Module - 4

Tkinter

- The wxPython is a platform independent toolkit for supporting GUI in python. The implementation of GUI with this toolkit is easy, but since wxPython gets frequently updated, frequent path updation are required for new feature requests.
- The Jython supports GUI for integrating with the java platform. It is mainly used for embedded scripting, interactive interpreter and rapid application development.
- The Tkinter is also called as Tk interface. It is a standard toolkit in python for GUI. It binds Tk widgets to python objects. To use this, a working Tcl/Tk system is required. The advantage of Tkinter is that is more portable than other GUI toolkits.
- Python offers multiple options for developing GUI (Graphical user interface). Out of all the GUI methods, tkinter is the most commonly used method. It is standard python interface to the Tk GUI toolkit shipped with python.

- Python with tkinter is the fastest and e way to create the GUI applications creating a GUI using tkinter is an easy task.

⇒ To create a tkinter app:

(i) importing the module - tkinter.

(ii) create the main window (container)

(iii) Add any number of widgets to the main win

(iv) Apply the event Trigger on the widgets.

- Importing tkinter is same as importing any module in the python code

```
import tkinter.
```

- There are two main methods used which th user needs to remember while creating the pyth application with GUI

(1) Tk(screenName = None, baseName = None, clas = 'Tk', useTk = 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:

m = 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 your application is ready to run. mainloop() is an infinite loop used to run the application, wait for an event to occur and process the event as long as the window is not closed.

m.mainloop()

⇒ Tkinter widgets.

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

Tkinter is implemented as a python wrapped for the Tcl interpreter embedded within the interpreter of python. Tk provides the following widgets:-

- button
- canvas
- combo-box
- entry

- frame
- labelframe
- listbox
- menu
- menubutton
- message
- notebook
- optionmenu
- panedwindow
- progressbar
- radiobutton
- scale
- scrollbar
- separator
- sizebox
- spinbox
- text
- treeview.

The common properties of widgets include color, fonts, dimensions, anchors, bitmaps and cursors.
The tkinter place all the components in the main window by using any one of the following methods.

- pack() method places components in blocks of parent widget.
- grid() method places components in blocks of parent widget
- place() method places components within the specified position of parent widget

⇒ Tkinter Label

This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

Syntax

Here is the simple syntax to create this widget -

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

Parameters:

- master :- This represents the parent window.
- options - Here is the list of most commonly used options for this widget. These options can used as key-value pairs separated by commas Commanly used options are height, width, font, bgcolor etc.

Example :-

```
from Tkinter import *
root = Tk()
var = StringVar()
```

```
label = Label (root, textvariable = var)
var.set ("Hey!? How are you doing?")
label.pack()
root.mainloop()
```

Result:-

Hey!? How are you doing.

⇒ Python - Tkinter Message.

This widget provides a multiline and noneditable object that displays ~~texts~~ texts, automatically breaking lines and justifying their contents.

Its functionality is very similar to the one provided by the label widget, except that it can also automatically wrap the text, maintaining a given width or aspect ratio.

Syntax

Here is the simple syntax to create this widget

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

Example:-

```
from Tkinter import *
root = Tk()
var = StringVar()
label = Message(root, textvariable = var, relief = RAISED)
var.set("Hey!? How are you doing?")
label.pack()
root.mainloop()
```

Result:-

Hey!? How
are you
doing?

⟹ Tkinter Entry:

The Entry widget is used to accept single-line text strings from a user.

- If you want to display multiple lines of text that can be edited, then you should use the Text widget.
- If you want to display one or more lines of text that cannot be modified by the user then you should use the Label widget.

Syntax.

Here is the simple syntax to create this widget.

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

Example

```
from Tkinter import *
top = TK()
L1 = Label (top, text = "User Name")
L1.pack (side = LEFT)
E1 = Entry (top, bd = 5)
E1.pack (side = RIGHT)
top.mainloop()
```

When the above code is executed, it produces the following result

User Name:

⇒ Python - Tkinter Text.

Text widgets provide advanced capabilities that allow you to edit a multiline text and format the way it has to be displayed, such a changing its color and font.

you can also use elegant structures like tabs and marks to locate specific sections of the text, and apply changes to those areas. Moreover, you can embed windows and images in the text because this widget was designed to handle both plain and formatted text.

Syntax

Here is the simple syntax to create this widget -

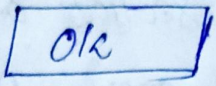
W = Text (master, option, ...)

