Python Programming

MODULE – V

Agenda:

- GUI Programming: Introduction
- Tkinter and Python Programming
- Brief Tour of Other GUIs
- Related Modules and Other GUIs.

Graphical User Interface (GUI)

Graphical User Interface (GUI) is nothing but a desktop application which helps you to interact with the computers. They are used to perform different tasks in the desktops, laptops and other electronic devices.

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

They are some different types of **GUI** apps which we daily use on the laptops or desktops.

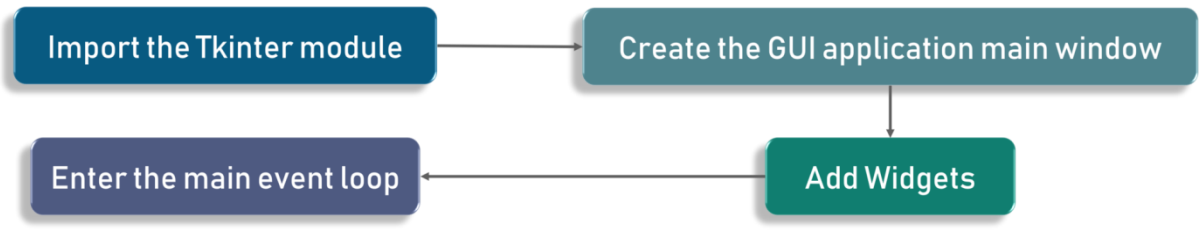
Python Libraries To Create Graphical User Interfaces:
Python has a excess of libraries and these 4 stands out mainly when it comes to GUI. There are as follows:

- Kivy
- Python QT
- wxPython
- Tkinter
- Jpython

Among all of these, Tkinter is preferred by learners and developers just because of how simple and easy it is.

Tkinter

- The primary GUI toolkit we will be using is Tk (Toolkit), Python's default GUI, We'll access Tk from its Python interface called Tkinter (i.e. **Tk interface**). It is originally designed for the **Tool Command Language** (**TCL**). Due to Tk's popularity, it has been ported to a variety of other scripting languages, including Perl (Perl/Tk), Ruby (Ruby/Tk), and Python (Tkinter).
- **Tkinter** is actually an inbuilt **Python** module used to create simple **GUI** apps. It is the most commonly used module for **GUI** apps in the **Python**.
- We no need to worry about installation of the **Tkinter** module as it comes with **Python 3.6 version** by default.
- Consider the following diagram, it shows how an application actually executes in Tkinter:



Developing desktop based applications with python Tkinter is not a complex task. An empty Tkinter top-level window 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.

If you noticed, there are 2 keywords here that you might not know at this point. These are the 2 keywords:

- Widgets
- Main Event Loop

An event loop is basically telling the code to keep displaying the window until we manually close it. It runs in an infinite loop in the back-end.

Example

```
from tkinter import *

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

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

Example

```
import tkinter
window = tkinter.Tk()

# to rename the title of the window window.title("MREC")
 # pack is used to show the object in the window

label = tkinter.Label(window, text = "Hello MREC CSE!").pack()
window.title("GUI Window")
 window.mainloop()
```

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**.

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python

		application.
2	Canvas	The canvas widget is used to draw the shapes in your GUI.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values (Input Fields)
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.
7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.
9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.
11	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	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	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	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container
18	MessageBox	This module is used to display the message-box in the desktop based applications.

These widgets are the reason that Tkinter is so popular. It makes it really easy to understand and use practically.

Python Tkinter Geometry Managers

The Tkinter geometry specifies the method by using which; the widgets are represented on display. The python Tkinter provides three geometry methods that help with positioning our widgets.

1. The pack() method
2. The grid() method
3. The place() method

Python Tkinterpack() method

The pack() widget is used to organize widget in the block, which mean it occupies the entire available width. It's a standard method to show the widgets in the window.

The syntax to use the pack() is given below.

Syntax:

widget.pack(options)

A list of possible options that can be passed in pack() is given below.

- **side:** it represents the side of the parent to which the widget is to be placed on the window.

Example

```
from tkinter import *
parent = Tk()
redbutton = Button(parent, text = "Red", fg = "red")
redbutton.pack( side = LEFT)
greenbutton = Button(parent, text = "Black", fg = "black")
greenbutton.pack( side = RIGHT )
bluebutton = Button(parent, text = "Blue", fg = "blue")
bluebutton.pack( side = TOP )
blackbutton = Button(parent, text = "Green", fg = "red")
blackbutton.pack( side = BOTTOM)
parent.mainloop()
```

Output:

Python Tkintergrid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

This is a more organized way to place the widgets to the python application. The syntax to use the grid() is given below.

Syntax

widget.grid(options)

A list of possible options that can be passed inside the grid() method is given below.

- **Column**
  The column number in which the widget is to be placed. The leftmost column is represented by 0.
- **row**
  The row number in which the widget is to be placed. The topmost row is represented by 0.

Example

```
from tkinter import *
parent = Tk()
name = Label(parent,text = "Name").grid(row = 0, column = 0)
e1 = Entry(parent).grid(row = 0, column = 1)
password = Label(parent,text = "Password").grid(row = 1, column = 0)
e2 = Entry(parent).grid(row = 1, column = 1)
submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
parent.mainloop()
```

Output:

Python Tkinterplace() method

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

Syntax

widget.place(options)

A list of possible options is given below.

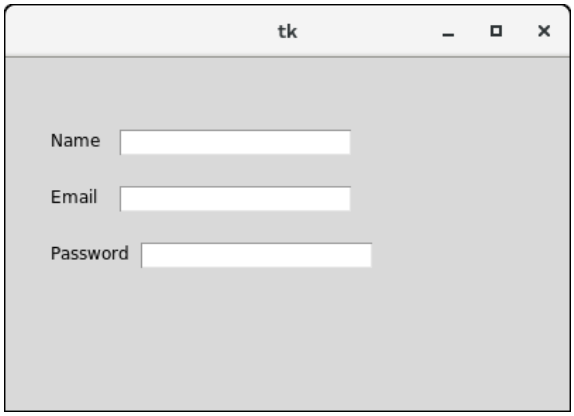
- **x, y:** It refers to the horizontal and vertical offset in the pixels.

Example

```
from tkinter import *
top = Tk()
top.geometry("400x250")
```

```
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



Python Tkinter Button

The button widget is used to add various types of buttons to the python application. Python allows us to configure the look of the button according to our requirements. Various options can be set or reset depending upon the requirements.

We can also associate a method or function with a button which is called when the button is pressed.

The syntax to use the button widget is given below.

Syntax

W = Button(parent, options)

A list of possible options is given below.

SN	Option	Description
1	activebackground	It represents the background of the button when the mouse hover the button.
2	activeforeground	It represents the font color of the button when the mouse hover the button.
3	Bd	It represents the border width in pixels.
4	Bg	It represents the background color of the button.
5	Command	It is set to the function call which is scheduled when the function is called.

