Python 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..,

Example

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

– tkinter

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

- Fastest and easiest way to create the GUI applications.

- Creating a GUI using tkinter is an easy task.

To create a tkinter:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. 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 is '**tkinter'**.

```
import tkinter
```

There are two main methods used you the user need to remember while creating the Python application with GUI.

1. To create a main window, tkinter offers a method

   tk(screenName=None, baseName=None, className='Tk', useTk=1)

   To change the name of the window, you can change the className to the desired one.

The basic code used to create the main window of the application is:

```
window=tkinter.Tk()
where m is the name of the main window object
```

2. **mainloop():**

- There is a method known by the name mainloop() is used when you are ready for the application to run.
- mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

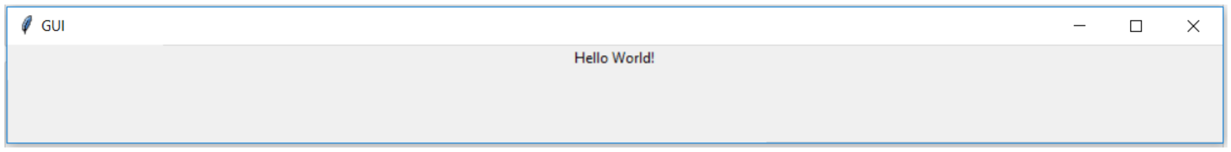
```
m.mainloop()
```

Example

```
import tkinter
window = tkinter.Tk()
# to rename the title of the window
window.title("GUI")
# pack is used to show the object in the window
label = tkinter.Label(window, text = "Hello World!").pack()
```

```
window.mainloop()
```

You will see a similar window like this.



Widgets

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

- **Widgets** are something like elements in the **HTML**

- You will find different types of **widgets** to the different types of elements in the **tkinter**.

Example: Button, Canvas, Checkbutton etc

Geometry Management

- Widgets in the **tkinter** will have some geometry measurements.

- These measurements give you to organize the widgets and their parent frames, windows, etc..,

There are mainly three geometry manager classes class.

1. **pack() method:**It organizes the widgets in blocks before placing in the parent widget.

2. **grid() method:**It organizes the widgets in grid (table-like structure) before placing in the parent widget.

3. **place() method:**It organizes the widgets by placing them on specific positions directed by the programmer

Organizing Layout And Widgets

To arrange the layout in the **window**, we will use **Frame**, class.

Example: Let's create a simple program to see how the **Frame** works.

Steps:-

- **Frame** is used to create the divisions in the window. You can align the frames as you like with **side** parameter of **pack()** method.

- **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..,

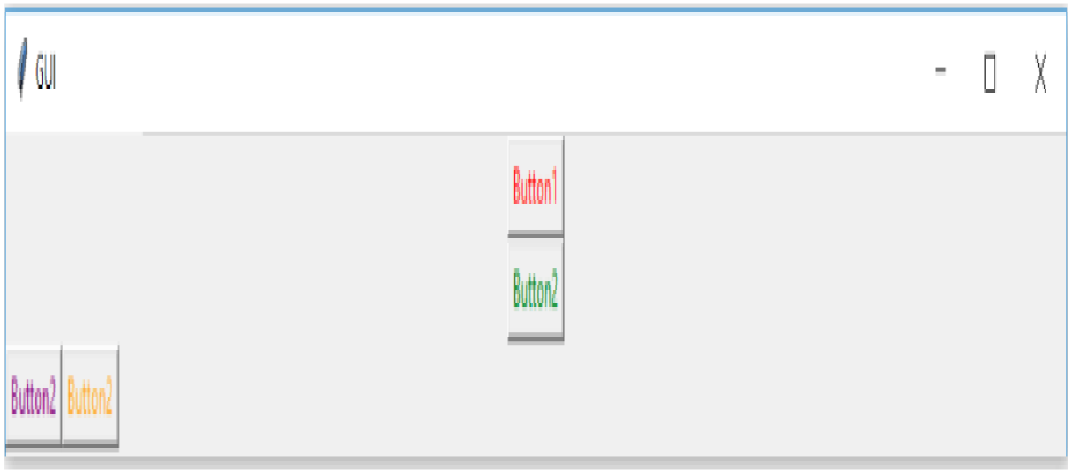
Note:- The parameter of any **widget** method must be where to place the widget.

