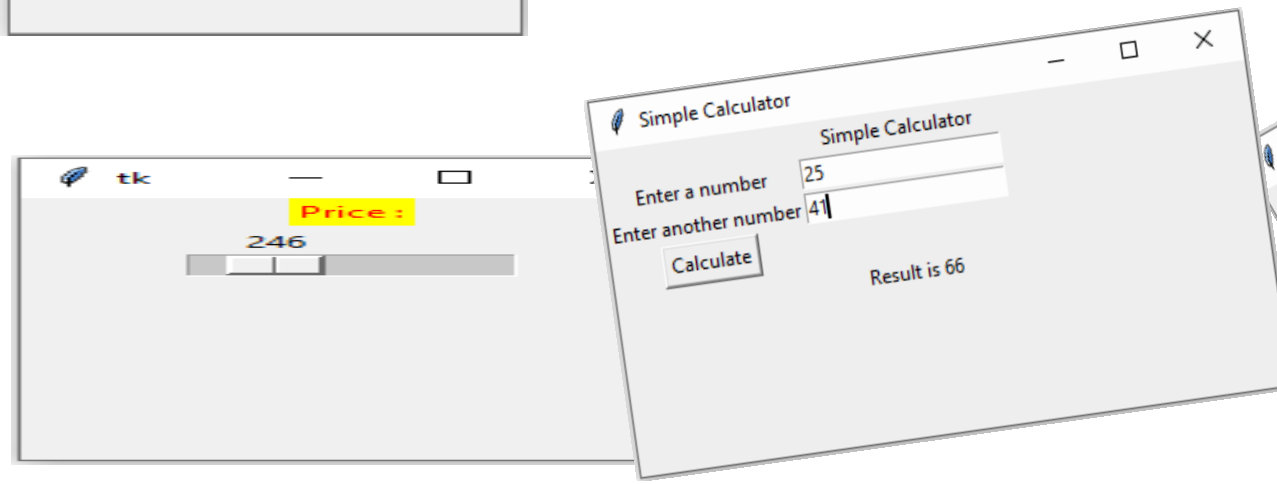
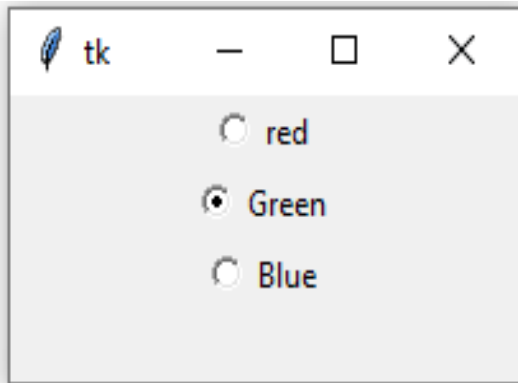
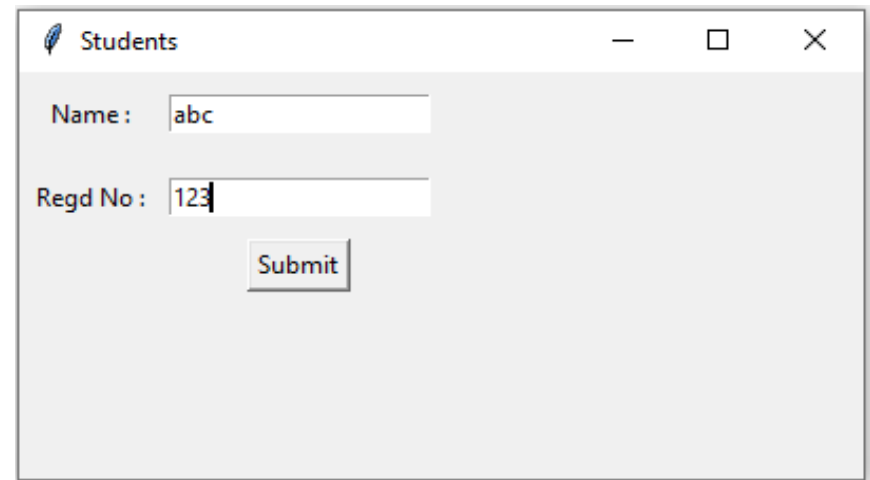


GUI Programming in Python



Introduction:

- A graphical user interface is an application that has buttons, windows, and lots of other widgets that the user can use to interact with your application.
- A good example would be a web browser. It has buttons, tabs, and a main window where all the content loads.
- In GUI programming, a **top-level root** windowing object contains all of the **little windowing objects** that will be part of your complete GUI application.
- These windowing objects can be text labels, buttons, list boxes, etc. These individual little GUI components are known as **widgets**.



- Python offers multiple options for developing GUI (Graphical User Interface). The most commonly used **GUI method** is **tkinter**.
- **Tkinter** is the easiest among all to get started with. It is Python's standard GUI (Graphical User Interface) package. It is the most commonly used toolkit for **GUI Programming** in Python
- since Tkinter is the Python interface to Tk (Tea Kay), it can be pronounced as **Tea-Kay-inter**. i.e **tkinter = t k inter**.

tkinter - GUI for Python:

- Python provides the standard library **tkinter** for creating the graphical user interface for **desktop based applications**.
- Developing desktop based applications with **tkinter** is not a complex task.
- A Tkinter window application can be created by using the following steps.
 1. **Import** the **tkinter** module.
 2. Create the **main application window**.
 3. Add the **widgets** like labels, buttons, frames, etc. to the window.
 4. Call the **main event loop** so that the actions can take place on the user's computer screen.