6	Fg	Foreground color of the button.
7	Font	The font of the button text.
8	Height	The height of the button. The height is represented in the number of text lines for the textual lines or the number of pixels for the images.
10	Highlightcolor	The color of the highlight when the button has the focus.
11	Image	It is set to the image displayed on the button.
12	Justify	It illustrates the way by which the multiple text lines are represented. It is set to LEFT for left justification, RIGHT for the right justification, and CENTER for the center.
13	Padx	Additional padding to the button in the horizontal direction.
14	Pady	Additional padding to the button in the vertical direction.
15	Relief	It represents the type of the border. It can be SUNKEN, RAISED, GROOVE, and RIDGE.
17	State	This option is set to DISABLED to make the button unresponsive. The ACTIVE represents the active state of the button.
18	Underline	Set this option to make the button text underlined.
19	Width	The width of the button. It exists as a number of letters for textual buttons or pixels for image buttons.
20	Wraplength	If the value is set to a positive number, the text lines will be wrapped to fit within this length.

Example

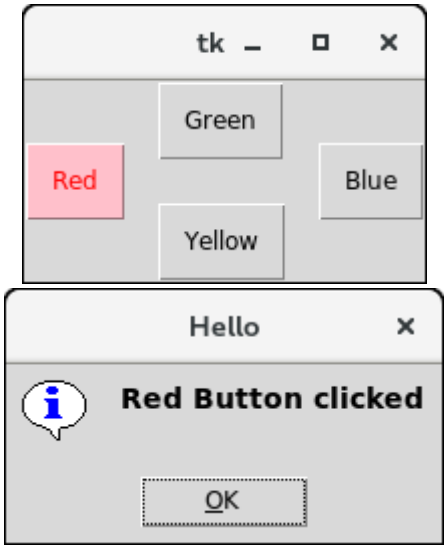
```
#python application to create a simple button

from tkinter import *
top = Tk()
top.geometry("200x100")

b = Button(top,text = "Simple")
b.pack()
top.mainloop()
```

Output:

Example

```
from tkinter import *
from tkinter import messagebox

top = Tk()
top.geometry("200x100")

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

b1 = Button(top,text = "Red",command = fun,activeforeground = "red",activebackg
round = "pink",pady=10)
b2 = Button(top, text = "Blue",activeforeground = "blue",activebackground = "pink
",pady=10)
b3 = Button(top, text = "Green",activeforeground = "green",activebackground = "pi
nk",pady = 10)
b4 = Button(top, text = "Yellow",activeforeground = "yellow",activebackground = "
pink",pady = 10) b1.pack(side = LEFT)
b2.pack(side = RIGHT)
b3.pack(side = TOP)
b4.pack(side = BOTTOM)
top.mainloop()
```

Output:



Python Tkinter Canvas

- The canvas widget is used to add the structured graphics to the python application. It is used to draw the graph and plots to the python application. The syntax to use the canvas is given below.

Syntax

w = canvas(parent, options)

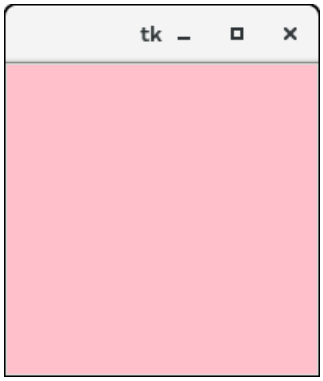
A list of possible options is given below.

SN	Option	Description
1	Bd	The represents the border width. The default width is 2.
2	Bg	It represents the background color of the canvas.
3	confine	It is set to make the canvas unscrollable outside the scroll region.
4	cursor	The cursor is used as the arrow, circle, dot, etc. on the canvas.
5	height	It represents the size of the canvas in the vertical direction.
6	highlightcolor	It represents the highlight color when the widget is focused.
7	relief	It represents the type of the border. The possible values are SUNKEN, RAISED, GROOVE, and RIDGE.
8	scrollregion	It represents the coordinates specified as the tuple containing the area of the canvas.
9	width	It represents the width of the canvas.
10	xscrollincrement	If it is set to a positive value. The canvas is placed only to the multiple of this value.
11	xscrollcommand	If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar.
12	yscrollincrement	Works like xscrollincrement, but governs vertical movement.
13	yscrollcommand	If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar.

Example

```
from tkinter import *
top = Tk()
top.geometry("200x200")
#creating a simple canvas
c = Canvas(top,bg = "pink",height = "200")
c.pack()
top.mainloop()
```

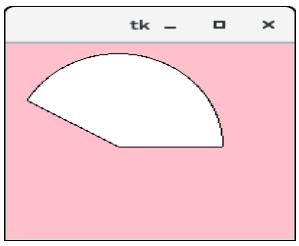
Output:



Example: Creating an arc

```
from tkinter import *

top = Tk()
top.geometry("200x200")

#creating a simple canvas
c = Canvas(top,bg = "pink",height = "200",width = 200)
arc = c.create_arc((5,10,150,200),start = 0,extent = 150, fill= "white")
c.pack()
top.mainloop()
```

Output:



Python TkinterCheckbutton

The Checkbutton is used to track the user's choices provided to the application. In other words, we can say that Checkbutton is used to implement the on/off selections.

The Checkbutton can contain the text or images. 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.

The syntax to use the checkbutton is given below.

Syntax

w = checkbutton(master, options)

A list of possible options is given below.

SN	Option	Description
1	activebackground	It represents the background color when the checkbutton is under the cursor.
2	activeforeground	It represents the foreground color of the checkbutton when the checkbutton is under the cursor.
3	Bg	The background color of the button.
4	Bitmap	It displays an image (monochrome) on the button.
5	Bd	The size of the border around the corner.
6	Command	It is associated with a function to be called when the state of the

		checkbutton is changed.
7	Cursor	The mouse pointer will be changed to the cursor name when it is over the checkbutton.
8	disableforeground	It is the color which is used to represent the text of a disabled checkbutton.
9	font	It represents the font of the checkbutton.
10	fg	The foreground color (text color) of the checkbutton.
11	height	It represents the height of the checkbutton (number of lines). The default height is 1.
12	highlightcolor	The color of the focus highlight when the checkbutton is under focus.
13	image	The image used to represent the checkbutton.
14	justify	This specifies the justification of the text if the text contains multiple lines.
15	offvalue	The associated control variable is set to 0 by default if the button is unchecked. We can change the state of an unchecked variable to some other one.
16	onvalue	The associated control variable is set to 1 by default if the button is checked. We can change the state of the checked variable to some other one.
17	padx	The horizontal padding of the checkbutton
18	pady	The vertical padding of the checkbutton.
19	relief	The type of the border of the checkbutton. By default, it is set to FLAT.
20	selectcolor	The color of the checkbutton when it is set. By default, it is red.
21	selectimage	The image is shown on the checkbutton when it is set.
22	state	It represents the state of the checkbutton. By default, it is set to normal. We can change it to DISABLED to make the checkbutton unresponsive. The state of the checkbutton is ACTIVE when it is under focus.
24	underline	It represents the index of the character in the text which is to be underlined. The indexing starts with zero in the text.
25	variable	It represents the associated variable that tracks the state of the checkbutton.
26	width	It represents the width of the checkbutton. It is represented in the number of characters that are represented in the form of texts.

27	wraplength	If this option is set to an integer number, the text will be broken into the number of pieces.

