedback.
- There are four message box functions in python.

alert() Function

- The `alert()` function displays a message box with a text and a single **OK** button.
- It has arguments text, title and button.

```
import pyautogui
pyautogui.alert('This is the message box')
```



confirm() Function

- The `confirm()` function displays a message box with OK and Cancel buttons.
- Customization of number and text of buttons can be done.
- It will display **OK** and **Cancel** buttons by default.

```
import pyautogui
pyautogui.confirm('Please select an option')
```



To change the buttons option pass a list of buttons you want-

```
import pyautogui
pyautogui.confirm('Please select an option', buttons=['M', 'F'])
```



prompt() Function

- The `prompt()` function displays a message box with text input, **OK** and **Cancel** buttons.

```
import pyautogui  
pyautogui.prompt('Enter your name')
```



password() Function

- The `password()` function displays a message box with text input, **OK** and **Cancel** buttons.
- The characters in this are in form of * to show that they are encrypted.

```
import pyautogui  
pyautogui.password('Enter the password')
```