1. 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** is '**tkinter**'.

import tkinter (or) **from tkinter import ***

2. After importing **tkinter** module we need to create a main window, tkinter offers a method '**Tk()**' to create **main window**. The basic code used to create the main window of the application is:

top = tkinter.Tk() (or) **top=Tk()**

3. After creating main window, we need to **add components** or **widgets** like labels, buttons, frames, etc.

4. After adding widgets to **main window**, we need to run the application, tkinter offers a method '**mainloop()**' to run application. The basic code used to run the application is:

top.mainloop ()

Example: tkndemo.py

```
import tkinter
```

```
top = tkinter.Tk()      #creating the application main window.
```

```
top.title("Welcome")      #title of main window
```

```
top.geometry("400x300")      #size of main window
```

```
top.mainloop()      #calling the event main loop
```

Output:

```
>>> python tkndemo.py      → Title of window
```



Main Window
(400x300)

- tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows.

Tkinter provides the following geometry methods

1. pack () method:

The **pack()** method is used to organize components or widgets in main window.

Syntax:

widget.pack (options)

The possible options are

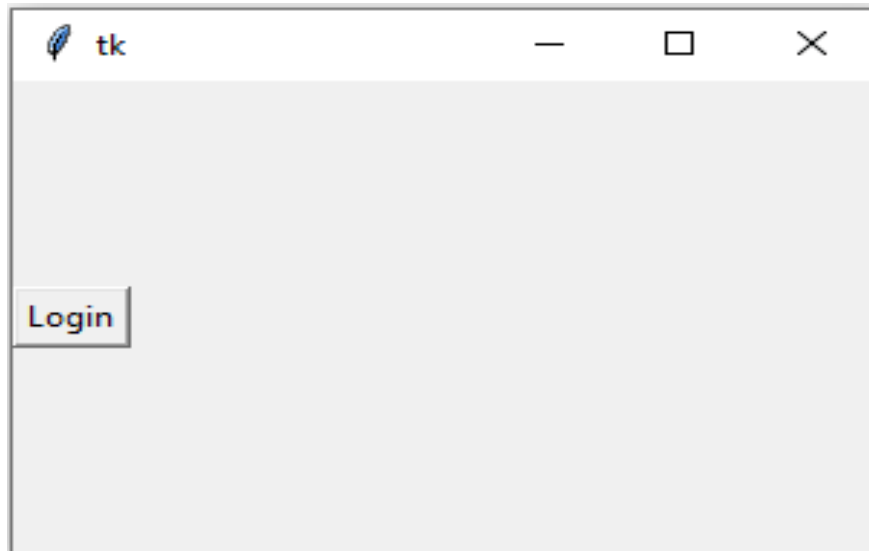
side: it represents the side to which the widget is to be placed on the window. Side may be **LEFT** or **RIGHT** or **TOP(default)** or **BOTTOM**.

Example: tknpack.py

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
btn1 = Button(top, text = "Login")  
btn1.pack( side = LEFT)  
top.mainloop()
```

Output:

```
>>> python tknpack.py
```



2. grid() method:

The **grid()** method organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call.

This is a more organized way to place the widgets to the python application.

Syntax:

widget.grid (options)

The possible options are

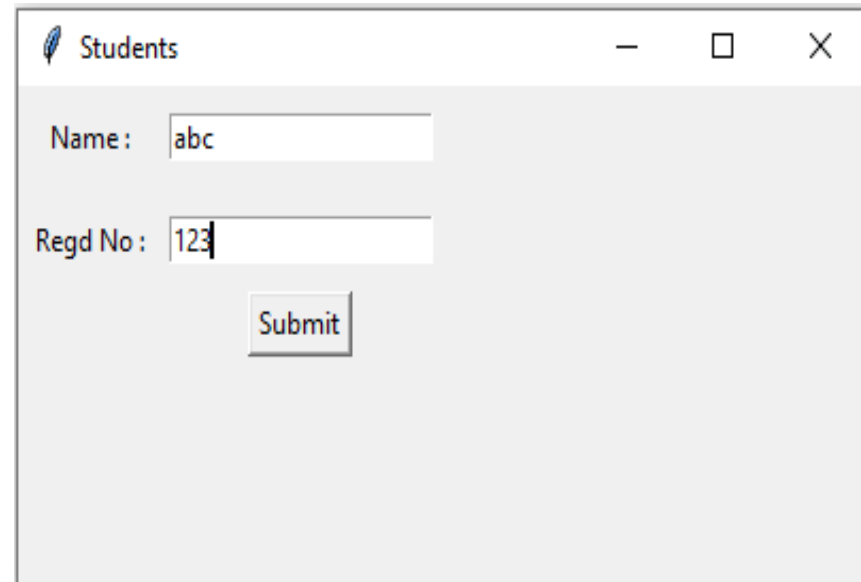
- **Column**
The column number in which the widget is to be placed. The leftmost column is represented by **0**.
- **padx, pady**
It represents the number of pixels to pad the widget outside the widget's border.
- **row**
The row number in which the widget is to be placed. The topmost row is represented by **0**.

Example: tkngrid.py