In the below code, we use to place in the **window**, **top_frame**, **bottom_frame**

```
import tkinter
```

```
window = tkinter.Tk()
window.title("GUI")

# 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
```

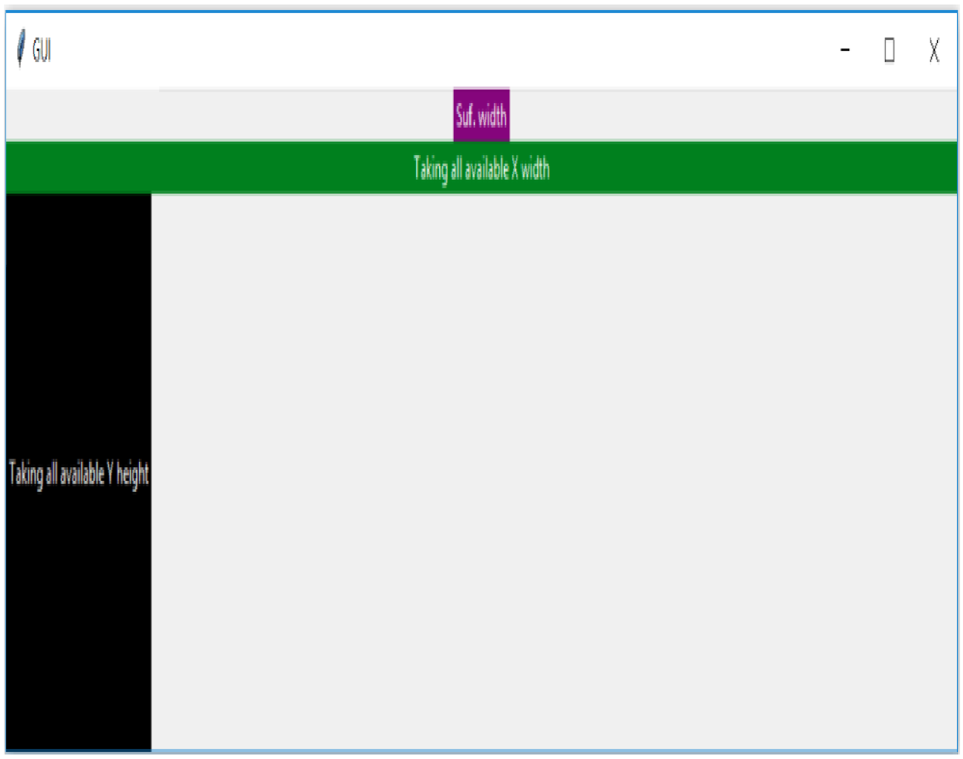
Above code produces the following **window**, if you didn't change the above code.



Using pack()

Example: Now, we will see how to use the **fill** parameter of **pack()**

```
import tkinter

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

# creating 3 simple Labels containing any text

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

# 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()
```



Using grid()

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

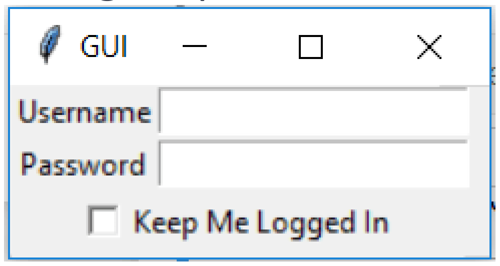
```
0 0   0 1

1 0   1 1
```

Example

```
import tkinter

window = tkinter.Tk()
```

```
window.title("GUI")

# 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

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()
```

You will get the following output.



Using place()

You can access the place manager through the **place** method which is available for all standard widget

When to use the Place Manager

- It is usually not a good idea to use **place** for ordinary window and dialog layouts; its simply to much work to get things working as they should.
- Use the **pack** or **grid** managers for such purposes.
- However, **place** has its uses in more specialized cases.
- Most importantly, it can be used by compound widget containers to implement various custom geometry managers.
- Another use is to position control buttons in dialogs.

In following example it packs a **Label** widget in a frame widget, and then places a **Button** in the upper right corner of the frame. The button will overlap the label.

```
pane = Frame(master)
Label(pane, text="Pane Title").pack()
b = Button(pane, width=12, height=12,
        image=launch_icon, command=self.launch)
b.place(relx=1, x=-2, y=2, anchor=NE)
```

You can combine the absolute and relative options. In such cases, the relative option is applied first, and the absolute value is then added to that position. In the following example, the widget *w* is almost completely covers its parent, except for a 5 pixel border around the widget.

```
 w.place(x=5, y=5, relwidth=1, relheight=1, width=-10, height=-
10)
```

You can also place a widget outside another widget.

For example, why not place two widgets on top of each other:

```
    w2.place(in_=w1, relx=0.5, y=-2, anchor=S,
 bordermode="outside")
```

Note the use of **relx** and **anchor** options to center the widgets vertically. You could also use (relx=0, anchor=SW) to get left alignment, or (relx=1, anchor=SE) to get right alignment.

https://effbot.org/tkinterbook/place.htm

Widgets

1. Button

To add a button in your application, this widget is used.

