**Student Name:**

**Roll No:**

**Section:**

Lab Series No. 13.

*Lab 13 –Introduction with Tkinter.*

## Lab Objectives:

1. Introduction with Tkinter

1. **Introduction with Tkinter**

**Tkinter** is an inbuilt **Python** module used to create simple **GUI** apps. It is the most commonly used module for **GUI** apps in the **Python**.

We don't need to worry about installation of the **Tkinter** module as it comes with **Python** default.

## What Is Graphical User Interface(GUI)?

**GUI** is a desktop app which helps you to interact with the computers. They are used to perform different tasks in the desktops, laptops, other electronic devices, etc.., Here, we mainly talking about the laptops and desktops.

1. **GUI** apps like **Text-Editors** are used to create, read, update and delete different types of files.
2. **GUI** apps like **Sudoku, Chess, Solitaire, etc..,** are games which you can play.
3. **GUI** apps like **Chrome, Firefox, Microsoft Edge, etc..,** are used to surf the **Internet**.

They are some different types of **GUI** apps which we daily use on the laptops or desktops. We are going to learn how to create those type of apps.

As this is an Introduction to GUI, we will create a simple **Calculator GUI** app.

## Introduction To Tkinter in Detail

Run the following code to create a simple window with the text **Hello World!**. **Necessary Steps**:

* + 1. import the module **tkinter**.
    2. Initialize the window manager with the **tkinter.Tk()** method and assign it to a variable **window**. This method creates a blank window with close, maximize and minimize buttons.
    3. Rename the title of the window as you like with the **window.title(title\_of\_the\_window)**.

**Student Name:**

**Roll No:**

**Section:**

* + 1. **Label** is used to insert some objects into the **window**. Here, we are adding a **Label** with some text.
    2. **pack()** attribute of the widget is used to display the **widget** in a size it requires.
    3. Finally, the **mainloop()** method to display the **window** until you manually close it.
    4. That's a basic program to create a simple **GUI** interface. You will see a similar window like this.

**Program 1:** Write a Python program to make a simple GUI with the name My Greetings and you may print a message as Assalam o Alekum and Welcome to UIT.

# Code:

import tkinter

window = tkinter.Tk()

# to rename the title of the window window.title("My Greetings")

# pack is used to show the object in the window

label = tkinter.Label(window, text = "Assalam O Alekum \n Welcome to UIT!").pack()

window.mainloop()

# Output:

## Tkinter Widgets

**Widgets** are something like elements in the **HTML**. You will find different types of **widgets** to the different types of elements in the **Tkinter**.

Let's see the brief introduction to all of these widgets in the **Tkinter**.

* + 1. **Button**: **Button** widget is used to place the buttons in the **tkinter**.
    2. **Canvas**: **Canvas** is used to draw shapes in your **GUI**.
    3. **Checkbutton**: **Checkbutton** is used to create the check buttons in your application. You can select more than one option at a time.
    4. **Entry**: **Entry** widget is used to create input fields in the **GUI**.

**Student Name:**

**Roll No:**

**Section:**

* + 1. **Frame**: **Frame** is used as containers in the **tkinter**.
    2. **Label**: **Label** is used to create a single line widgets like **text**, **images**, etc..,
    3. **Menu**: **Menu** is used to create menus in the **GUI**.
    4. Here we have identified only some of the widgets that are present in **Tkinter**. You can find the complete list of widgets at official **Python** documentation.

## Geometry Management

All widgets in the **tkinter** will have some geometry measurements. These measurements give you to organize the widgets and their parent frames, windows, etc..,

**Tkinter** has the following three Geometry Manager classes.

1. **pack()**: It organizes the widgets in the block, which mean it occupies the entire available width. It's a standard method to show the widgets in the window
2. **grid()**: It organizes the widgets in table-like structure. You will see details about **grid** later in this tutorial.
3. **place()**: It's used to place the widgets at a specific position you want.

## Organizing Layout And Widgets

To arrange the layout in the **window**, we will use **Frame**, class. Let's create a simple program to see how the **Frame** works.

**Necessary Steps:**

* + 1. **Frame** is used to create the divisions in the window. You can align the frames as you like with **side** parameter of **pack()** method.
    2. **Button** is used to create a button in the window. It takes several parameters like **text**(Value of the Button), **fg**(Color of the text), **bg**(Background color), etc..,

**Note:** The parameter of any **widget** method must be where to place the widget. In the below code, we use to place in the **window**, **top\_frame**, **bottom\_frame**.

