Experiment No. 6

Title: GUI development using Python

Batch:A2 **Roll No:16010421063** **Experiment**

No.:6 Aim: To introduce GUI development using tkinter module in Python

---

Resources needed: Python IDE

Theory:

Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

application using Tkinter:

module
I application main window.
re widgets to the GUI application.
event loop to take action against each event triggered by the user.

win = tkinter.Tk( )

#Code to add widgets

win.mainloop( )

Widgets:

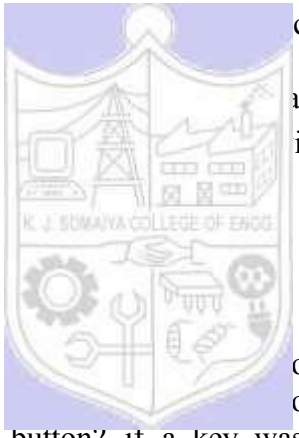
Widget class	Description
Frame	A frame is a widget that displays as a simple rectangle. Frames help to organize your user interface, often both visually and at the coding level. Frames often act as master widgets for a geometry manager like grid, which manages the slave widgets contained within the frame.
Label	A label is a widget that displays text or images, typically that users will just view but not otherwise interact with. Labels are used to identify controls or other parts of the user interface, provide textual feedback or results, etc.
Button	A button, unlike a frame or label, is very much there to interact with. Users

	press a button to perform an action. Like labels, they can display text or images but accept additional options to change their behavior.
Entry	An entry widget presents users with a single-line text field where they can type in a string value. These can be just about anything: a name, a city, a password, social security number, etc.
Text	A text widget provides users with an area so that they can enter multiple lines of text. Text widgets are part of the classic Tk widgets, not the themed Tk widgets.

More widgets can be referred from **https://tkdocs.com/tutorial/widgets.html**

Application Layout Management:

Placing widgets on the screen (and precisely where they are placed) is called as geometry management. Application layout in Tkinter is controlled with following geometry managers:



ces the element in the center and uses top bottom approach for placing of

ces the element at the given position(x,y)

ides the root window into grids and then places the elements in different

loop that receives events from the operating system. These are things like okes, mouse movement, window resizing, and so on. Tkinter takes care of oop. It figures out what widget the event applies to (did a user click on this button? If a key was pressed, which textbox had the focus?), and dispatch it accordingly. Individual widgets know how to respond to events; for example, a button might change color when the mouse moves over it and revert back when the mouse leaves.

Command Callbacks:

For those events that are most frequently customized (what good is a button without something happening when you press it?), the widget allows to specify a callback as a widget configuration option. This can be done with the command option of the button.

Example to accept message and display message when submit button is clicked:

```
import tkinter as tk

def show(event):
    val_msg = ent_msg.get()
    print(val_msg)
    lbl_output = tk.Label(text=val_msg,bg="cyan")
    lbl_output.pack(padx=5,pady=5)


="Enter Message:", width=10,bg="green",fg="white")


idth=20,bg="blue",fg="white")
,pady=5)


# btn_display =tk.Button(text="Display",command=show)
# btn_display.pack()


btn_display
=tk.Button(text="Submit")
btn_display.bind("<Button-1>",show)
btn_display.pack(padx=10,pady=10)


window.mainloop()
```

Activities:

1. Design a GUI application using tkinter and perform event handling

---

Result: (script and output)

```
from tkinter import *

window=Tk()
window.configure(bg='purple')


keep=StringVar()
name=StringVar()
password=StringVar()

def checkbox():
   print(keep.get())

def signIn():
   print("name: ",name.get())
   print("password: ",password.get())

main=Frame(window,padx=10,pady=10)
main.grid(row=0,column=0)
main.configure(bg='purple')

container=Frame(main,padx=10)
container.grid(row=0,column=1)



SignIn_label=Label(container,text="Sign in to your account",pady=10)
SignIn_label.grid(row=0,column=0)


name_entry=Entry(container,text="aryaajitnair@gmail.com",textvariable=name)
name_entry.grid(row=2,column=0,pady=5)


password_entry=Entry(container,text="password",textvariable=password,show='*'
)
```

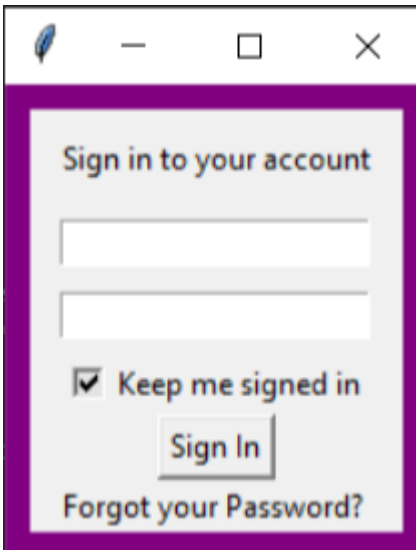
```
password_entry.grid(row=4,column=0,pady=5)

keep_check=Checkbutton(container,text='Keep me signed
in',command=checkbox,onvalue='agree',offvalue='disagree',variable=keep)
keep_check.grid(row=5,column=0)


SignIn=Button(container,text='Sign In',command=signIn)
SignIn.grid(row=6,column=0,columnspan=2)

forgot_label=Label(container,text="Forgot your Password? ")
forgot_label.grid(row=7,column=0)

window.mainloop()
```



Conclusion: (Conclusion to be based on the objectives and outcomes achieved)

Event handling is being performed at signIn button as well as the Checkbox. While doing so understood the concepts of tkinter

---

References:

1. https://tkdocs.com/
2. Daniel Arbuckle, Learning Python Testing, Packt Publishing, 1st Edition, 2014
3. Wesly J Chun, Core Python Applications Programming, O'Reilly, 3rd Edition, 2015
4. Wes McKinney, Python for Data Analysis, O'Reilly, 1st Edition, 2017
5. Albert Lukaszewsk, MySQL for Python, Packt Publishing, 1st Edition, 2010
6. Eric Chou, Mastering Python Networking, Packt Publishing, 2nd Edition, 2017