The general syntax is:

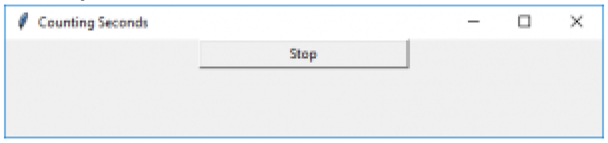
```
w=Button(master, option=value)
```

master is the parameter used to represent the parent window.
There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: to set the background color when button is under the cursor.
- **activeforeground**: to set the foreground color when button is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=25,
command=r.destroy)
button.pack()
r.mainloop()
```

Output:



2. Canvas

It is used to draw pictures and other complex layout like graphics, text and widgets.
The general syntax is:

```
w = Canvas(master, option=value)
master is the parameter used to represent the parent window.
```

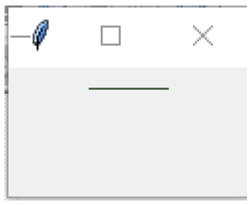
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used in the canvas.
- **highlightcolor**: to set the color shown in the focus highlight.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
master = Tk()
w = Canvas(master, width=40, height=60)
w.pack()
canvas_height=20
canvas_width=200
```

```
y = int(canvas_height / 2)
w.create_line(0, y, canvas_width, y )
mainloop()
```

Outputt:



3. Checkbutton

To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is:

```
w = CheckButton(master, option=value)
```

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **Title**: To set the title of the widget.
- **activebackground**: to set the background color when widget is under the cursor.
- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal backgrouSteganography
  Break

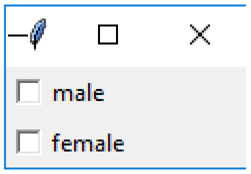
  Secret Code:

  Attach a File:nd color.
- **command**: to call a function.

- **font**: to set the font on the button label.
- **image**: to set the image on the widget.

```
from tkinter import *
master = Tk()
var1 = IntVar()
Checkbutton(master, text='male', variable=var1).grid(row=0,
sticky=W)
var2 = IntVar()
Checkbutton(master, text='female', variable=var2).grid(row=1,
sticky=W)
mainloop()
```

Output:



4. Entry

It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used. The general syntax is:

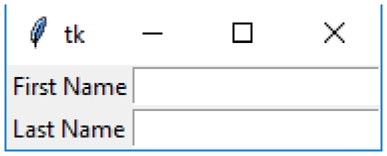
```
w=Entry(master, option=value)
```

master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used.
- **command**: to call a function.
- **highlightcolor**: to set the color shown in the focus highlight.
- **width**: to set the width of the button.
- **height**: to set the height of the button.

```
from tkinter import *
master = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```

Output:



5. Frame

It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

```
w = Frame(master, option=value)
```

master is the parameter used to represent the parent window.

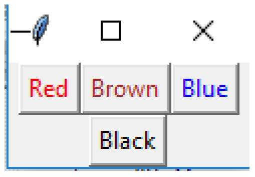
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **bd**: to set the border width in pixels.
- **bg**: to set the normal background color.
- **cursor**: to set the cursor used.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text = 'Red', fg ='red')
redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue', fg ='blue')
```

```
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
root.mainloop()
```

Output:



6. Label

It refers to the display box where you can put any text or image which can be updated any time as per the code.
The general syntax is:

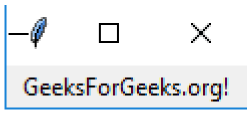
```
w=Label(master, option=value)

master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bg** to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the button.
- **width**: to set the width of the button.
- **height**” to set the height of the button.

```
from tkinter import *
root = Tk()
w = Label(root, text='GeeksForGeeks.org!')
w.pack()
root.mainloop()
```

Output:


GeeksForGeeks.org!

7. Listbox

It offers a list to the user from which the user can accept any number of options.
The general syntax is:

```
w = Listbox(master, option=value)
```

master is the parameter used to represent the parent window.

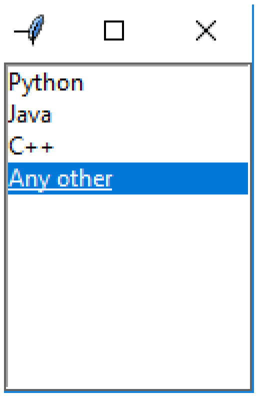
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **bg**: to set he normal background color.
- **bd**: to set the border width in pixels.

- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *

top = Tk()
Lb = Listbox(top)
Lb.insert(1, 'Python')
Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4, 'Any other')
Lb.pack()
top.mainloop()
```

Output:



```
Python
Java
C++
Any other
```

8. Menubutton

It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality. The general syntax is:

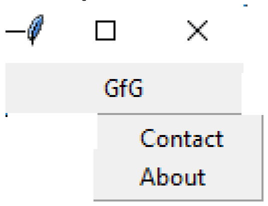
```
w = MenuButton(master, option=value)

master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: To set the background when mouse is over the widget.
- **activeforeground**: To set the foreground when mouse is over the widget.
- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.
- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.