**Student Name:**

**Roll No:**

**Section:**

**Program 2:** Write a Python program to create two frames with widgets two buttons in top and two in bottom.

# Code:

import tkinter

window = tkinter.Tk()

window.title("Two Frames with Widgets")

# creating 2 frames TOP and BOTTOM top\_frame = tkinter.Frame(window).pack()

bottom\_frame = tkinter.Frame(window).pack(side = "bottom")

# now, create some widgets in the top\_frame and bottom\_frame

btn1 = tkinter.Button(top\_frame, text = "Button1", fg = "red").pack()# 'fg - foreground' is used to color the contents

btn2 = tkinter.Button(top\_frame, text = "Button2", fg = "green").pack()# 'text' is used to write the text on the Button btn3 = tkinter.Button(bottom\_frame, text = "Button2", fg = "purple").pack(side = "left")# 'side' is used to align the widgets btn4 = tkinter.Button(bottom\_frame, text = "Button2", fg = "orange").pack(side = "left")

window.mainloop()

# Output:

**Program 3:** Create a GUI with two frames one is at the left and other is the complete frame with proper labels to mentioned. Use fill parameter of pack().

# Code:

import tkinter

window = tkinter.Tk() window.title("Playing with GUI")

# creating 3 simple Labels containing some text # sufficient width

tkinter.Label(window, text = "Sufficient width", fg = "white", bg = "purple").pack()

**Student Name:**

**Roll No:**

**Section:**

# width of X

tkinter.Label(window, text = "Taking all available X width", fg = "white", bg = "green").pack(fill = "x")

# height of Y

tkinter.Label(window, text = "Taking all available Y height", fg = "white", bg = "black").pack(side = "left", fill = "y")

window.mainloop()

# Output:

## 1.6.1. Grid

**Grid** is another way to organize the **widgets**. It uses the **Matrix row column** concepts. Something like this 2 x 2 Matrix.

**Program 4:** Create a GUI using grid concept in which you can make a Login Window name the GUI as My Login Window having the Username and Password and a Checked Button to Keep me logged. Keep in mind that the username and the password will take the input as string.

# Code:

import tkinter

window = tkinter.Tk() window.title("My Login Window")

# creating 2 text labels and input labels

tkinter.Label(window, text = "Username").grid(row = 0) # this is placed in 0 0

# 'Entry' is used to display the input-field tkinter.Entry(window).grid(row = 0, column = 1) # this is placed in 0 1

**Student Name:**

**Roll No:**

**Section:**

tkinter.Label(window, text = "Password").grid(row = 1) # this is placed in 1 0

tkinter.Entry(window).grid(row = 1, column = 1) # this is placed in 1 1

# 'Checkbutton' is used to create the check buttons tkinter.Checkbutton(window, text = "Keep Me Logged In").grid(columnspan = 2) # 'columnspan' tells to take the width of 2 columns

# you can also use 'rowspan' in the similar manner window.mainloop()

# Output:

## Binding Functions

Calling functions whenever an event occurs refers to a binding function.

**Program 4:** Create a GUI with a button name it "Click Me", whenever user will click it, it will say Assalam o Alekum. You can use a function say\_Assalam\_o\_Alekum, with text having "Assalam o Alekum". Use the pack() in GUI.

# Code:

import tkinter

window = tkinter.Tk() window.title("Binding Functions")

# creating a function called say\_Assalam\_o\_Alekum() def say\_Assalam\_o\_Alekum():

tkinter.Label(window, text = "Assalam o Alekum").pack()

tkinter.Button(window, text = "Click Me!", command = say\_Assalam\_o\_Alekum).pack() # 'command' is executed when you click the button

**Student Name:**

**Roll No:**

**Section:**

#

in this above case we're calling the function 'say\_Assalam\_o\_Alekum'. window.mainloop()

# Output:

## 1.7.1 Events Capture

Another way to bind functions is using **events**. Events are something like **mousemove, mouseover, clicking, scrolling, etc..,**.

The following program also produces the same output as the above one.

