|  |
| --- |
| Experiment No. 8 |
| Creating GUI with python containing widgets such as labels, textbox, radio, checkboxes and custom dialog boxes |
| Date of Performance:11/3/2024 |
| Date of Submission:18/3/2024 |

**Experiment No. 8**

**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 from tkinter import messagebox class Application(tk.Tk): def \_\_init\_\_(self): super().\_\_init\_\_() self.title("Python GUI") self.create\_widgets() def create\_widgets(self):

label = tk.Label(self, text="Welcome!") label.pack() textbox = tk.Entry(self) textbox.pack() self.radio\_var = tk.StringVar(value="Option 1")

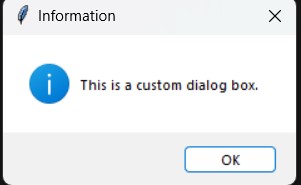
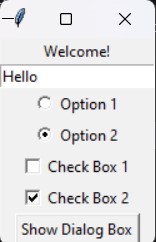
radio1 = tk.Radiobutton(self, text="Option 1", variable=self.radio\_var,

value="Option1") radio1.pack()

radio2 = tk.Radiobutton(self, text="Option 2", variable=self.radio\_var,

value="Option2") radio2.pack() self.check\_var1 = tk.IntVar() self.check\_var2 = tk.IntVar() check1 = tk.Checkbutton(self, text="Check Box 1", variable=self.check\_var1) check1.pack() check2 = tk.Checkbutton(self, text="Check Box 2", variable=self.check\_var2) check2.pack() button = tk.Button(self, text="Show Dialog Box", command=self.show\_message\_box) button.pack() def show\_message\_box(self): messagebox.showinfo("Information", "This is a custom dialog box.") if \_\_name\_\_ == "\_\_main\_\_":

app = Application() app.mainloop() **Output:**



**Conclusion:**Tkinter, as a user-friendly GUI toolkit for Python, provides an array of widgets including buttons, labels, textboxes, and more. Buttons, specifically, are fundamental interactive components in Tkinter, enabling users to trigger actions or events within the application. They are easily customizable with text, colors, and event handlers, making them essential for user interaction. With Tkinter's straightforward button implementation, developers can effortlessly create intuitive interfaces for their Python applications.