```
from tkinter import *  
parent = Tk()  
parent.title("Students")  
parent.geometry("300x200")  
name = Label(parent,text = "Name : ")  
name.grid(row = 0, column = 0,pady=10,padx=5)  
e1 = Entry(parent)  
e1.grid(row = 0, column = 1)  
regno = Label(parent,text = "Regd No : ")  
regno.grid(row = 1, column = 0,pady=10,padx=5)  
e2 = Entry(parent)  
e2.grid(row = 1, column = 1)  
btn = Button(parent, text = "Submit")  
btn.grid(row = 3, column = 1)  
parent.mainloop()
```

Output:

```
>>>python tkngrid.py
```



3. place() method:

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

Syntax:

widget.place(x,y)

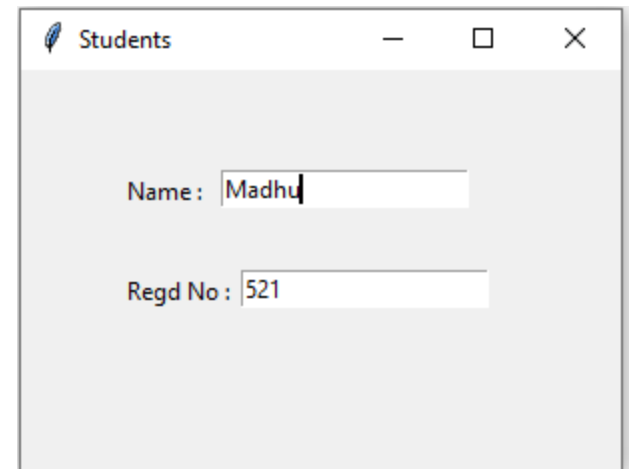
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Example: tknplace.py

```
from tkinter import *  
parent = Tk()  
parent.title("Students")  
parent.geometry("300x200")  
name = Label(parent,text = "Name : ")  
name.place(x=50,y=50)  
e1 = Entry(parent)  
e1.place(x=100,y=50)  
regno = Label(parent,text = "Regd No : ")  
regno.place(x=50,y=100)  
e2 = Entry(parent)  
e2.place(x=110,y=100)  
parent.mainloop()
```

Output:

```
>>>python tknplace.py
```



- **Tkinter widgets or components:**

Tkinter supports various widgets or components to build GUI application in python.

Widget	Description
Button	Creates various buttons in Python Application.
Checkbutton	Select one or more options from multiple options.(Checkbox)
Entry	Allows the user to enter single line of text(Textbox)
Frame	Acts like a container which can be used to hold the other widgets
Label	Used to display non editable text on window
Listbox	Display the list items, The user can choose one or more items.
Radiobutton	Select one option from multiple options.
Text	Allows the user to enter single or multiple line of text(Textarea)
Scale	Creates the graphical slider, the user can slide through the range of values
Toplevel	Used to create and display the top-level windows(Open a new window)

❖ Button Widget in Tkinter:

- The Button is used to add various kinds of buttons to the python application. We can also associate a method or function with a button which is called when the button is pressed.

Syntax: **name = Button(parent, options)**

The options are

- **activebackground:**It represents the background of the button when it is active.
- **activeforeground:**It represents the font color of the button when it is active..
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the button.
- **command:**It is set to the function call which is scheduled when the function is called.
- **text:** It is set to the text displayed on the button.
- **fg:** Foreground color of the button.
- **height:**The height of the button.
- **padx:**Additional padding to the button in the horizontal direction.
- **pady:**Additional padding to the button in the vertical direction.
- **width:**The width of the button.

Example: **btndemo1.py**