* '**<Button-1>**' parameter of **bind** method is the left clicking event, i.e., when you click the left button the **bind** method call the function **say\_hi**
* <Button-1> for **left** click
* <Button-2> for **middle** click
* <Button-3> for **right** click
* Here, we are **binding** the **left** click event to a **button**. You can bind it to any other **widget** you want.
* You will have different parameters for different events

## Mouse Clicking Events

Clicking events are of 3 different types namely **leftClick**, **middleClick**, and **rightClick**. Now, you will learn how to call a particular function based on the event that occurs.

* Run the following program and click the **left, middle, right** buttons to calls a specific **function**.
* That **function** will create a new label with the mentioned text.

**Student Name:**

**Roll No:**

**Section:**

**Program 5:** Create a GUI which can capture the events that have been fired from the mouse. The mouse has right click, left click and middle click. So whenever the user will click the mouse button it will capture the event and write it on the GUI main window.

# Code:

import tkinter

window = tkinter.Tk()

window.title("Capturing the Mouse Events on GUI") #creating 3 different functions for 3 events

def left\_click(event):

tkinter.Label(window, text = "Left Click!").pack() def middle\_click(event):

tkinter.Label(window, text = "Middle Click!").pack() def right\_click(event):

tkinter.Label(window, text = "Right Click!").pack() window.bind("<Button-1>", left\_click) window.bind("<Button-2>", middle\_click) window.bind("<Button-3>", right\_click) window.mainloop()

# Output:

**Student Name:**

## Classes

**Roll No:**

**Section:**

**Classes** is handy when you're developing a large software or something that's big. Let's see how we use **Classes** in the **GUI** apps.

**Program 6:** Create a GUI which have one class name it MyGUI. The class has one method that is say\_greeting(). Whenever this method is called it will say Welcome to UIT.

# Code:

import tkinter class MyGUI:

def init (self, window):

self.text\_btn = tkinter.Button(window, text = "Click Me!", command = self.say\_greetings)

self.text\_btn.pack()

self.close\_btn = tkinter.Button(window, text = "Close", command = window.quit)

self.close\_btn.pack() def say\_greetings(self):

tkinter.Label(window, text = "Welcome to Usman Institute of Technology").pack()

window = tkinter.Tk() window.title("GUI with Class Concept") my\_gui = MyGUI(window) window.mainloop()

# Output:

**Student Name:**

**Roll No:**

**Section:**

## Drop-Down Menus

I hope all of you know what drop-down menus are. You will create drop-down menus in **tkinter** using the class **Menu**. Follow the below steps to create drop-down menus.

## Steps:-

* Create a **root menu** to insert different types of **menu options** using **tkinter.Menu(para)** and it takes a parameter where to place the **Menu**
* You have to tell the **tkinter** to initiate **Menu** using **window\_variable.config(menu = para)** and it takes a parameter called **menu** which is the **root menu** you previously defined.
* Now, creating **sub menus** using same method **tkinter.Menu(para)** and it takes the parameter **root menu**.
* **root menu.add\_cascade(para1, menu = para2)** creates the name of the **sub menu**, and it takes 2 parameters one is **label** which is the name of the **sub menu**, and another one is **menu** which is **sub menu**.
* **sub menu.add\_command()** adds an option to the **sub menu**.
* **sub menu.add\_separator()** adds a separator Let's see the example to understand it fully.

**Program 7:** Create a GUI which have drop down menus of File having New File, Open File, Exit and next menu will have Edit in which Undo and Redo as drop down.

# Code:

import tkinter

window = tkinter.Tk() window.title("My GUI with Menu") def function():

pass

# creating a root menu to insert all the sub menus root\_menu = tkinter.Menu(window) window.config(menu = root\_menu)

# creating sub menus in the root menu

**Student Name:**

**Roll No:**

**Section:**

file\_menu = tkinter.Menu(root\_menu) # it intializes a new su menu in the root menu

root\_menu.add\_cascade(label = "File", menu = file\_menu) # it creates the name of the sub menu

file\_menu.add\_command(label = "New file.....", command = function) # it adds a option to the sub menu 'command' parameter is used to do some action

file\_menu.add\_command(label = "Open files", command = function) file\_menu.add\_separator() # it adds a line after the 'Open files' option

file\_menu.add\_command(label = "Exit", command = window.quit)

# creting another sub menu edit\_menu = tkinter.Menu(root\_menu)

root\_menu.add\_cascade(label = "Edit", menu = edit\_menu) edit\_menu.add\_command(label = "Undo", command = function) edit\_menu.add\_command(label = "Redo", command = function)

window.mainloop()

# Output:

**Student Name:**

**Roll No:**

**Section:**

## Alert Box

You can create alert boxes in the **tkinter** using **messagebox** method. You can also create **questions** using the **messasgebox** method.