```
from tkinter import *

top = Tk()
mb =  Menubutton ( top, text = &quot;GfG&quot;)
mb.grid()
mb.menu  =  Menu ( mb, tearoff = 0 )
mb[&quot;menu&quot;]  =  mb.menu
cVar  = IntVar()
aVar = IntVar()
mb.menu.add_checkbutton ( label ='Contact', variable = cVar )
mb.menu.add_checkbutton ( label = 'About', variable = aVar )
mb.pack()
top.mainloop()
```

Output:



9. Menu

It is used to create all kinds of menus used by the application.
The general syntax is:

```
w = Menu(master, option=value)
```

master is the parameter used to represent the parent window.

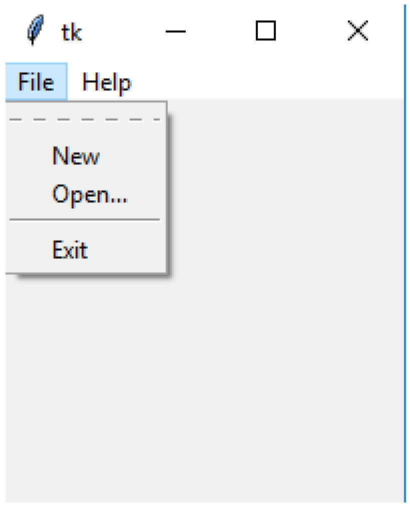
There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **title**: To set the title of the widget.
- **activebackground**: to set the background color when widget is under the cursor.

- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.

```
from tkinter import *

root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About')
mainloop()
```

Output:



10. Message

It refers to the multi-line and non-editable text. It works same as that of Label.
The general syntax is:

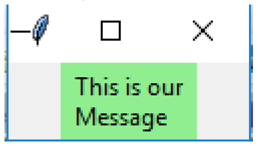
```
w = Message(master, option=value)

master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bd**: to set the border around the indicator.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
main = Tk()
ourMessage ='This is our Message'
messageVar = Message(main, text = ourMessage)
messageVar.config(bg='lightgreen')
messageVar.pack( )
main.mainloop( )
```

Output:



This is our Message

11. Radiobutton

It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.
The general syntax is:

```
w = RadioButton(master, option=value)
```

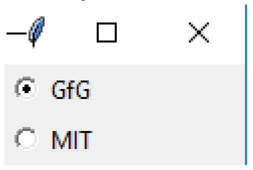
There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **activebackground**: to set the background color when widget is under the cursor.

- **activeforeground**: to set the foreground color when widget is under the cursor.
- **bg**: to set he normal background color.
- **command**: to call a function.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the label in characters.
- **height**: to set the height of the label in characters.

```
from tkinter import *
root = Tk()
v = IntVar()
Radiobutton(root, text='GfG', variable=v,
value=1).pack(anchor=W)
Radiobutton(root, text='MIT', variable=v,
value=2).pack(anchor=W)
mainloop()
```

Output:



12. Scale

It is used to provide a graphical slider that allows to select any value from that scale. The general syntax is:

```
w = Scale(master, option=value)
```

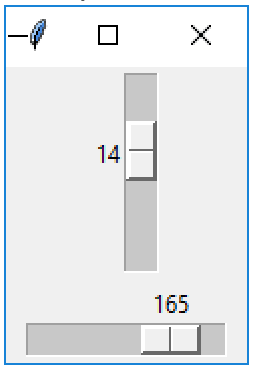
master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **cursor**: To change the cursor pattern when the mouse is over the widget.
- **activebackground**: To set the background of the widget when mouse is over the widget.
- **bg**: to set he normal background color.
- **orient**: Set it to HORIZONTAL or VERTICAL according to the requirement.
- **from_**: To set the value of one end of the scale range.
- **to**: To set the value of the other end of the scale range.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.

```
from tkinter import *
master = Tk()
w = Scale(master, from_=0, to=42)
w.pack()
w = Scale(master, from_=0, to=200, orient=HORIZONTAL)
w.pack()
mainloop()
```

Output:



13. Scrollbar

It refers to the slide controller which will be used to implement listed widgets.
The general syntax is:

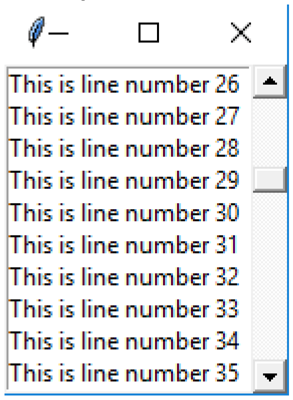
```
w = Scrollbar(master, option=value)

master is the parameter used to represent the parent window.
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **width**: to set the width of the widget.
- **activebackground**: To set the background when mouse is over the widget.
- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.

```
from tkinter import *
root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill = Y )
mylist = Listbox(root, yscrollcommand = scrollbar.set )
for line in range(100):
  mylist.insert(END, 'This is line number' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )
mainloop()
```

Output:



```
This is line number 26
This is line number 27
This is line number 28
This is line number 29
This is line number 30
This is line number 31
This is line number 32
This is line number 33
This is line number 34
This is line number 35
```

14. Text

To edit a multi-line text and format the way it has to be displayed.