```
from tkinter import *
from tkinter import messagebox

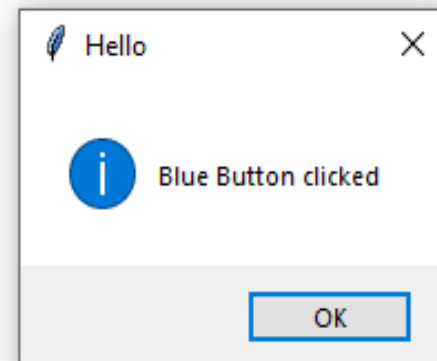
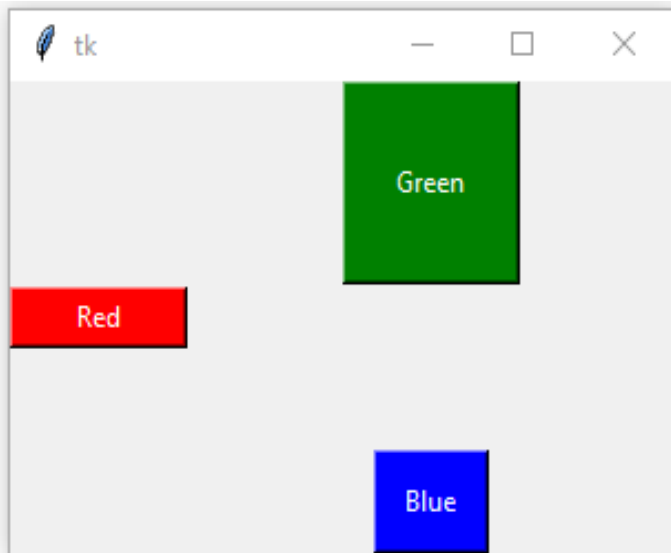
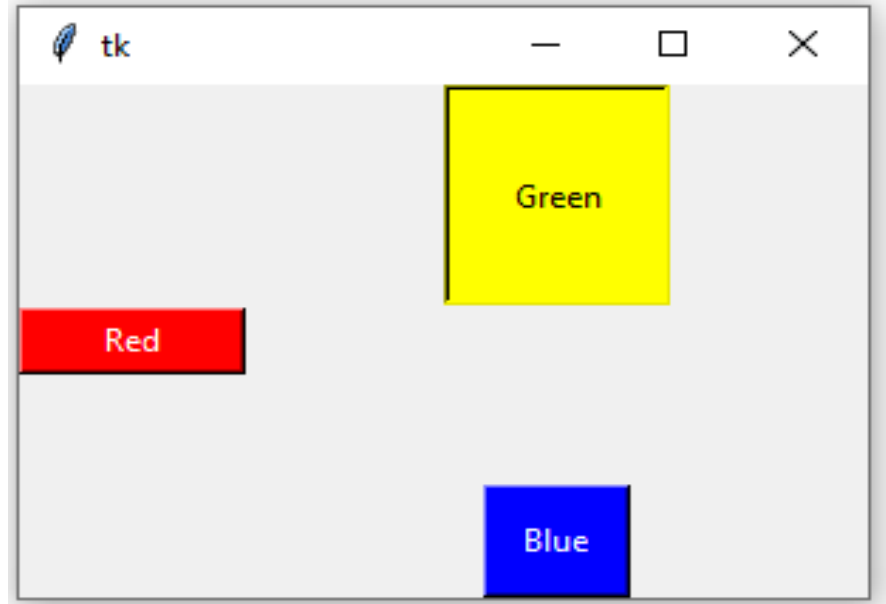
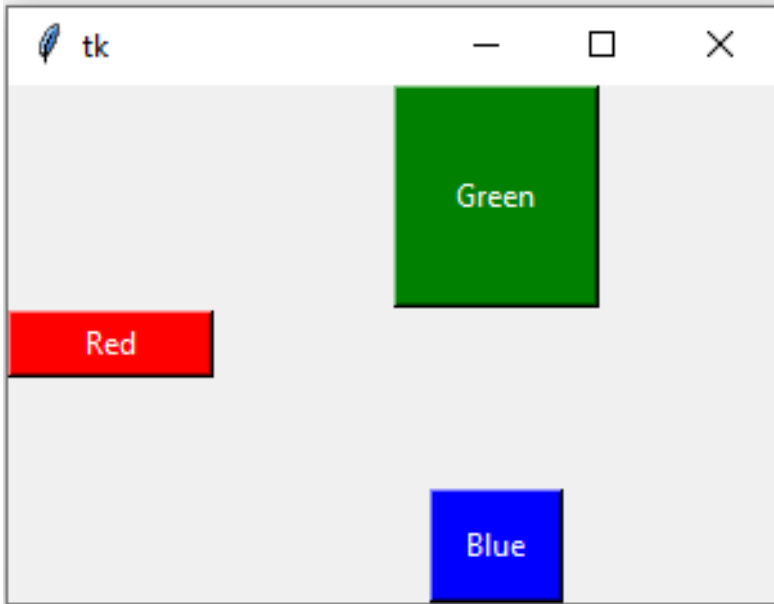
top = Tk()
top.geometry("300x200")

def fun():
    messagebox.showinfo("Hello", "Blue Button clicked")

btn1 = Button(top, text = "Red",bg="red",fg="white",width=10)
btn1.pack( side = LEFT)
btn2 = Button(top, text = "Green",bg="green",fg="white",width=10,height=5,
    activebackground="yellow")
btn2.pack( side = TOP)
btn3 = Button(top, text ="Blue",bg="blue",fg="white",padx=10,pady=10,
    command=fun)
btn3.pack( side = BOTTOM)
top.mainloop()
```

Output:

