Assignment 2

Name : Shyam Nagnath Lokhande

Class : TE COMP

Roll No : 036

Batch : t1

**Title:** GUI in Python

**Problem Statement:** Design a user interface in Python

# Learning Objectives:

* To design a user interface in Python
* To learn simplicity, user centric approach of a GUI in designing

# Learning Outcomes:

A simple GUI designed using the Tkinter library in Python.

# Requirements:

Tkinter - standard GUI library for Python

# Implementation Steps:

import the Python GUI Tkinter module:

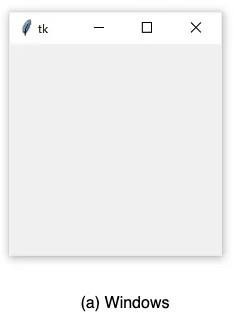
>>> import tkinter as tk

A window is an instance of Tkinter’s Tk class. Go ahead and create a new window and assign it

to the variable window:

>>> window = tk.Tk()

When we execute the above code, a new window pops up on your screen. How it looks depends on your operating system:



Adding a Widget

Use the tk.Label class to add some text to a window. Create a Label widget with the text "Hello,

Tkinter" and assign it to a variable called greeting:

>>>

>>> greeting = tk.Label(text="Hello, Tkinter") Working With Widgets

Each widget in Tkinter is defined by a class. Here are some of the widgets available:

# Widget Class Description

Label A widget used to display text on the screen

Button A button that can contain text and can perform an action when clicked

Entry A text entry widget that allows only a single line of text

Text A text entry widget that allows multiline text entry

Frame A rectangular region used to group related widgets or provide padding between widgets

# Displaying Text and Images With Label Widgets

Label widgets are used to display text or images. The text displayed by a Label widget can’t be edited by the user. It’s for display purposes only. As you saw in the example at the beginning of this tutorial, you can create a Label widget by instantiating the Label class and passing a string to the text parameter:

label button frame canvas

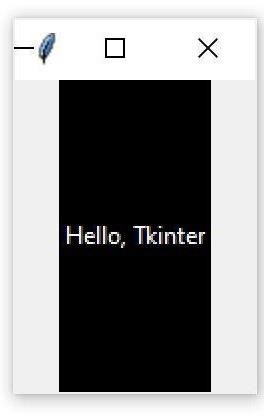
dropdown menu

Label widgets display text with the default system text color and the default system text background color. These are typically black and white, respectively, but you may see different colors if you have changed these settings in your operating system.

You can control Label text and background colors using the foreground and background parameters:

label = tk.Label( text="Hello, Tkinter", foreground="white", # Set the text color to white background="black" # Set the background color to black )

We can also control the width and height of a label with the width and height parameters:

label = tk.Label( text="Hello, Tkinter",fg="white",bg="black",width=10,height=10 Here’s what this label looks like in a window:

# Displaying Clickable Buttons With Button Widgets

button = tk.Button( text="Click me!", width=25, height=5, bg="blue", fg="yellow", )

# GUI Window Getting User Input With Entry Widgets

The following code creates a widget with a blue background, some yellow text, and a width of 50 text units:

entry = tk.Entry(fg="yellow", bg="blue", width=50)

The best way to get an understanding of Entry widgets is to create one and interact with it. Open up a Python shell and follow along with the examples in this section. First, import tkinter and create a new window:

>>> import tkinter as tk

>>> window = tk.Tk()

Now create a Label and an Entry widget:

>>> label = tk.Label(text="Name")

>>> entry = tk.Entry()

The Label describes what sort of text should go in the Entry widget. It doesn’t enforce any sort of requirements on the Entry, but it tells the user what your program expects them to put there. Youneed to .pack() the widgets into the window so that they’re visible:

>>> label.pack()

>>> entry.pack()

Here’s what that looks like:

# Python Tkinter pack() method :

* + The pack() widget is used to organize the widget in the block.
  + The position widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call. **Syntax:**

# widget.pack(options) Options: expand

**Fill Size**

# Python Tkinter place() method:

* The place() geometry manager organizes the widgets to the specific x and y coordinates.