Methods

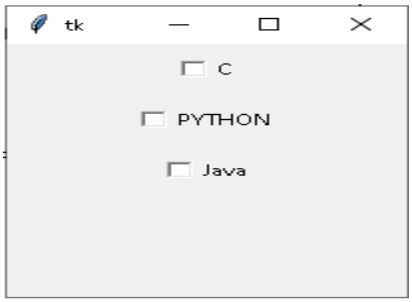
The methods that can be called with the Checkbuttons are described in the following table.

SN	Method	Description
1	deselect()	It is called to turn off the checkbutton.
2	flash()	The checkbutton is flashed between the active and normal colors.
3	invoke()	This will invoke the method associated with the checkbutton.
4	select()	It is called to turn on the checkbutton.
5	toggle()	It is used to toggle between the different Checkbuttons.

Example

```
from tkinter import *

top = Tk()

top.geometry("200x200")

chkbtn1 = Checkbutton(top, text = "C", )

chkbtn2 = Checkbutton(top, text = "PYTHON", )

chkbtn3 = Checkbutton(top, text = "Java", )

chkbtn1.pack()

chkbtn2.pack()

chkbtn3.pack()

top.mainloop()
```

Output:



Python Tkinter Entry

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. It can only be used for one line of text from the user. For multiple lines of text, we must use the text widget.

The syntax to use the Entry widget is given below.

Syntax

w = Entry (parent, options)

A list of possible options is given below.

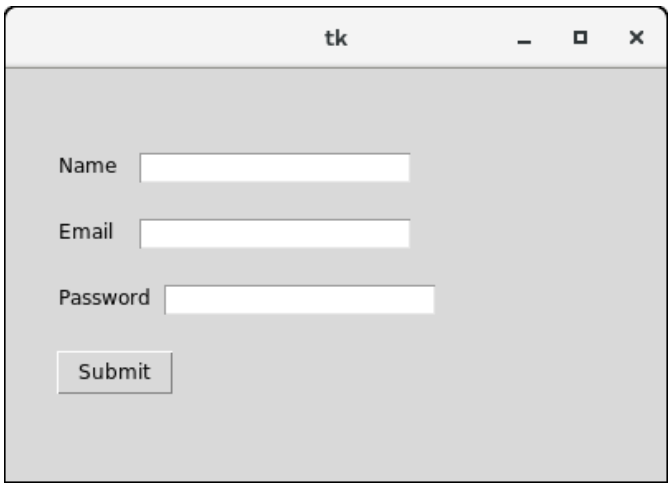
SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	The border width of the widget in pixels.
3	Cursor	The mouse pointer will be changed to the cursor type set to the arrow, dot, etc.
4	Exportselection	The text written inside the entry box will be automatically copied to the clipboard by default. We can set the exportselection to 0 to not copy this.
5	Fg	It represents the color of the text.
6	Font	It represents the font type of the text.
7	Highlightbackground	It represents the color to display in the traversal highlight region when the widget does not have the input focus.
8	Highlightcolor	It represents the color to use for the traversal highlight rectangle that is drawn around the widget when it has the input focus.
9	Highlightthickness	It represents a non-negative value indicating the width of the highlight rectangle to draw around the outside of the widget when it has the input focus.
10	Insertbackground	It represents the color to use as background in the area covered by the insertion cursor. This color will normally override either the normal background for the widget.
11	Insertborderwidth	It represents a non-negative value indicating the width of the 3-D border to draw around the insertion cursor. The value may have any of the forms acceptable to Tk_GetPixels.
12	Insertofftime	It represents a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "off" in each blink cycle. If this option is zero, then the cursor doesn't blink: it is on all the time.

13	Insertontime	Specifies a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "on" in each blink cycle.
14	Insertwidth	It represents the value indicating the total width of the insertion cursor. The value may have any of the forms acceptable to Tk_GetPixels.
15	Justify	It specifies how the text is organized if the text contains multiple lines.
16	Relief	It specifies the type of the border. Its default value is FLAT.
17	Selectbackground	The background color of the selected text.
18	Selectborderwidth	The width of the border to display around the selected task.
19	Selectforeground	The font color of the selected task.
20	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 (*).
21	Textvariable	It is set to the instance of the StringVar to retrieve the text from the entry.
22	Width	The width of the displayed text or image.
23	Xscrollcommand	The entry widget can be linked to the horizontal scrollbar if we want the user to enter more text then the actual width of the widget.

Example

```
from tkinter import *

top = Tk()
top.geometry("400x250")
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
sbmitbtn = Button(top, text = "Submit",activebackground = "pink", activeforegroun
d = "blue").place(x = 30, y = 170)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



Entry widget methods

Python provides various methods to configure the data written inside the widget. There are the following methods provided by the Entry widget.

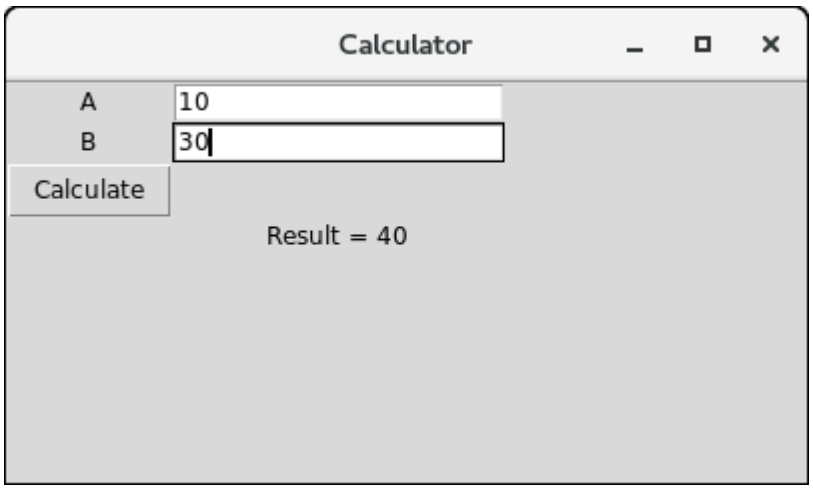
SN	Method	Description
1	delete(first, last = none)	It is used to delete the specified characters inside the widget.
2	get()	It is used to get the text written inside the widget.
3	icursor(index)	It is used to change the insertion cursor position. We can specify the index of the character before which, the cursor to be placed.
4	index(index)	It is used to place the cursor to the left of the character written at the specified index.
5	insert(index,s)	It is used to insert the specified string before the character placed at the specified index.
6	select_adjust(index)	It includes the selection of the character present at the specified index.
7	select_clear()	It clears the selection if some selection has been done.
8	select_form(index)	It sets the anchor index position to the character specified by the index.
9	select_present()	It returns true if some text in the Entry is selected otherwise returns false.
10	select_range(start,end)	It selects the characters to exist between the specified range.
11	select_to(index)	It selects all the characters from the beginning to the specified index.
12	xview(index)	It is used to link the entry widget to a horizontal scrollbar.

13	xview_scroll(number,what)	It is used to make the entry scrollable horizontally.

Example: A simple calculator