```
>>>python btndemo1.py
```



❖ Checkbutton Widget in Tkinter:

- The Checkbutton is used to display the CheckButton on the window. The Checkbutton is mostly used to provide many choices to the user among which, the user needs to choose the one. It generally implements many of many selections.

Syntax: **name = Checkbutton(parent, options)**

The options are

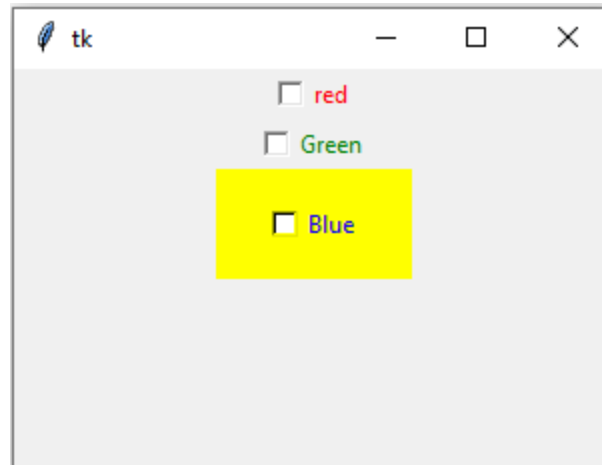
- **activebackground:** It represents the background of the Checkbutton when it is active.
- **activeforeground:** It represents the font color of the Checkbutton when when it is active.
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Checkbutton.
- **command:** It is set to the function call which is scheduled when the function is called.
- **text:** It is set to the text displayed on the Checkbutton.
- **fg:** Foreground color of the Checkbutton.
- **height:** The height of the Checkbutton.
- **padx:** Additional padding to the Checkbutton in the horizontal direction.
- **pady:** Additional padding to the Checkbutton in the vertical direction.
- **width:** The width of the Checkbutton.

Example: **chbtndemo.py**

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
cbtn1 = Checkbutton(top, text="red",fg="red")  
cbtn1.pack()  
cbtn2 = Checkbutton(top, text="Green",fg="green",activebackground="orange")  
cbtn2.pack()  
cbtn3 = Checkbutton(top, text="Blue",fg="blue",bg="yellow",width=10,height=3)  
cbtn3.pack()  
top.mainloop()
```

Output:

>>>python chbtndemo.py



❖ Entry Widget in Tkinter:

- The Entry widget is used to provide the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user.

Syntax: **name = Entry(parent, options)**

The options are

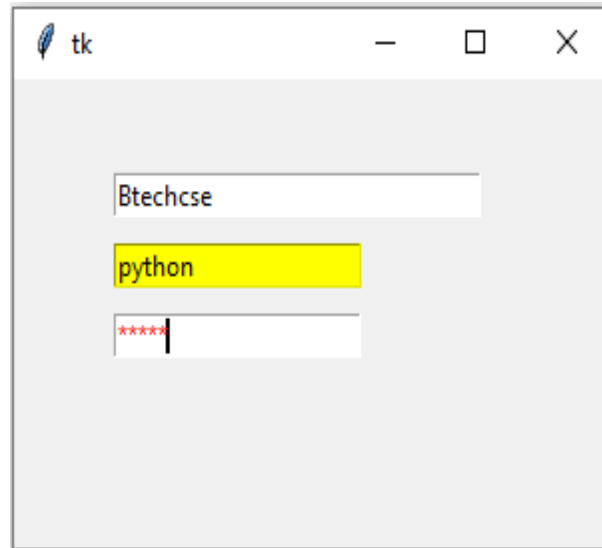
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Entry.
- **show:** It is used to show the entry text of some other type instead of the string. For example, the password is typed using stars (*).
- **fg:** Foreground color of the Entry.
- **width:** The width of the Entry.

Example: **entrydemo.py**

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
enty0 = Entry(top,width="30")  
enty0.place(x=50,y=40)  
enty1 = Entry(top,bg="yellow")  
enty1.place(x=50,y=70)  
enty2 = Entry(top,fg="red",show="*")  
enty2.place(x=50,y=100)  
top.mainloop()
```

Output:

>>>python entrydemo.py



❖ Frame Widget in Tkinter:

- Frame widget is used to organize the group of widgets. It acts like a container which can be used to hold the other widgets. The rectangular areas of the screen are used to organize the widgets to the python application.

Syntax: **name = Frame(parent, options)**

The options are

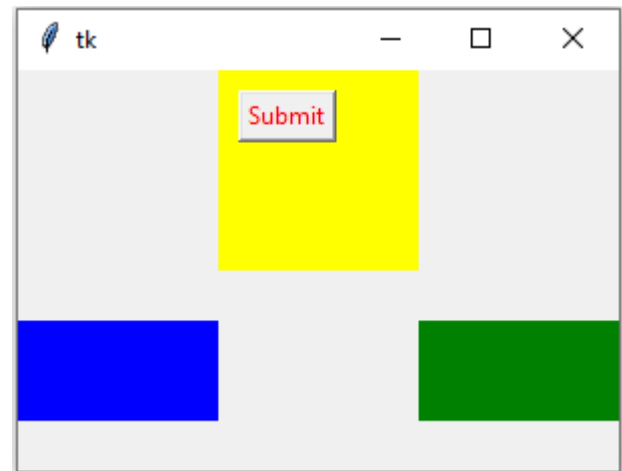
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the frame.
- **width:** The width of the frame.
- **height:** The height of the frame.

Example: **framedemo.py**

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
tframe = Frame(top,width="100",height="100",bg="yellow")  
tframe.pack()  
lframe = Frame(top,width="100",height="50",bg="blue")  
lframe.pack(side = LEFT)  
rframe = Frame(top,width="100",height="50",bg="green")  
rframe.pack(side = RIGHT)  
btn1 = Button(tframe, text="Submit", fg="red")  
btn1.place(x=10,y=10)  
top.mainloop()
```

Output:

>>>python framedemo.py



❖ Label Widget in Tkinter:

- The Label is used to specify the container box where we can place the text or images.