# Syntax

**widget.place(options)**

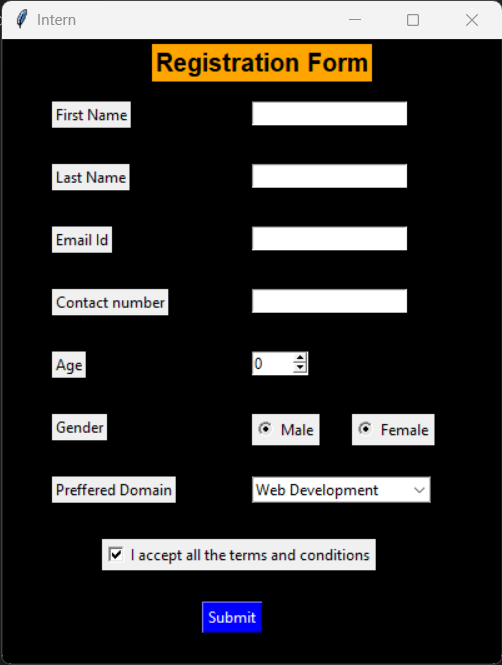
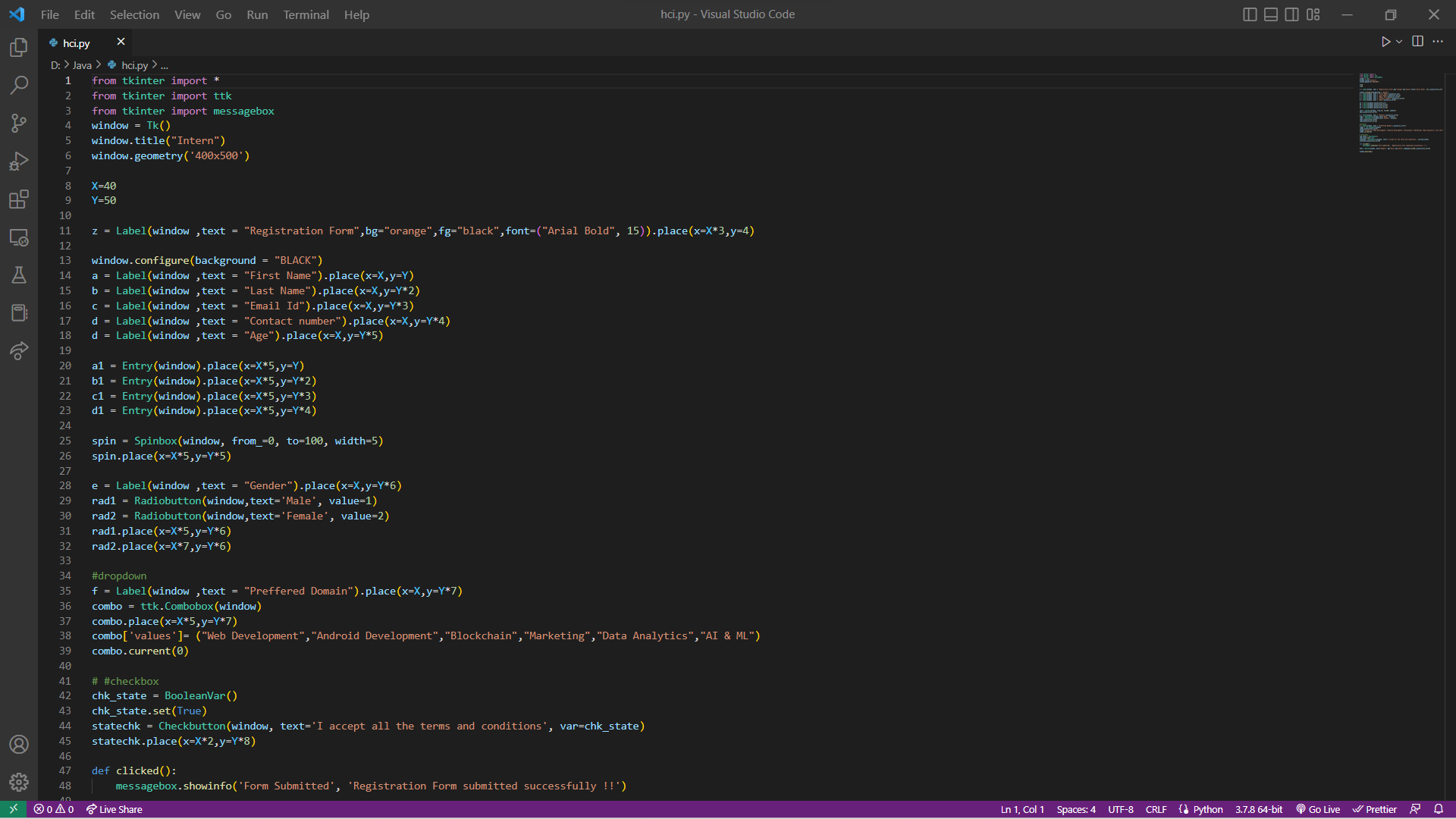
* 1. **Anchor:** It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)
  2. **bordermode:** The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.
  3. **height, width:** It refers to the height and width in pixels.
  4. **relheight, relwidth:** It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.
  5. **relx, rely:** It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.
  6. **x, y:** It refers to the horizontal and vertical offset in the pixels.