```
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 = %d" % result)
    return

root = tk.Tk()
root.geometry('400x200+100+200')

root.title('Calculator')

number1 = tk.StringVar()
number2 = tk.StringVar()

labelNum1 = tk.Label(root, text="A").grid(row=1, column=0)
labelNum2 = tk.Label(root, text="B").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, c
olumn=0)
root.mainloop()
```

Output:



Python Tkinter Frame

Python 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.

The syntax to use the Frame widget is given below.

Syntax

w = Frame(parent, options)

A list of possible options is given below.

SN	Option	Description
1	Bd	It represents the border width.
2	Bg	The background color of the widget.
3	Cursor	The mouse pointer is changed to the cursor type set to different values like an arrow, dot, etc.
4	Height	The height of the frame.
5	highlightbackground	The color of the background color when it is under focus.
6	Highlightcolor	The text color when the widget is under focus.
7	highlightthickness	It specifies the thickness around the border when the widget is under the focus.
8	Relief	It specifies the type of the border. The default value if FLAT.
9	Width	It represents the width of the widget.

Example

```
from tkinter import *

top = Tk()
top.geometry("140x100")
frame = Frame(top)
frame.pack()

leftframe = Frame(top)
leftframe.pack(side = LEFT)

rightframe = Frame(top)
rightframe.pack(side = RIGHT)

btn1 = Button(frame, text="Submit", fg="red",activebackground = "red")
btn1.pack(side = LEFT)

btn2 = Button(frame, text="Remove", fg="brown", activebackground = "brown")
btn2.pack(side = RIGHT)

btn3 = Button(rightframe, text="Add", fg="blue", activebackground = "blue")
btn3.pack(side = LEFT)

btn4 = Button(leftframe, text="Modify", fg="black", activebackground = "white")
btn4.pack(side = RIGHT)

top.mainloop()
```

Output:

Python Tkinter Label

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

There are the various options which can be specified to configure the text or the part of the text shown in the Label.

The syntax to use the Label is given below.

Syntax

w = Label (master, options)

A list of possible options is given below.

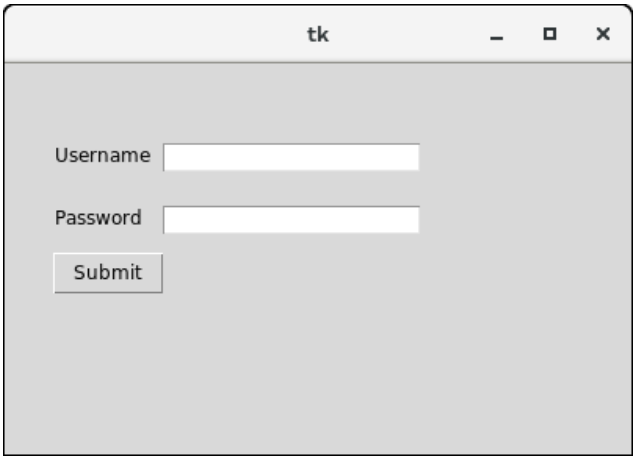
SN	Option	Description
1	Anchor	It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
2	Bg	The background color displayed behind the widget.
3	Bitmap	It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
4	Bd	It represents the width of the border. The default is 2 pixels.
5	Cursor	The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
6	Font	The font type of the text written inside the widget.
7	Fg	The foreground color of the text written inside the widget.
8	Height	The height of the widget.
9	Image	The image that is to be shown as the label.
10	Justify	It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for right justification, and CENTER for center justification.
11	Padx	The horizontal padding of the text. The default value is 1.
12	Pady	The vertical padding of the text. The default value is 1.
13	Relief	The type of the border. The default value is FLAT.
14	Text	This is set to the string variable which may contain one or more line of text.
15	textvariable	The text written inside the widget is set to the control variable StringVar so that it can be accessed and changed accordingly.
16	underline	We can display a line under the specified letter of the text. Set this option to the number of the letter under which the line will be displayed.
17	Width	The width of the widget. It is specified as the number of characters.
18	wraplength	Instead of having only one line as the label text, we can break it to the number of lines where each line has the number of characters specified to

		this option.

Example 1

```
from tkinter import *

top = Tk()

top.geometry("400x250")

#creating label
uname = Label(top, text = "Username").place(x = 30,y = 50)

#creating label
password = Label(top, text = "Password").place(x = 30, y = 90)

sbmitbtn = Button(top, text = "Submit",activebackground = "pink", activeforegroun
d = "blue").place(x = 30, y = 120)

e1 = Entry(top,width = 20).place(x = 100, y = 50)
e2 = Entry(top, width = 20).place(x = 100, y = 90)
top.mainloop()
```

Output:



Python TkinterListbox

The Listbox widget is used to display the list items to the user. We can place only text items in the Listbox and all text items contain the same font and color.

The user can choose one or more items from the list depending upon the configuration.

The syntax to use the Listbox is given below.

w = Listbox(parent, options)

A list of possible options is given below.

SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	It represents the size of the border. Default value is 2 pixel.
3	Cursor	The mouse pointer will look like the cursor type like dot, arrow, etc.
4	Font	The font type of the Listbox items.
5	Fg	The color of the text.
6	Height	It represents the count of the lines shown in the Listbox. The default value is 10.
7	highlightcolor	The color of the Listbox items when the widget is under focus.
8	highlightthickness	The thickness of the highlight.
9	Relief	The type of the border. The default is SUNKEN.
10	selectbackground	The background color that is used to display the selected text.
11	selectmode	It is used to determine the number of items that can be selected from the list. It can set to BROWSE, SINGLE, MULTIPLE, EXTENDED.
12	Width	It represents the width of the widget in characters.
13	xscrollcommand	It is used to let the user scroll the Listbox horizontally.
14	yscrollcommand	It is used to let the user scroll the Listbox vertically.

Methods

There are the following methods associated with the Listbox.

SN	Method	Description
1	activate(index)	It is used to select the lines at the specified index.
2	curselection()	It returns a tuple containing the line numbers of the selected element or elements, counting from 0. If nothing is selected, returns an empty tuple.
3	delete(first, last = None)	It is used to delete the lines which exist in the given range.
4	get(first, last = None)	It is used to get the list items that exist in the given range.
5	index(i)	It is used to place the line with the specified index at the top of the widget.
6	insert(index, *elements)	It is used to insert the new lines with the specified number of elements before the specified index.
7	nearest(y)	It returns the index of the nearest line to the y coordinate of

		the Listbox widget.
8	see(index)	It is used to adjust the position of the listbox to make the lines specified by the index visible.
9	size()	It returns the number of lines that are present in the Listbox widget.
10	xview()	This is used to make the widget horizontally scrollable.
11	xview_moveto(fraction)	It is used to make the listbox horizontally scrollable by the fraction of width of the longest line present in the listbox.
12	xview_scroll(number, what)	It is used to make the listbox horizontally scrollable by the number of characters specified.
13	yview()	It allows the Listbox to be vertically scrollable.
14	yview_moveto(fraction)	It is used to make the listbox vertically scrollable by the fraction of width of the longest line present in the listbox.
15	yview_scroll (number, what)	It is used to make the listbox vertically scrollable by the number of characters specified.

Example 1

```
from tkinter import *

top = Tk()

top.geometry("200x250")

lbl = Label(top,text = "A list of favourite countries...")

