



Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

Experiment No. 7
Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes
Date of Performance:20/03/2024
Date of Submission:27/03/2024



Experiment No. 7

Title: Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes

Aim: To study and create GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes

Objective: To introduce GUI, TKinter in python

Theory:

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI applications. Creating a GUI using tkinter is an easy task.

To create a tkinter app:

Importing the module – tkinter

Create the main window (container)

Add any number of widgets to the main window

Apply the event Trigger on the widgets.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.

**Code:**

```
import tkinter as tk
```

```
def greet():
```

```
    name = entry.get()
```

```
    gender = gender_var.get()
```

```
    greeting_label.config(text=f"Hello, {name}! You are {gender}.")
```

```
def show_info():
```

```
    info_label.config(text="Hi \n My Name is PMG")
```

```
# Create the main window
```

```
root = tk.Tk()
```

```
root.title("Simple GUI")
```

```
# Add a Label
```

```
label = tk.Label(root, text="Enter your name:")
```

```
label.pack()
```

```
# Add an Entry widget
```

```
entry = tk.Entry(root)
```

```
entry.pack()
```



Add a Label for gender selection

```
gender_label = tk.Label(root, text="Select your gender:")
```

```
gender_label.pack()
```

Variable to hold the selected gender

```
gender_var = tk.StringVar()
```

Add Male and Female Radiobuttons

```
male_radio = tk.Radiobutton(root, text="Male", variable=gender_var, value="male")
```

```
male_radio.pack()
```

```
female_radio = tk.Radiobutton(root, text="Female", variable=gender_var, value="female")
```

```
female_radio.pack()
```

Set default gender

```
gender_var.set("male")
```

Add a Button

```
button = tk.Button(root, text="Greet", command=greet)
```

```
button.pack()
```

Add a Label to display the greeting

```
greeting_label = tk.Label(root, text="")
```



```
greeting_label.pack()
```

```
# Add more buttons and labels
```

```
info_button = tk.Button(root, text="Show Info", command=show_info)
```

```
info_button.pack()
```

```
info_label = tk.Label(root, text="")
```

```
info_label.pack()
```

```
# Start the GUI event loop
```

```
root.mainloop()
```



Output:

A screenshot of a simple GUI window titled "Simple...". The window has a light gray background. At the top, there is a label "Enter your name:" followed by a text input field. Below this, there is a label "Select your gender:" followed by two radio buttons: "Male" (which is selected) and "Female". At the bottom, there are two buttons: "Greet" and "Show Info".A screenshot of the same GUI window after the "Greet" button has been clicked. The text input field now contains the name "Prathamesh". Below the gender selection, a message "Hello, Prathamesh! You are male." is displayed. The "Show Info" button remains visible at the bottom.



Conclusion:

This Python GUI, developed with Tkinter, exemplifies the core functionalities of crafting interactive interfaces. It employs basic widgets like labels, entry boxes, radio buttons, and buttons to facilitate user engagement. Users can input their name and select their gender, triggering a personalized greeting upon clicking the "Greet" button. Additionally, a "Show Info" button displays predetermined information in a label upon activation. Tkinter's intuitive design and straightforward syntax make it an accessible choice for developers aiming to create engaging GUIs in Python, with its extensive widget library and event-driven programming model providing ample flexibility.

Tkinter's simplicity and versatility empower developers to effortlessly create dynamic interfaces. Its concise syntax and comprehensive documentation make it suitable for developers of all skill levels. With robust capabilities and active community support, Tkinter remains a preferred framework for GUI development in Python, catering to diverse application needs effectively.