The general syntax is:

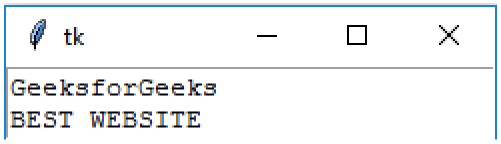
```
w  =Text(master, option=value)
```

There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
- **insertbackground**: To set the background of the widget.
- **bg**: to set he normal background color.
- **font**: to set the font on the button label.
- **image**: to set the image on the widget.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30)
T.pack()
T.insert(END, 'GeeksforGeeks\nBEST WEBSITE\n')
mainloop()
```

Output:



```
GeeksforGeeks
BEST WEBSITE
```

15. Toplevel

This widget is directly controlled by the window manager. It don’t need any parent window to work on.The general syntax is:

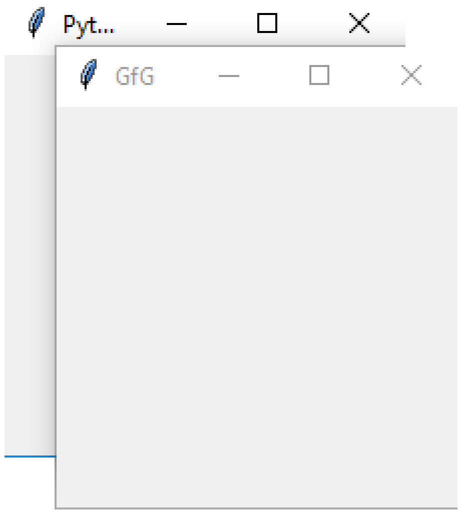
```
w = TopLevel(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
root = Tk()
root.title('GfG')
top = Toplevel()
top.title('Python')
top.mainloop()
```

Output:



16. Spinbox

It is an entry of ‘Entry’ widget. Here, value can be input by selecting a fixed value of numbers.The general syntax is:

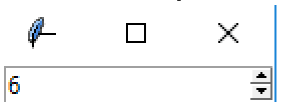
```
w = SpinBox(master, option=value)
```

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **command**: To call a function.
- **width**: to set the width of the widget.
- **activebackground**: To set the background when mouse is over the widget.
- **disabledbackground**: To disable the background when mouse is over the widget.
- **from_**: To set the value of one end of the range.
- **to**: To set the value of the other end of the range.

```
from tkinter import *
master = Tk()
w = Spinbox(master, from_ = 0, to = 10)
w.pack()
mainloop()
```

Output:


```
6
```

17. PanedWindow

It is a container widget which is used to handle number of panes arranged in it. The general syntax is:

```
w = PannedWindow(master, option=value)
```

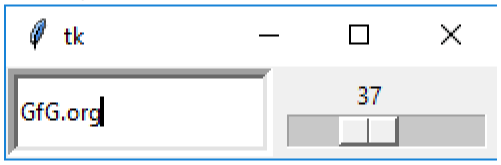
master is the parameter used to represent the parent window. There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

- **bg**: to set he normal background color.
- **bd**: to set the size of border around the indicator.
- **cursor**: To appear the cursor when the mouse over the menubutton.
- **width**: to set the width of the widget.
- **height**: to set the height of the widget.

```
from tkinter import *
m1 = PanedWindow()
m1.pack(fill = BOTH, expand = 1)
left = Entry(m1, bd = 5)
m1.add(left)
m2 = PanedWindow(m1, orient = VERTICAL)
m1.add(m2)
```

```
top = Scale( m2, orient = HORIZONTAL)
m2.add(top)
mainloop()
```

Output:



18. LabelFrame

A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

This widget has the features of a frame plus the ability to display a label.

Syntax

w = LabelFrame(master, option, ...)

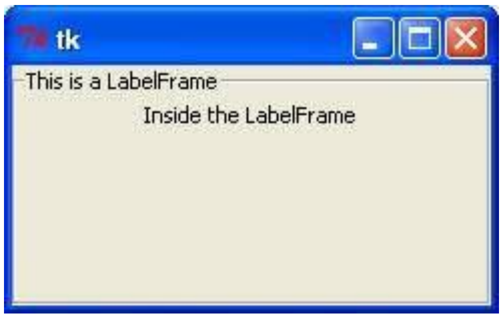
master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

- **Bg** The normal background color displayed behind the label and indicator.
- **Bd** The size of the border around the indicator. Default is 2 pixels.

- **Cursor** If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton.
- **Font** The vertical dimension of the new frame.
- **Height** The vertical dimension of the new frame.
- **labelAnchor** Specifies where to place the label.
- **Highlightbackground** Color of the focus highlight when the frame does not have focus
- **Highlightcolor** Color shown in the focus highlight when the frame has the focus.
- **Highlightthickness** Thickness of the focus highlight.
- **Relief** With the default value, relief=FLAT, the checkbutton does not stand out from its background. You may set this option to any of the other styles
- **Text** Specifies a string to be displayed inside the widget.
- **Width** Specifies the desired width for the window.