listbox = Listbox(top)

listbox.insert(1,"India")
listbox.insert(2, "USA")
listbox.insert(3, "Japan")
listbox.insert(4, "Austrelia")

lbl.pack()
listbox.pack()

top.mainloop()
```

Output:

Example 2: Deleting the active items from the list

```
from tkinter import *

top = Tk()
top.geometry("200x250")

lbl = Label(top,text = "A list of favourite countries...")
listbox = Listbox(top)

listbox.insert(1,"India")
listbox.insert(2, "USA")
listbox.insert(3, "Japan")
listbox.insert(4, "Austrelia")

#this button will delete the selected item from the list

btn = Button(top, text = "delete", command = lambda listbox=listbox: listbox.delete
(ANCHOR))

lbl.pack()
listbox.pack()
btn.pack()
top.mainloop()
```

Output:

After pressing the delete button.

Python Tkinter Menu

The Menu widget is used to create various types of menus (top level, pull down, and pop up) in the python application.

The top-level menus are the one which is displayed just under the title bar of the parent window. We need to create a new instance of the Menu widget and add various commands to it by using the add() method.

The syntax to use the Menu widget is given below.

Syntax

w = Menu(top, options)

A list of possible options is given below.

SN	Option	Description
1	activebackground	The background color of the widget when the widget is under the focus.

2	activeborderwidth	The width of the border of the widget when it is under the mouse. The default is 1 pixel.
3	activeforeground	The font color of the widget when the widget has the focus.
4	Bg	The background color of the widget.
5	Bd	The border width of the widget.
6	cursor	The mouse pointer is changed to the cursor type when it hovers the widget. The cursor type can be set to arrow or dot.
7	disabledforeground	The font color of the widget when it is disabled.
8	Font	The font type of the text of the widget.
9	Fg	The foreground color of the widget.
10	postcommand	The postcommand can be set to any of the function which is called when the mourse hovers the menu.
11	relief	The type of the border of the widget. The default type is RAISED.
12	image	It is used to display an image on the menu.
13	selectcolor	The color used to display the checkbutton or radiobutton when they are selected.
14	tearoff	By default, the choices in the menu start taking place from position 1. If we set the tearoff = 1, then it will start taking place from 0th position.
15	Title	Set this option to the title of the window if you want to change the title of the window.

Methods

The Menu widget contains the following methods.

SN	Option	Description
1	add_command(options)	It is used to add the Menu items to the menu.
2	add_radiobutton(options)	This method adds the radiobutton to the menu.
3	add_checkbutton(options)	This method is used to add the checkbuttons to the menu.
4	add_cascade(options)	It is used to create a hierarchical menu to the parent menu by associating the given menu to the parent menu.
5	add_seperator()	It is used to add the seperator line to the menu.
6	add(type, options)	It is used to add the specific menu item to the menu.
7	delete(startindex, endindex)	It is used to delete the menu items exist in the specified range.
8	entryconfig(index,	It is used to configure a menu item identified by the given

	options)	index.
9	index(item)	It is used to get the index of the specified menu item.
10	insert_seperator(index)	It is used to insert a seperator at the specified index.
11	invoke(index)	It is used to invoke the associated with the choice given at the specified index.
12	type(index)	It is used to get the type of the choice specified by the index.

Creating a top level menu

A top-level menu can be created by instantiating the Menu widget and adding the menu items to the menu.

Example 1

```
from tkinter import *

top = Tk()

def hello():
   print("hello!")

# create a toplevel menu
menubar = Menu(top)
menubar.add_command(label="Hello!", command=hello)
menubar.add_command(label="Quit!", command=top.quit)

# display the menu
top.config(menu=menubar)

top.mainloop()
```

Output:

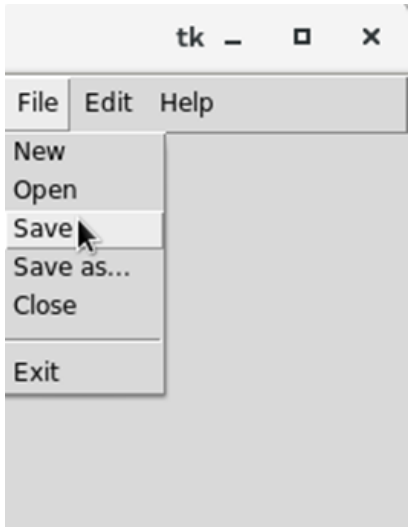
Clicking the hello Menubutton will print the hello on the console while clicking the Quit Menubutton will make an exit from the python application.

Example 2

```
from tkinter import Toplevel, Button, Tk, Menu

top = Tk()
menubar = Menu(top)
file = Menu(menubar, tearoff=0)
file.add_command(label="New")
file.add_command(label="Open")
file.add_command(label="Save")
file.add_command(label="Save as...")
file.add_command(label="Close")

file.add_separator()

file.add_command(label="Exit", command=top.quit)

menubar.add_cascade(label="File", menu=file)
edit = Menu(menubar, tearoff=0)
edit.add_command(label="Undo")

edit.add_separator()

edit.add_command(label="Cut")
edit.add_command(label="Copy")
edit.add_command(label="Paste")
edit.add_command(label="Delete")
edit.add_command(label="Select All")

menubar.add_cascade(label="Edit", menu=edit)
```

```
help = Menu(menubar, tearoff=0)
help.add_command(label="About")
menubar.add_cascade(label="Help", menu=help)

top.config(menu=menubar)
top.mainloop()
```

Output:



Python Tkinter Message

The Message widget is used to show the message to the user regarding the behaviour of the python application. The message widget shows the text messages to the user which can not be edited.

The message text contains more than one line. However, the message can only be shown in the single font.

The syntax to use the Message widget is given below.

Syntax

w = Message(parent, options)

A list of possible options is given below.

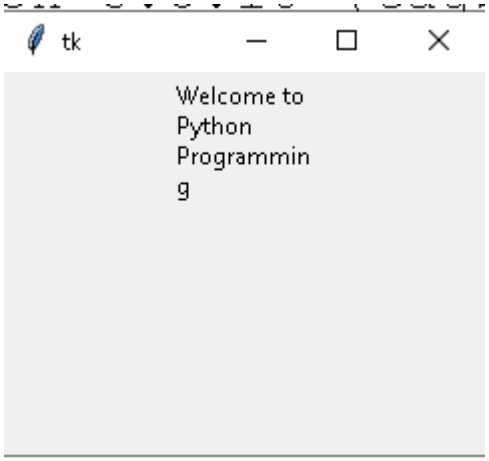
SN	Option	Description
1	Anchor	It is used to decide the exact position of the text within the space provided to the widget if the widget contains more space than the need of the text. The default is CENTER.
2	Bg	The background color of the widget.
3	Bitmap	It is used to display the graphics on the widget. It can be set to any graphical or image object.
4	Bd	It represents the size of the border in the pixel. The default size is 2 pixel.
5	Cursor	The mouse pointer is changed to the specified cursor type. The cursor

		type can be an arrow, dot, etc.
6	Font	The font type of the widget text.
7	Fg	The font color of the widget text.
8	Height	The vertical dimension of the message.
9	Image	We can set this option to a static image to show that onto the widget.
10	Justify	This option is used to specify the alignment of multiple line of code with respect to each other. The possible values can be LEFT (left alignment), CENTER (default), and RIGHT (right alignment).
11	Padx	The horizontal padding of the widget.
12	Pady	The vertical padding of the widget.
13	Relief	It represents the type of the border. The default type is FLAT.
14	Text	We can set this option to the string so that the widget can represent the specified text.
15	Textvariable	This is used to control the text represented by the widget. The textvariable can be set to the text that is shown in the widget.
16	Underline	The default value of this option is -1 that represents no underline. We can set this option to an existing number to specify that nth letter of the string will be underlined.
17	Width	It specifies the horizontal dimension of the widget in the number of characters (not pixel).
18	Wraplength	We can wrap the text to the number of lines by setting this option to the desired number so that each line contains only that number of characters.

Example