Syntax: **name = Label(parent, options)**

The options are

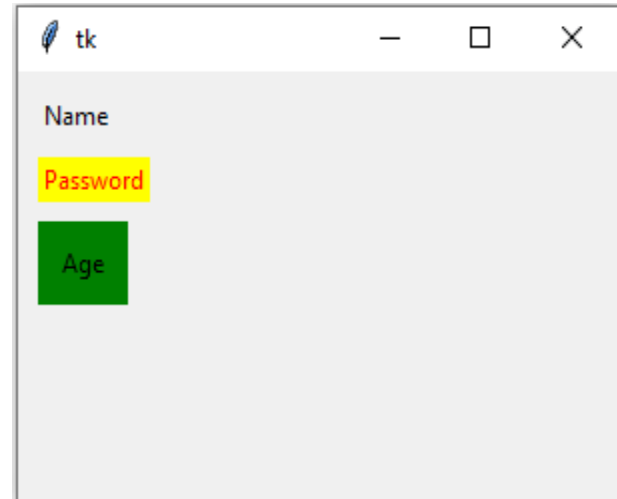
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the label.
- **text:** It is set to the text displayed on the label.
- **fg:** Foreground color of the label.
- **height:** The height of the label.
- **image:** It is set to the image displayed on the label.
- **padx:** Additional padding to the label in the horizontal direction.
- **pady:** Additional padding to the label in the vertical direction.
- **width:** The width of the label.

Example: **labeldemo.py**

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
lbl1 = Label(top, text="Name")  
lbl1.place(x=10,y=10)  
lbl2 = Label(top, text="Password", fg="red",bg="yellow")  
lbl2.place(x=10,y=40)  
lbl3 = Label(top, text="Age", padx=10,pady=10,bg="green")  
lbl3.place(x=10,y=70)  
top.mainloop()
```

Output:

>>>python labeldemo.py



❖ Listbox Widget in Tkinter:

- The Listbox widget is used to display the list items to the user. We can place only text items in the Listbox. The user can choose one or more items from the list.

Syntax: **name = Listbox(parent, options)**

The options are

- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the listbox.
- **fg:** Foreground color of the listbox.
- **width:** The width of the listbox.
- **height:** The height of the listbox.

The following method is associated with the Listbox to insert list item to listbox at specified index.i.e, **insert ()**.

Syntax:

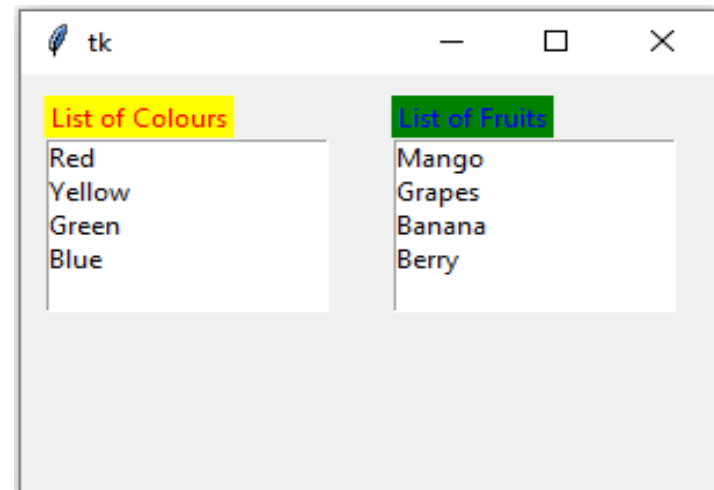
Listbox.insert (index, item)

Example: listboxdemo.py