|  |  |  |
| --- | --- | --- |
| **No** | **Widget** | **Description** |
| 1 | [Button](https://www.javatpoint.com/python-tkinter-button) | The Button is used to add various kinds of buttons to the python application. |
| 2 | [Canvas](https://www.javatpoint.com/python-tkinter-canvas) | The canvas widget is used to draw the canvas on the window. |
| 3 | [Checkbutton](https://www.javatpoint.com/python-tkinter-checkbutton) | The Checkbutton is used to display the CheckButton on the window. |
| 4 | [Entry](https://www.javatpoint.com/python-tkinter-entry) | The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values. |

|  |  |  |
| --- | --- | --- |
| 5 | [Frame](https://www.javatpoint.com/python-tkinter-frame) | It can be defined as a container to which, another widget can be added and organized. |
| 6 | [Label](https://www.javatpoint.com/python-tkinter-label) | A label is a text used to display some message or information about the other widgets. |
| 7 | [ListBox](https://www.javatpoint.com/python-tkinter-listbox) | The ListBox widget is used to display a list of options to the user. |
| 8 | [Menubutton](https://www.javatpoint.com/python-tkinter-menubutton) | The Menubutton is used to display the menu items to the user. |
| 9 | [Menu](https://www.javatpoint.com/python-tkinter-menu) | It is used to add menu items to the user. |
| 10 | [Message](https://www.javatpoint.com/python-tkinter-message) | The Message widget is used to display the message-box to the user. |
| 11 | [Radiobutton](https://www.javatpoint.com/python-tkinter-radiobutton) | The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them. |
| 12 | [Scale](https://www.javatpoint.com/python-tkinter-scale) | It is used to provide the slider to the user. |
| 13 | [Scrollbar](https://www.javatpoint.com/python-tkinter-scrollbar) | It provides the scrollbar to the user so that the user can scroll the window up and down. |
| 14 | [Text](https://www.javatpoint.com/python-tkinter-text) | It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it. |
| 14 | [Toplevel](https://www.javatpoint.com/python-tkinter-toplevel) | It is used to create a separate window container. |
| 15 | [Spinbox](https://www.javatpoint.com/python-tkinter-spinbox) | It is an entry widget used to select from options of values. |
| 16 | [PanedWindow](https://www.javatpoint.com/python-tkinter-panedwindow) | It is like a container widget that contains horizontal or vertical panes. |
| 17 | [LabelFrame](https://www.javatpoint.com/python-tkinter-labelframe) | A LabelFrame is a container widget that acts as the container |
| 18 | [MessageBox](https://www.javatpoint.com/python-tkinter-messagebox) | This module is used to display the message-box in the desktop based applications. |

**Implementation:** // Source code as applicable

**Output:** Images of GUI as per the Assignment 1 GUI



**Conclusion:**

Hence we have successfully created GUI using python having various functionalities like drop down menu, frames, canvas, buttons, labels , messagebox etc