```
from tkinter import *

top = Tk()
top.geometry("100x100")

msg = Message( top, text = "Welcome to Python Programming")

msg.pack()
top.mainloop()
```

Output:



Python TkinterRadiobutton

The Radiobutton widget is used to implement one-of-many selection in the python application. It shows multiple choices to the user out of which, the user can select only one out of them. We can associate different methods with each of the radiobutton.

We can display the multiple line text or images on the radiobuttons. To keep track the user's selection the radiobutton, it is associated with a single variable. Each button displays a single value for that particular variable.

The syntax to use the Radiobutton is given below.

Syntax

w = Radiobutton(top, options)

SN	Option	Description
1	activebackground	The background color of the widget when it has the focus.
2	activeforeground	The font color of the widget text when it has the focus.
3	anchor	It represents the exact position of the text within the widget if the widget contains more space than the requirement of the text. The default value is CENTER.
4	Bg	The background color of the widget.
5	bitmap	It is used to display the graphics on the widget. It can be set to any graphical or image object.
6	borderwidth	It represents the size of the border.
7	command	This option is set to the procedure which must be called every-time when the state of the radiobutton is changed.
8	cursor	The mouse pointer is changed to the specified cursor type. It can be set to the arrow, dot, etc.
9	Font	It represents the font type of the widget text.
10	Fg	The normal foreground color of the widget text.

11	height	The vertical dimension of the widget. It is specified as the number of lines (not pixel).
12	highlightcolor	It represents the color of the focus highlight when the widget has the focus.
13	highlightbackground	The color of the focus highlight when the widget is not having the focus.
14	image	It can be set to an image object if we want to display an image on the radiobutton instead the text.
15	justify	It represents the justification of the multi-line text. It can be set to CENTER(default), LEFT, or RIGHT.
16	padx	The horizontal padding of the widget.
17	pady	The vertical padding of the widget.
18	relief	The type of the border. The default value is FLAT.
19	selectcolor	The color of the radio button when it is selected.
20	selectimage	The image to be displayed on the radiobutton when it is selected.
21	state	It represents the state of the radio button. The default state of the Radiobutton is NORMAL. However, we can set this to DISABLED to make the radiobutton unresponsive.
22	Text	The text to be displayed on the radiobutton.
23	textvariable	It is of String type that represents the text displayed by the widget.
24	underline	The default value of this option is -1, however, we can set this option to the number of character which is to be underlined.
25	value	The value of each radiobutton is assigned to the control variable when it is turned on by the user.
26	variable	It is the control variable which is used to keep track of the user's choices. It is shared among all the radiobuttons.
27	width	The horizontal dimension of the widget. It is represented as the number of characters.

28	wraplength	We can wrap the text to the number of lines by setting this option to the desired number so that each line contains only that number of characters.

Methods

The radiobutton widget provides the following methods.

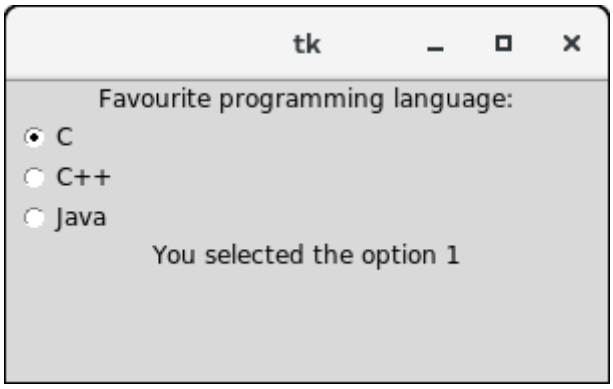
SN	Method	Description
1	deselect()	It is used to turn of the radiobutton.
2	flash()	It is used to flash the radiobutton between its active and normal colors few times.
3	invoke()	It is used to call any procedure associated when the state of a Radiobutton is changed.
4	select()	It is used to select the radiobutton.

Example

```
from tkinter import *

def selection():
   selection = "You selected the option " + str(radio.get())
   label.config(text = selection)

top = Tk()
top.geometry("300x150")
radio = IntVar()
lbl = Label(text = "Favourite programming language:")
lbl.pack()
R1 = Radiobutton(top, text="C", variable=radio, value=1,command=selection)
R1.pack( anchor = W )

R2 = Radiobutton(top, text="C++", variable=radio, value=2, command=selection)

R2.pack( anchor = W )

R3 = Radiobutton(top, text="Java", variable=radio, value=3,  command=selection)

R3.pack( anchor = W)
```

```
label = Label(top)
label.pack()
top.mainloop()
```

Output:



Python Tkinter Scrollbar

The scrollbar widget is used to scroll down the content of the other widgets like listbox, text, and canvas. However, we can also create the horizontal scrollbars to the Entry widget.

The syntax to use the Scrollbar widget is given below.

Syntax

w = Scrollbar(top, options)

A list of possible options is given below.

SN	Option	Description
1	activebackground	The background color of the widget when it has the focus.
2	Bg	The background color of the widget.
3	Bd	The border width of the widget.
4	command	It can be set to the procedure associated with the list which can be called each time when the scrollbar is moved.
5	cursor	The mouse pointer is changed to the cursor type set to this option which can be an arrow, dot, etc.
6	elementborderwidth	It represents the border width around the arrow heads and slider. The default value is -1.
7	Highlightbackground	The focus highlighcolor when the widget doesn't have the focus.
8	highlighcolor	The focus highlighcolor when the widget has the focus.
9	highlightthickness	It represents the thickness of the focus highlight.
10	jump	It is used to control the behavior of the scroll jump. If it set to 1,

		then the callback is called when the user releases the mouse button.
11	orient	It can be set to HORIZONTAL or VERTICAL depending upon the orientation of the scrollbar.
12	repeatdelay	This option tells the duration up to which the button is to be pressed before the slider starts moving in that direction repeatedly. The default is 300 ms.
13	repeatinterval	The default value of the repeat interval is 100.
14	takefocus	We can tab the focus through this widget by default. We can set this option to 0 if we don't want this behavior.
15	troughcolor	It represents the color of the trough.
16	width	It represents the width of the scrollbar.

Methods

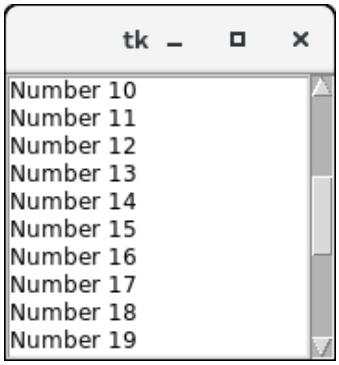
The widget provides the following methods.

SN	Method	Description
1	get()	It returns the two numbers a and b which represents the current position of the scrollbar.
2	set(first, last)	It is used to connect the scrollbar to the other widget w. The yscrollcommand or xscrollcommand of the other widget to this method.

Example