```
from tkinter import *
top = Tk()
top.geometry("300x200")
lbl1 = Label(top, text="List of Colours",fg="red",bg="yellow")
lbl1.place(x=10,y=10)
lb = Listbox(top,height=5)
lb.insert(1,"Red")
lb.insert(2, "Yellow")
lb.insert(3, "Green")
lb.insert(4, "Blue")
lb.place(x=10,y=30)
lbl2 = Label(top, text="List of Fruits",fg="blue",bg="green")
lbl2.place(x=160,y=10)
lb1 = Listbox(top,height=5)
lb1.insert(1,"Mango")
lb1.insert(2, "Grapes")
lb1.insert(3, "Banana")
lb1.insert(4, "Berry")
lb1.place(x=160,y=30)
top.mainloop()
```

Output:

>>>python listboxdemo.py



❖ Radiobutton Widget in Tkinter:

- The Radiobutton widget is used to select one option among multiple options. 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.

Syntax: **name = Radiobutton(parent, options)**

The options are

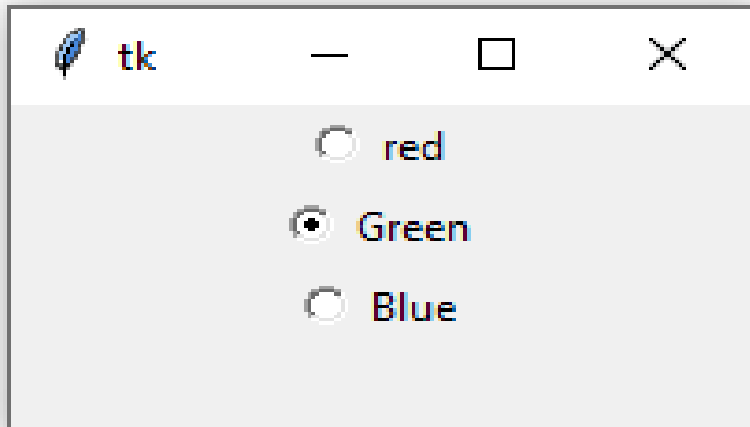
- **activebackground:** It represents the background of the Radiobutton when it is active.
- **activeforeground:** It represents the font color of the Radiobutton when when it is active.
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Radiobutton.
- **command:** It is set to the function call which is scheduled when the function is called.
- **text:** It is set to the text displayed on the Radiobutton.
- **fg:** Foreground color of the Radiobutton.
- **height:** The height of the Radiobutton.
- **padx:** Additional padding to the Radiobutton in the horizontal direction.
- **pady:** Additional padding to the Radiobutton in the vertical direction.
- **width:** The width of the Radiobutton.
- **Variable:** It is used to keep track of the user's choices. It is shared among all the radiobuttons.

Example: rbtdemo.py

```
from tkinter import *  
top = Tk()  
top.geometry("200x100")  
radio = IntVar()  
rbtn1 = Radiobutton(top, text="red",variable=radio,value="1")  
rbtn1.pack()  
rbtn2 = Radiobutton(top, text="Green",variable=radio,value="2")  
rbtn2.pack()  
rbtn3 = Radiobutton(top, text="Blue",variable=radio,value="3")  
rbtn3.pack()  
top.mainloop()
```

Output:

>>>python rbtdemo.py



❖ Text Widget in Tkinter:

- The Text widget allows the user to enter multiple lines of 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.

Syntax: **name = Text(parent, options)**

The options are

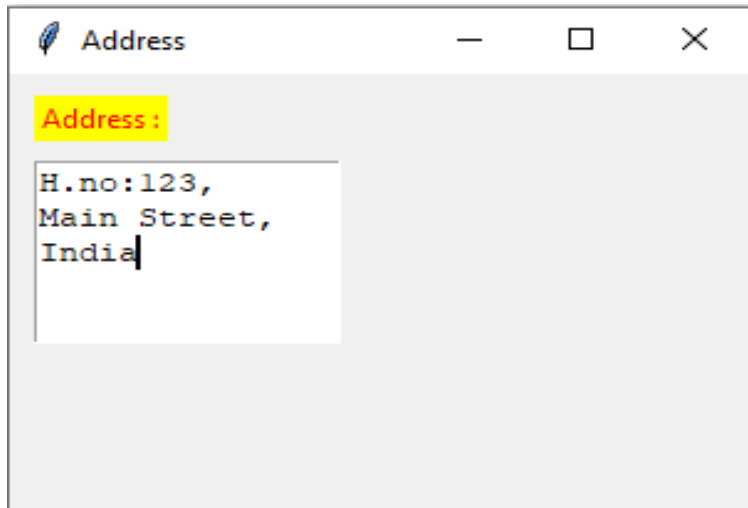
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Text.
- **show:** It is used to show the entry text of some other type instead of the string. For example, the password is typed using stars (*).
- **fg:** Foreground color of the Text.
- **width:** The width of the Text.
- **height:** The vertical dimension of the widget in lines.

Example: textdemo.py

```
from tkinter import *  
top = Tk()  
top.title("Address")  
top.geometry("300x200")  
lbl=Label(top,text="Address :",fg="red",bg="yellow")  
lbl.place(x=10,y=10)  
txt=Text(top,width=15,height=5)  
txt.place(x=10,y=40)  
top.mainloop()
```

Output:

```
>>>python textdemo.py
```



❖ Scale Widget in Tkinter:

- The Text widget allows the user to enter multiple lines of 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.

Syntax: **name = Scale(parent, options)**

The options are

- **activebackground:** It represents the background of the Scale when it is active.
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Scale.
- **command:** It is set to the function call which is scheduled when the function is called.
- **fg:** Foreground color of the Scale.
- **from_:** It is used to represent one end of the widget range.
- **to:** It represents a float or integer value that specifies the other end of the range represented by the scale.
- **orient:** It can be set to horizontal or vertical depending upon the type of the scale.

Example: scaledemo.py