**Program 8:** Create a GUI which will generate an alert message when you execute the GUI and give you option that do you love python if you press yes it will write the message otherwise it will say you don’t love python on GUI.

# Code:

import tkinter

import tkinter.messagebox

window = tkinter.Tk() window.title("Alert Message GUI")

# creating a simple alert box

tkinter.messagebox.showinfo("Alert Message", "This is just a alert message!")

# creating a question to get the response from the user [Yes or No Question]

response = tkinter.messagebox.askquestion("Simple Question", "Do you love Python?")

# If user clicks 'Yes' then it returns 1 else it returns 0 if response == 1:

tkinter.Label(window, text = "You love Python!").pack() else:

tkinter.Label(window, text = "You don't love Python!").pack() window.mainloop()

# Output:

**Student Name:**

**Roll No:**

**Section:**

## Simple Shapes

You are going to draw some basic shapes with the **Canvas** provided by **tkinter** in **GUI**. You will see the following shapes in your **GUI** window. Just run **dir(tkinter.Canvas)** to see all the available methods for creating different shapes.

**Program 9:** Create a GUI which will use tkinter canvas method to generate the shapes such as a

rectangle with green color, a line which is black color and another line which is red. Now you can use a canvas.delete() to delete line 1.

**Code:**

import tkinter

window = tkinter.Tk() window.title("Sahpes on your GUI")

# creating the 'Canvas' area of width and height 500px canvas = tkinter.Canvas(window, width = 500, height = 500) canvas.pack()

# 'create\_line' is used to create a line. Parameters:- (starting x- point, starting y-point, ending x-point, ending y-point)

line1 = canvas.create\_line(25, 25, 250, 150) # parameter:- (fill = color\_name)

line2 = canvas.create\_line(25, 250, 250, 150, fill = "red")

# 'create\_rectangle' is used to create rectangle. Parameters:- (starting x-point, starting y-point, width, height, fill)

# starting point the coordinates of top-left point of rectangle rect = canvas.create\_rectangle(500, 25, 175, 75, fill = "green")

**Student Name:**

**Roll No:**

**Section:**

# you 'delete' shapes using delete method passing the name of the variable as parameter.

canvas.delete(line1)

# you 'delete' all the shapes by passing 'ALL' as parameter to the 'delete' method

# canvas.delete(tkinter.ALL)

window.mainloop()

# Output:

## Images And Icons

You can add **Images** and **Icons** using **PhotoImage** method.Let's how it works. You can see the icon in the **GUI**.

**Program 10:** Create a GUI which will use tkinter PhotoImage method to show the image on the GUI.

**Code:**

import tkinter import tkinter

window = tkinter.Tk() window.title("Image or Logo on GUI")

# taking image from the directory and storing the source in a variable icon = tkinter.PhotoImage(file = "UITlogo.png")

**Student Name:**

**Roll No:**

**Section:**

# displaying the picture using a 'Label' by passing the 'picture' variriable to 'image' parameter

label = tkinter.Label(window, image = icon) label.pack()

window.mainloop()

# Output:

## Creating Calculator

Every **GUI** apps include two steps.

* + 1. Creating User Interface
    2. Adding functionalities to the **GUI**
    3. Let's start creating **Calculator**.