```
from tkinter import *

top = Tk()
sb = Scrollbar(top)
sb.pack(side = RIGHT, fill = Y)

mylist = Listbox(top, yscrollcommand = sb.set )

for line in range(30):
   mylist.insert(END, "Number " + str(line))

mylist.pack( side = LEFT )
sb.config( command = mylist.yview )

mainloop()
```

Output:



Python Tkinter Text

The Text widget is used to show the text data on the Python application. However, Tkinter provides us the Entry widget which is used to implement the single line text box.

The Text widget is used to display the multi-line formatted text with various styles and attributes. The Text widget is mostly used to provide the text editor to the user.

The Text widget also facilitates us to use the marks and tabs to locate the specific sections of the Text. We can also use the windows and images with the Text as it can also be used to display the formatted text.
The syntax to use the Text widget is given below.

Syntax

w = Text(top, options)

A list of possible options that can be used with the Text widget is given below.

SN	Option	Description
1	Bg	The background color of the widget.
2	Bd	It represents the border width of the widget.
3	Cursor	The mouse pointer is changed to the specified cursor type, i.e. arrow, dot, etc.
4	Exportselection	The selected text is exported to the selection in the window manager. We can set this to 0 if we don't want the text to be exported.
5	Font	The font type of the text.
6	Fg	The text color of the widget.
7	Height	The vertical dimension of the widget in lines.
8	highlightbackground	The highlightcolor when the widget doesn't has the focus.
9	highlightthickness	The thickness of the focus highlight. The default value is 1.
10	Highlighcolor	The color of the focus highlight when the widget has the focus.

11	Insertbackground	It represents the color of the insertion cursor.
12	insertborderwidth	It represents the width of the border around the cursor. The default is 0.
13	Insertofftime	The time amount in Milliseconds during which the insertion cursor is off in the blink cycle.
14	Insertontime	The time amount in Milliseconds during which the insertion cursor is on in the blink cycle.
15	Insertwidth	It represents the width of the insertion cursor.
16	Padx	The horizontal padding of the widget.
17	Pady	The vertical padding of the widget.
18	Relief	The type of the border. The default is SUNKEN.
19	Selectbackground	The background color of the selected text.
20	selectborderwidth	The width of the border around the selected text.
21	spacing1	It specifies the amount of vertical space given above each line of the text. The default is 0.
22	spacing2	This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. The default is 0.
23	spacing3	It specifies the amount of vertical space to insert below each line of the text.
24	State	It the state is set to DISABLED, the widget becomes unresponsive to the mouse and keyboard unresponsive.
25	Tabs	This option controls how the tab character is used to position the text.
26	Width	It represents the width of the widget in characters.
27	Wrap	This option is used to wrap the wider lines into multiple lines. Set this option to the WORD to wrap the lines after the word that fit into the available space. The default value is CHAR which breaks the line which gets too wider at any character.
28	Xscrollcommand	To make the Text widget horizontally scrollable, we can set this option to the set() method of Scrollbar widget.
29	Yscrollcommand	To make the Text widget vertically scrollable, we can set this option to the set() method of Scrollbar widget.

Methods

We can use the following methods with the Text widget.

SN	Method	Description
1	delete(startindex, endindex)	This method is used to delete the characters of the specified range.
2	get(startindex, endindex)	It returns the characters present in the specified range.
3	index(index)	It is used to get the absolute index of the specified index.
4	insert(index, string)	It is used to insert the specified string at the given index.
5	see(index)	It returns a boolean value true or false depending upon whether the text at the specified index is visible or not.

Mark handling methods

Marks are used to bookmark the specified position between the characters of the associated text.

SN	Method	Description
1	index(mark)	It is used to get the index of the specified mark.
2	mark_gravity(mark, gravity)	It is used to get the gravity of the given mark.
3	mark_names()	It is used to get all the marks present in the Text widget.
4	mark_set(mark, index)	It is used to inform a new position of the given mark.
5	mark_unset(mark)	It is used to remove the given mark from the text.

Tag handling methods

The tags are the names given to the separate areas of the text. The tags are used to configure the different areas of the text separately. The list of tag-handling methods along with the description is given below.

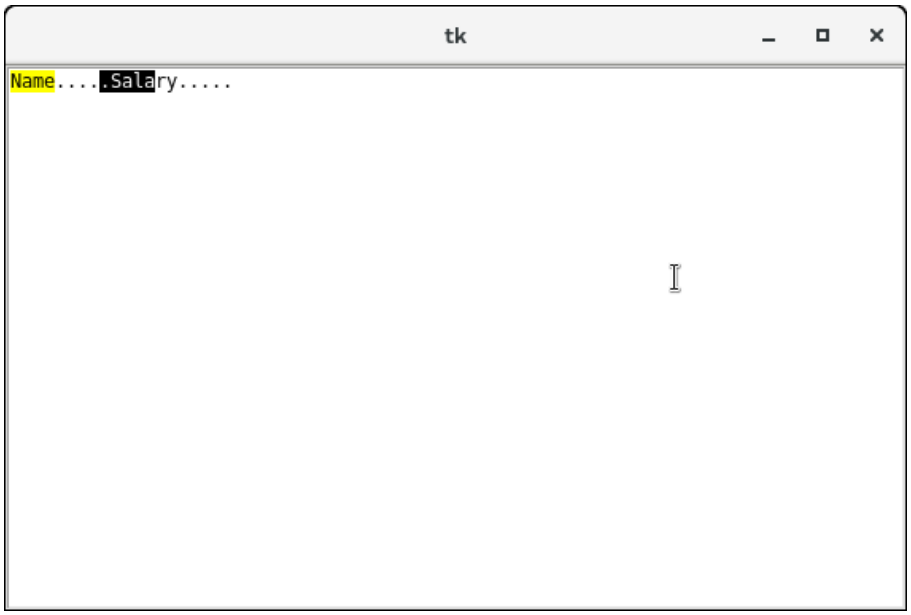
SN	Method	Description
1	tag_add(tagname, startindex, endindex)	This method is used to tag the string present in the specified range.
2	tag_config	This method is used to configure the tag properties.
3	tag_delete(tagname)	This method is used to delete a given tag.
4	tag_remove(tagname, startindex, endindex)	This method is used to remove a tag from the specified range.

Example

```
from tkinter import *
```

```
top = Tk()
text = Text(top)
text.insert(INSERT, "Name.....")
text.insert(END, "Salary.....")