```
from tkinter import *  
top = Tk()  
top.geometry("200x200")  
lbl=Label(top,text="Price :",bg="yellow",fg="red")  
lbl.pack()  
scale = Scale( top, from_ = 100, to = 1000, orient = HORIZONTAL)  
scale.pack(anchor=CENTER)  
top.mainloop()
```

Output:

>>>python scaledemo.py



❖ Toplevel Widget in Tkinter:

- The Toplevel widget is used to create and display the toplevel windows which are directly managed by the window manager.

Syntax: **name = Toplevel(options)**

The options are

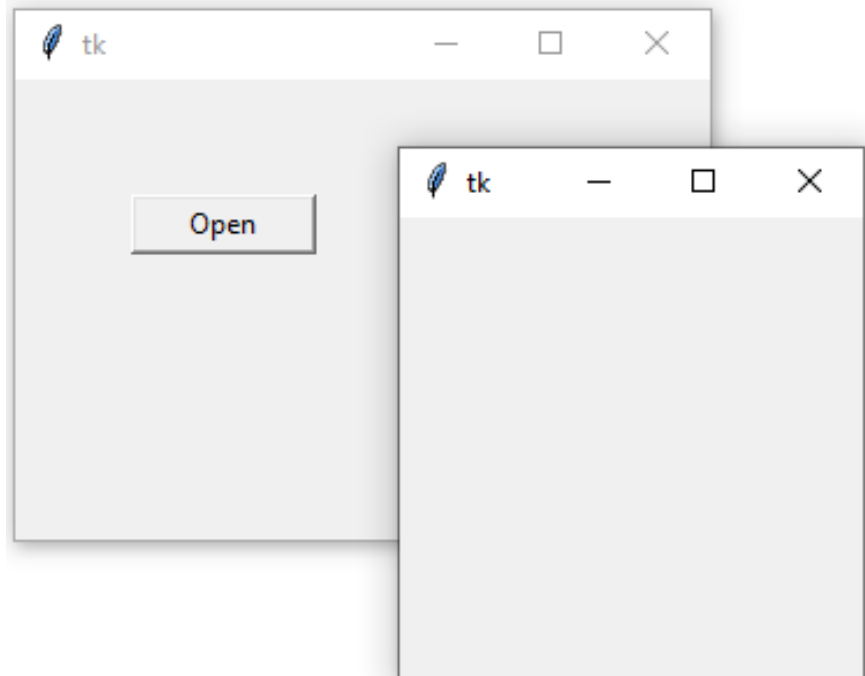
- **bd:** It represents the border width in pixels.
- **bg:** It represents the background color of the Toplevel.
- **fg:** Foreground color of the Toplevel.
- **width:** The width of the Toplevel.
- **height:** The vertical dimension of the widget in lines.

Example: topleveldemo.py

```
from tkinter import *  
top = Tk()  
top.geometry("300x200")  
def fun():  
    chld = Toplevel(top)  
    chld.mainloop()  
btn1 = Button(top, text = "Open",width=10,command=fun)  
btn1.place(x=50,y=50)  
top.mainloop()
```

Output:

```
>>>python topleveldemo.py
```



Example: simplecalc.py

```
import tkinter as tk
from functools import partial
def call_result(label_result, n1, n2):
    num1 = (n1.get())
    num2 = (n2.get())
    result = int(num1)+int(num2)
    label_result.config(text="Result is %d" % result)
    return
root = tk.Tk()
root.geometry('400x200+100+200')
root.title('Simple Calculator')
number1 = tk.StringVar()
number2 = tk.StringVar()
```

```
labelTitle = tk.Label(root, text="Simple Calculator").grid(row=0, column=2)
labelNum1 = tk.Label(root, text="Enter a number").grid(row=1, column=0)
labelNum2 = tk.Label(root, text="Enter another number").grid(row=2,
    column=0)
labelResult = tk.Label(root)
labelResult.grid(row=7, column=2)
entryNum1 = tk.Entry(root, textvariable=number1).grid(row=1, column=2)
entryNum2 = tk.Entry(root, textvariable=number2).grid(row=2, column=2)
call_result = partial(call_result, labelResult, number1, number2)
buttonCal = tk.Button(root, text="Calculate",
    command=call_result).grid(row=3, column=0)
root.mainloop()
```

Simple Calculator

Enter a number

Enter another number

Calculate

Simple Calculator

Enter a number

Enter another number

Calculate

Result is 66

❖ **Brief Tour of Other GUIs:**

- Python offers multiple options for developing GUI (Graphical User Interface). The most commonly used GUI methods are

1. Tix (Tk Interface eXtensions):

- Tix, which stands for Tk Interface Extension, is an extension library for Tcl/Tk. Tix adds many new widgets, image types and other commands that allows you to create compelling Tcl/Tk-based GUI applications.
- Tix includes the standard, widgets those are `tixGrid`, `tixHList`, `tixInputOnly`, `tixTlist` and etc.

2. Pmw (Python MegaWidgets Tkinter extension):

- Pmw is a toolkit for building high-level compound widgets in Python using the Tkinter module.
- It consists of a set of base classes and a library of flexible and extensible megawidgets built on this foundation. These megawidgets include notebooks, comboboxes, selection widgets, paned widgets, scrolled widgets and dialog windows.

3. wxPython (Python binding to wxWidgets):

- wxPython is a blending of the wxWidgets GUI classes and the Python programming language.
- wxPython is a Python package that can be imported at runtime that includes a collection of Python modules and an extension module (native code). It provides a series of Python classes that mirror (or shadow) many of the wxWidgets GUI classes.