Experiment No. 5

Title: GUI development using Python

Batch: B-1 **Roll No: 16010422234** **Name: Chandana Ramesh Galgali**

Experiment No.: 5

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.

Steps to design GUI application using Tkinter:

1. Import tkinter module
2. Create the GUI application main window.
3. Add one or more widgets to the GUI application.
4. Enter the main event loop to take action against each event triggered by the user.

Example:

```
import tkinter
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:

- pack( ) – it places the element in the center and uses top bottom approach for placing of elements
- place( ) – it places the element at the given position(x,y)
- grid ( ) – it divides the root window into grids and then places the elements in different grids

Event Handling:
Tkinter runs an event loop that receives events from the operating system. These are things like button presses, keystrokes, mouse movement, window resizing, and so on. Tkinter takes care of managing this event loop. 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)

window = tk.Tk()
lbl_msg=tk.Label(text="Enter Message:", width=10,bg="green",fg="white")
lbl_msg.pack()

ent_msg = tk.Entry(width=20,bg="blue",fg="white")
ent_msg.pack(padx=5,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

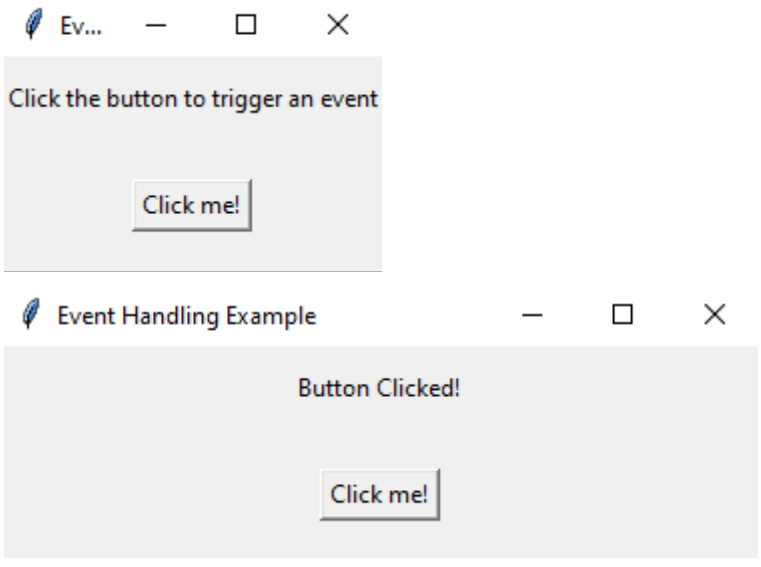
---

Result: (script and output)

Code:

```
import tkinter as tk

def on_button_click():
    label.config(text="Button Clicked!")

# Create the main window
app = tk.Tk()
app.title("Event Handling Example")

# Create a label
label = tk.Label(app, text="Click the button to trigger an event")
label.pack(pady=10)

# Create a button and associate the on_button_click function with its click event
button = tk.Button(app, text="Click me!", command=on_button_click)
button.pack(pady=20)

# Start the main event loop
app.mainloop()
```

Output:



Outcomes: Designing a graphical interface for python applications.

Conclusion: (Conclusion to be based on the objectives and outcomes achieved)
The objective of introducing GUI development using the tkinter module has been successfully met, and we are now well-prepared to embark on our journey of creating visually appealing and interactive Python applications.

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