text.pack()

text.tag_add("Write Here", "1.0", "1.4")
text.tag_add("Click Here", "1.8", "1.13")

text.tag_config("Write Here", background="yellow", foreground="black")
text.tag_config("Click Here", background="black", foreground="white")

top.mainloop()
```

Output:



Other GUIs

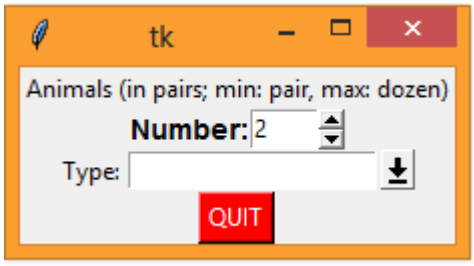
General GUI development using many of the abundant number of graphical toolkits that exist under Python, but alas, that is for the future. As a proxy, we would like to present a single simple GUI application written using four of the more popular and available toolkits out there: Tix (Tk Interface eXtensions), Pmw (Python MegaWidgetsTkinter extension), wxPython (Python binding to wxWidgets), and PyGTK (Python binding to GTK+).

Tix, the Tk Interface eXtension, is a powerful set of user interface components that expands the capabilities of your Tcl/Tk and Python applications. Using Tix together with Tk will greatly enhance the appearance and functionality of your application.

It uses the Tix module. Tix comes with Python!
Example:

```
from tkinter import *
from tkinter.tix import Control, ComboBox

top = Tk()
top.tk.eval('package require Tix')

lb = Label(top,text='Animals (in pairs; min: pair, max: dozen)')
lb.pack()

ct = Control(top, label='Number:',integer=True, max=12, min=2, value=2, step=2)
ct.label.config(font='Helvetica -14 bold')
ct.pack()

cb = ComboBox(top, label='Type:', editable=True)
for animal in ('dog', 'cat', 'hamster', 'python'):
cb.insert(END, animal)
cb.pack()

qb = Button(top, text='QUIT',command=top.quit, bg='red', fg='white')
qb.pack()

top.mainloop()
```

Output:



Python Mega Widgets

PMW (Python Mega Widgets) is a toolkit for building high -level widgets in Python using the Tkinter module. This toolkit provides a frame work that contains a variety of widgets richer than the one provided by Tkinter.

It basically helps the extend its longevity by adding more modern widgets to the GUI Palette.This package is 100% written in Python, which turns out to be a cross -platform

widget library. Being highly configurable allows it to create additional widget collections by extending the basic Tkinter widget core set.

PMW provides many interesting and complex widgets, including: About Dialog, Balloon, Button Box, Combo Box, Combo Box Dialog, Counter, CounterDialog, Dialog, Entry Field, Group, Labeled Widget, MenuBar, Message Bar, Message Dialog, Note BookR, Note BookS, Note Book, Option Menu, Paned Widget, Prompt Dialog, RadioSelect, Scrolled Canvas, Scrolled Field, ScrolledFrame, Scrolled Listbox, ScrolledText, Selection Dialog, Text Dialog, and Time Counter.

Example:

```
from Tkinter import Button, END, Label, W
from Pmw import initialise, ComboBox, Counter

top = initialise()

lb = Label(top,
text='Animals (in pairs; min: pair, max: dozen)')
lb.pack()

ct = Counter(top, labelpos=W, label_text='Number:',
datatype='integer', entryfield_value=2,
increment=2, entryfield_validate={'validator':'integer', 'min': 2, 'max': 12})
ct.pack()
cb = ComboBox(top, labelpos=W, label_text='Type:')
for animal in ('dog', 'cat', 'hamster', 'python'):
cb.insert(end, animal)
cb.pack()

qb = Button(top, text='QUIT',
command=top.quit, bg='red', fg='white')
qb.pack()
```

wxWidgets and wxPython

wxWidgets (formerly known as wxWindows) is a cross-platform toolkit used to build graphical user applications. It is implemented using C++ and is available on a wide number of platforms to which wxWidgets defines a consistent and common API. The best part of all is that wxWidgets uses the native GUI on each platform, so your program will have the same ook-and-feel as all the other applications on your desktop. Another feature is

that you are not restricted to developing wxWidgets applications in C++. There are interfaces to both Python and Perl.

Related Modules and Other GUIs

There are other GUI development systems that can be used with Python. We present the appropriate modules along with their corresponding window systems. Table represents the GUI Systems Available for Python

GUI Module or System Description

Tk-Related Modules	
Tkinter	TK INTERface: Python's default GUI toolkit http://wiki.python.org/moin/TkInter
Pmw	Python MegaWidgets (Tkinter extension) http://pmw.sf.net
Tix	Tk Interface eXtension (Tk extension) http://tix.sf.net
TkZinc (Zinc)	Extended Tk canvas type (Tk extension) http://www.tkzinc.org
EasyGUI (easygui)	Very simple non-event-driven GUIs (Tkinter extension) http://ferg.org/easygui
TIDE + (IDE Studio)	Tix Integrated Development Environment (including IDE Studio, a Tix-enhanced version of the standard IDLE IDE) http://starship.python.net/crew/mike

wxWidgets-Related Modules	
`wxPython`	Python binding to wxWidgets, a cross-platform GUI framework (formerly known as wxWindows)http://wxpython.org
Boa Constructor	Python IDE and wxPython GUI builder http://boa-constructor.sf.net
`PythonCard`	wxPython-based desktop application GUI construction kit (inspired by HyperCard) http://pythoncard.sf.net
wxGlade	another wxPython GUI designer (inspired by Glade, the GTK+/GNOME GUI builder) http://wxglade.sf.net
GTK+/GNOME-Related Modules	
PyGTK	Python wrapper for the GIMP Toolkit (GTK+) library http://pygtk.org
`GNOME-Python`	Python binding to GNOME desktop and development libraries http://gnome.org/start/unstable/bindings http://download.gnome.org/sources/gnome-python
`Glade`	a GUI builder for GTK+ and GNOME http://glade.gnome.org
PyGUI(`GUI`)	cross-platform "Pythonic" GUI API (built on Cocoa [MacOS X] and GTK+ [POSIX/X11 and Win32]) http://www.cosc.canterbury.ac.nz/~greg/python_gui
Qt/KDE-Related Modules	
PyQt	Python binding for the Qt GUI/XML/SQL C++ toolkit from Trolltech (partially open source [dual-license]) http://riverbankcomputing.co.uk/pyqt
PyKDE	Python binding for the KDE desktop environment http://riverbankcomputing.co.uk/pykde
eric	Python IDE written in PyQt using QScintilla editor widget http://die-offenbachs.de/detlev/eric3 http://ericide.python-hosting.com/
PyQtGPL	Qt (Win32 Cygwin port), Sip, QScintilla, PyQt bundle http://pythonqt.vanrietpaap.nl
Other Open Source GUI Toolkits	
FXPy	Python binding to FOX toolkit (http://fox-toolkit.org) http://fxpy.sf.net
pyFLTK (`fltk`)	Python binding to FLTK toolkit (http://fltk.org) http://pyfltk.sf.net
PyOpenGL (`OpenGL`)	Python binding to OpenGL (http://opengl.org) http://pyopengl.sf.net
Commercial	
`win32ui`	Microsoft MFC (via Python for Windows Extensions) http://starship.python.net/crew/mhammond/win32
`swing`	Sun Microsystems Java/Swing (via Jython) http://jython.org