```
from Tkinter import *

root = Tk()

labelframe = LabelFrame(root, text="This is a LabelFrame")
labelframe.pack(fill="both", expand="yes")

left = Label(labelframe, text="Inside the LabelFrame")
left.pack()

root.mainloop()
```



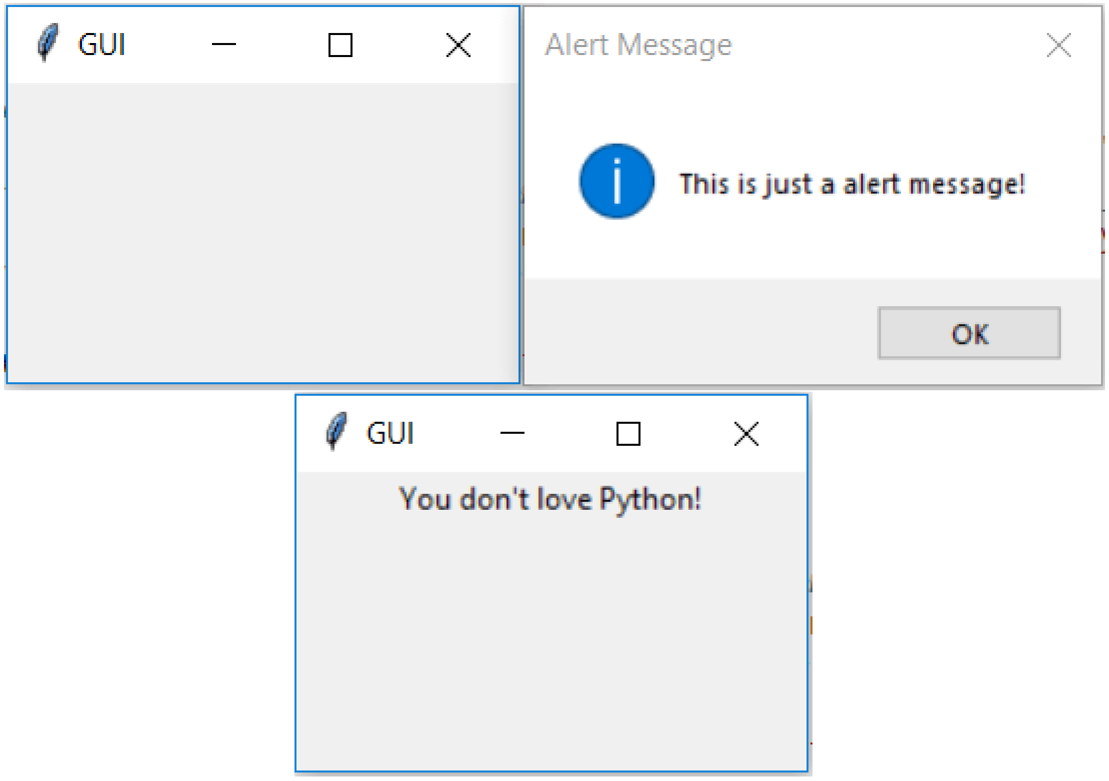
19. Tk MessageBox

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