**Program 11:** Write a program in python that can create a GUI for a Simple Calculator.

from tkinter import \* # creating basic window window = Tk()

window.geometry("312x324") # size of the window width:- 500, height:- 375

window.resizable(0, 0) # this prevents from resizing the window window.title("Calcualtor")

**Student Name:**

**Roll No:**

**Section:**

################################### functions ######################################

# 'btn\_click' function continuously updates the input field whenever you enters a number

def btn\_click(item): global expression

expression = expression + str(item) input\_text.set(expression)

# 'btn\_clear' function clears the input field def btn\_clear():

global expression expression = "" input\_text.set("")

# 'btn\_equal' calculates the expression present in input field def btn\_equal():

global expression

result = str(eval(expression)) # 'eval' function evalutes the string expression directly

# you can also implement your own function to evalute the expression istead of 'eval' function

input\_text.set(result) expression = ""

expression = ""

# 'StringVar()' is used to get the instance of input field input\_text = StringVar()

**Student Name:**

**Roll No:**

**Section:**

# creating a frame for the input field

input\_frame = Frame(window, width = 312, height = 50, bd = 0, highlightbackground = "black", highlightcolor = "black", highlightthickness = 1)

input\_frame.pack(side = TOP)

# creating a input field inside the 'Frame'

input\_field = Entry(input\_frame, font = ('arial', 18, 'bold'), textvariable = input\_text, width = 50, bg = "#eee", bd = 0, justify = RIGHT)

input\_field.grid(row = 0, column = 0)

input\_field.pack(ipady = 10) # 'ipady' is internal padding to increase the height of input field

# creating another 'Frame' for the button below the 'input\_frame' btns\_frame = Frame(window, width = 312, height = 272.5, bg = "grey") btns\_frame.pack()

# first row

clear = Button(btns\_frame, text = "C", fg = "black", width = 32, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_clear()).grid(row = 0, column = 0, columnspan = 3, padx = 1, pady

= 1)

divide = Button(btns\_frame, text = "/", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_click("/")).grid(row = 0, column = 3, padx = 1, pady = 1)

# second row

seven = Button(btns\_frame, text = "7", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(7)).grid(row = 1, column = 0, padx = 1, pady = 1)

eight = Button(btns\_frame, text = "8", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(8)).grid(row = 1, column = 1, padx = 1, pady = 1)

**Student Name:**

**Roll No:**

**Section:**

nine = Button(btns\_frame, text = "9", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(9)).grid(row = 1, column = 2, padx = 1, pady = 1)

multiply = Button(btns\_frame, text = "\*", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_click("\*")).grid(row = 1, column = 3, padx = 1, pady = 1)

# third row

four = Button(btns\_frame, text = "4", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(4)).grid(row = 2, column = 0, padx = 1, pady = 1)

five = Button(btns\_frame, text = "5", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(5)).grid(row = 2, column = 1, padx = 1, pady = 1)

six = Button(btns\_frame, text = "6", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(6)).grid(row = 2, column = 2, padx = 1, pady = 1)

minus = Button(btns\_frame, text = "-", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_click("-")).grid(row = 2, column = 3, padx = 1, pady = 1)

# fourth row

one = Button(btns\_frame, text = "1", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(1)).grid(row = 3, column = 0, padx = 1, pady = 1)

two = Button(btns\_frame, text = "2", fg = "black", width = 10, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(2)).grid(row = 3, column = 1, padx = 1, pady = 1)

three = Button(btns\_frame, text = "3", fg = "black", width = 10, height = 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(3)).grid(row = 3, column = 2, padx = 1, pady = 1)

plus = Button(btns\_frame, text = "+", fg = "black", width = 10, height

= 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_click("+")).grid(row = 3, column = 3, padx = 1, pady = 1)

**Student Name:**

# fourth row

**Roll No:**

**Section:**

zero = Button(btns\_frame, text = "0", fg = "black", width = 21, height

= 3, bd = 0, bg = "#fff", cursor = "hand2", command = lambda: btn\_click(0)).grid(row = 4, column = 0, columnspan = 2, padx = 1, pady

= 1)

point = Button(btns\_frame, text = ".", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_click(".")).grid(row = 4, column = 2, padx = 1, pady = 1)

equals = Button(btns\_frame, text = "=", fg = "black", width = 10, height = 3, bd = 0, bg = "#eee", cursor = "hand2", command = lambda: btn\_equal()).grid(row = 4, column = 3, padx = 1, pady = 1)

window.mainloop()

# Output:

**Programming Exercise**

* + - 1. Write a program which will make a scientific calculator.
      2. Give proper color to your calculator
      3. Design any creative application from the exercises that have been taught to you in this lab. Such as POS systems for any Pharmacy, or any grocery store, or BBQ restaurant, or Pizza Hut or any application which you like.