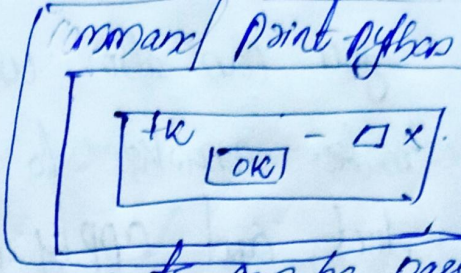
Button widget:

The button widget is used to add a button in the GUI window.

```
from tkinter import *
def abc():
    print ("OK button is clicked")
a = Tk()
b1 = Button (a, text = "OK", width = 30, command = abc
                                                    Pack()
a.mainloop()
```

output :

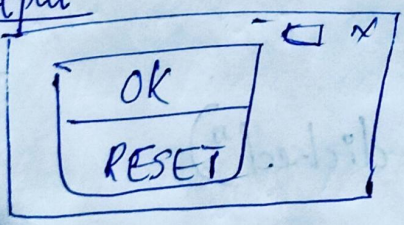




is case of buttons, the arguments can be pass to the command function by using expression.

```
from tkinter import *
def abc(number):
    Print(number, "button is clicked")
a = Tk()
b1 = Button(a, text="Ok", width=30, command =
                lambda:abc(1).Pack()
b2 = Button(a, text="RESET", width=30, command =
                lambda:abc(2).Pack()
a.mainloop()
```

output



⇒ checkbutton widget :

The checkbutton widget is used to add the checkbutton in the GUI window.

```
from tkinter import *
a=Tk()
var1=IntVar()
var2=IntVar()
var3=IntVar()
var4=IntVar()
c1=Checkbutton(a,text="Value 1",variable=var1).p
c2=Checkbutton(a,text="Value 2",variable=var2).pack()
c3=Checkbutton(a,text="Value 3",variable=var3).pack()
c4=Checkbutton(a,text="Value 4",variable=var4).pack()
Print("Value of var1:",var1)
Print("Value of var2:",var2)
Print("Value of var3:",var3)
Print("Value of var4:",var4)
a.mainloop()
```

⇒ Entry Widget

The entry widget is used to add the textbox in the GUI window. The entry widget will take single line text input from the user.

```
from tkinter import *
a=Tk()
l1=Label(a,text="Enter your name").pack()
e1=Entry(a).pack()
b1=Button(a,text="SUBMIT").pack()
a.mainloop()
```

⇒ Listbox widget:

The Listbox widget is used to add the drop down list box component in the GUI window. It allow user to select multiple options. The elements can be added with insert() method and deleted with deleat() method.

⇒ Frame widget:

The Frame widget is used to arrange other widgets in the container. In general, the other widget are under the control of geometry manager.

```
from tkinter import *
a=Tk()
def abc():
    # print(e1.get())
    te = e1.get()
    f = Frame(a)
    f.pack()
    l = Label(f, text = te).pack()
l1 = Label(a, text = "Enter your name").pack()
e1 = Entry(a)
e1.pack()
b1 = Button(a, text = "SUBMIT", command = abc).pack()
a.mainloop()
```

⇒ LabelFrame Widget.

The LabelFrame widget is a variant of Frame widget. The LabelFrame widget draws a border around its child widgets. It also the title display property. The LabelFrame widget is based for grouping multiple related widgets.

```
from tkinter import *
m=Tk()
g=LabelFrame(m, text="Entry widget group",
padx=10, pady=10)
g.pack(padx=20, pady=20)
W1=Entry(g)
W1.pack()
W2=Entry(g)
W2.pack()
mainloop()
```

② Message widget:

The message widget is used to display multiple lines of text. It is a variant of Label widget where the Label widget is used to display single widget.

```
from tkinter import *
m=Tk()
W=Message(m, text="this is a message widget
        multi line message")
W.pack()
mainloop()
```

⇒ combobox widget

The combobox widget is used to add the drop down selection element in the window.

```
from tkinter import *
from tkinter import ttk
m = Tk()
ttk.Label(m, text = 'Choose the colour:').grid(column=1, row
n = ttk.Combobox(m, width = 12)#3.
n['values'] = ("red", "green", "blue", "orange", "yellow")
n.grid(column=1, row = 1)
n.current(0)
mainloop().
```

⇒ Scale widget

The Scale widget allows the user to give a numerically bounded input and the Scale bar. it is similar to Entry widget but it will select the data by using the Scale box.

```
from tkinter import *
m = Tk()
x1 = Scale(m, from_ = 0, to = 5)
x1.pack()
x2 = Scale(m, from_ = 0, to = 6, orient = HORIZONTAL
```

```
x2.pack()
x3=Scale(m, from_=0, to=7, orient=VERTICAL)
x3.pack()
mainloop()
```

⇒ Canvas widget:

The canvas widget is used to draw the plo and graphics in Tkinter. It is used for designing the various custom widget.

Eg:
```
from tkinter import*
m=Tk()
x=canvas(m, width=200, height=100)
x.pack()
x.create_line(0,0,200,100)
x.create_rectangle(50,25,150,75, fill="red")
x.create_oval(100,20,150,75, fill="green")
x.create_oval(50,20,100,75, fill="blue")
mainloop().
```