```
import tkinter
import tkinter.messagebox

window = tkinter.Tk()
window.title("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()
```

You will see the following output.



This is just a alert message!

You don't love Python!

Simple Shapes

You are going to draw some basic shapes with the **Canvas** provided by **tkinter** in **GUI**.

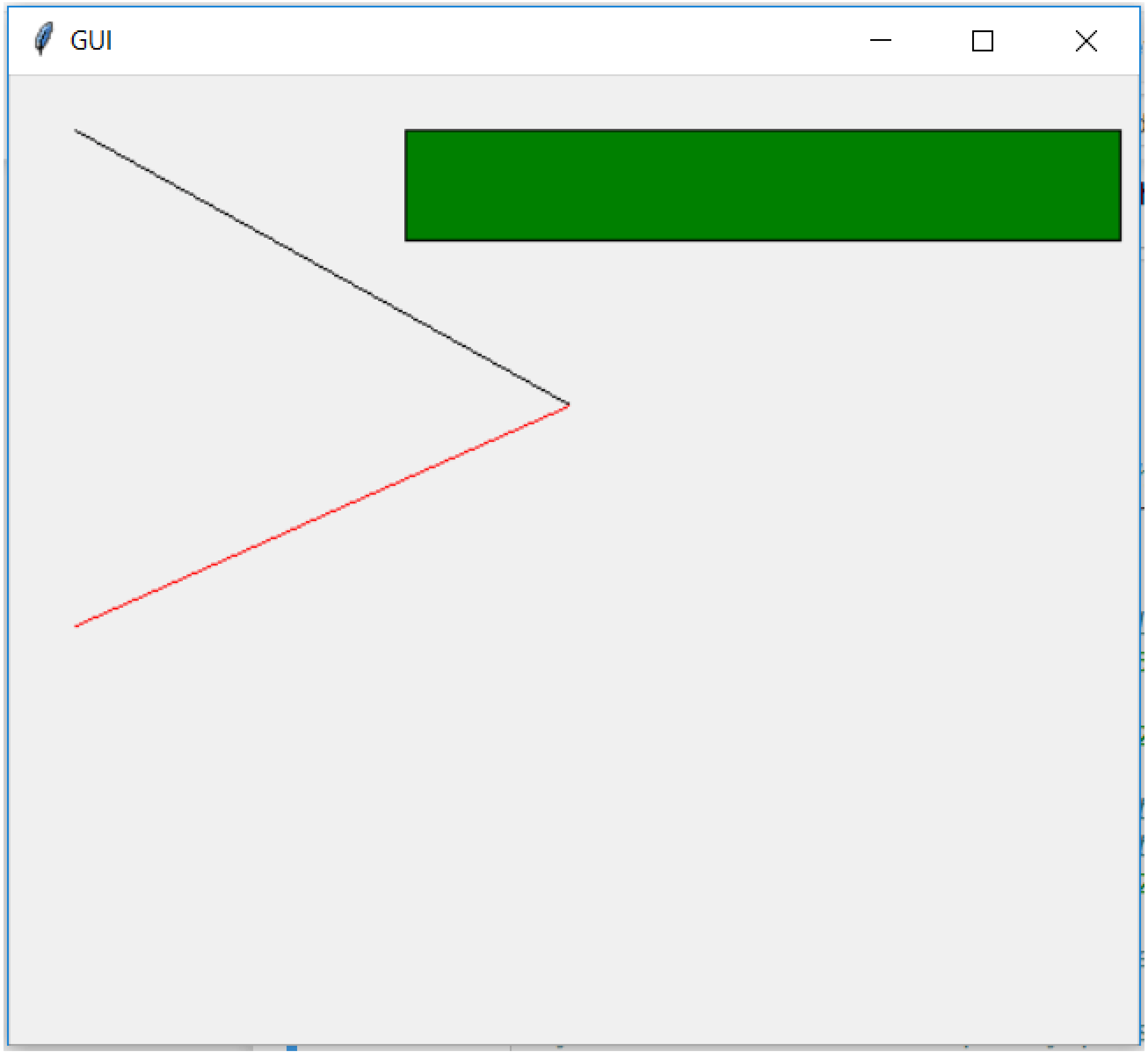
See the example

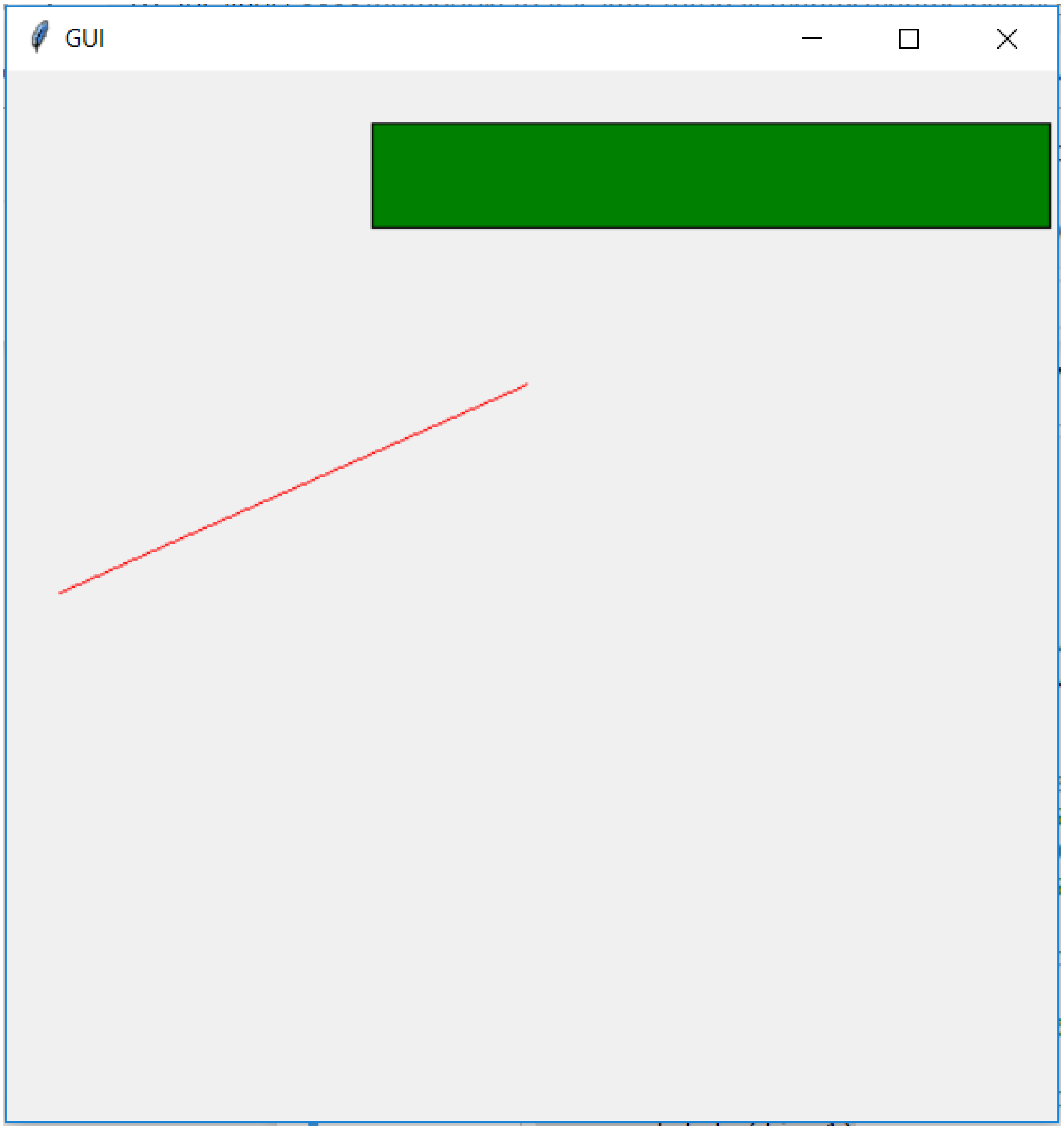
```
import tkinter
```

```python
window = tkinter.Tk()
window.title("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")

# 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()
```

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.





Images and Icons

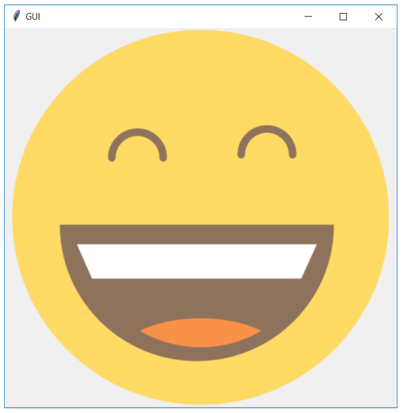
You can add **Images** and **Icons** using **PhotoImage** method.

Let's how it works.

```
import tkinter
```

```
window = tkinter.Tk()
window.title("GUI")

# taking image from the directory and storing the source in a variable
icon = tkinter.PhotoImage(file = "images/haha.png")
# displaying the picture using a 'Label' by passing the 'picture' variriable to 'image' parameter
label = tkinter.Label(window, image = icon)
label.pack()

window.mainloop()
```

You can see the icon in the **GUI**.



Project

Creating Calculator

Every **GUI** apps include two steps.

- Creating User Interface
- Adding functionalities to the **GUI**

Let's start creating **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")


###################################### 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("")
```

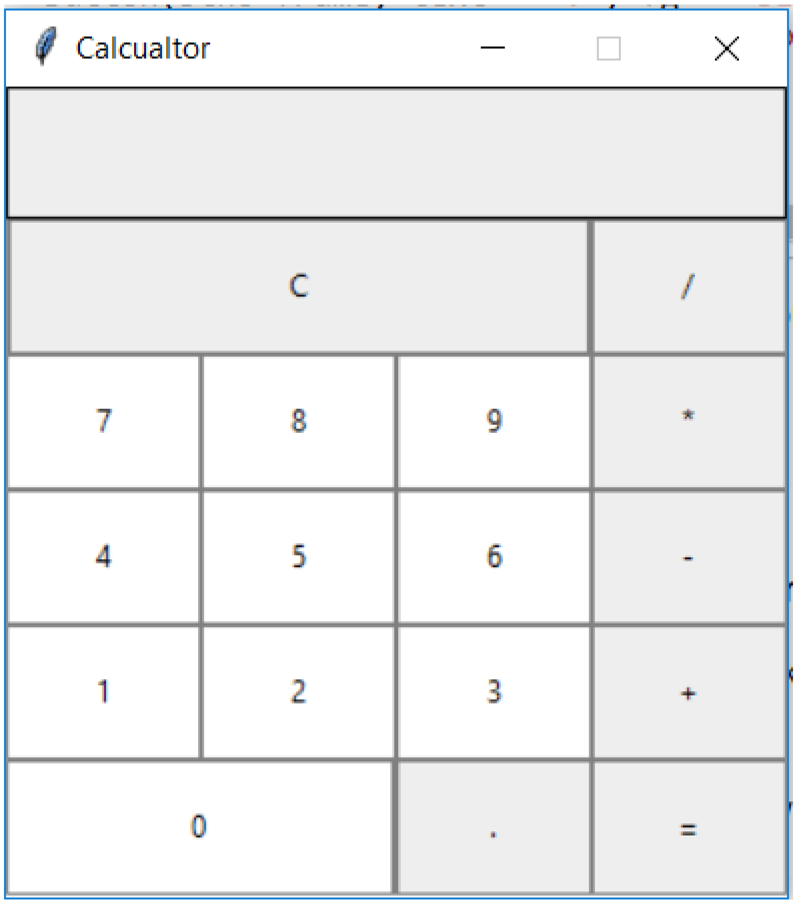
```
# 'btn_equal' calculates the expression present in input field
def btn_equal():
    global expression
    result = str(eval(expression)) # 'eval' function evalutes the
string expression directly
    # you can also implement your own function to evalute the
expression istead of 'eval' function
    input_text.set(result)
    expression = ""

expression = ""
# 'StringVar()' is used to get the instance of input field
input_text = StringVar()


# 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)
```

```
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)


# fourth row
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()
```


Calcualtor
C
/
7
8
9
*
4
5
6
-
1
2
3
+
0
.
=