Python Tkinter Tutorial

Tkinter tutorial provides basic and advanced concepts of Python Tkinter. Our Tkinter tutorial is designed for beginners and professionals.

Python provides the standard library Tkinter for creating the graphical user interface for desktop based applications.

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.

- import the Tkinter module.
- Create the main application window.
- Add the widgets like labels, buttons, frames, etc. to the window.
- Call the main event loop so that the actions can take place on the user's computer screen.

Example

```
from tkinter import *

#creating the application main window.

top = Tk()

#Entering the event main loop

top.mainloop()
```

Tkinter widgets

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 canvas on the window.

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.
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.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

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

Let's discuss each one of them in detail.

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

However, the controls are less and widgets are generally added in the less organized manner.

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.

expand: If the expand is set to true, the widget expands to fill any space.

Fill: By default, the fill is set to NONE. However, we can set it to X or Y to determine whether the widget contains any extra space.

size: 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()
```

Python Tkinter grid() 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.

Columnspan

The width of the widget. It represents the number of columns up to which, the column is expanded.

ipadx, ipady

It represents the number of pixels to pad the widget inside the widget's border.

padx, pady

It represents the number of pixels to pad the widget outside the widget's border.

row

The row number in which the widget is to be placed. The topmost row is represented by 0.

rowspan

The height of the widget, i.e. the number of the row up to which the widget is expanded.

Sticky

If the cell is larger than a widget, then sticky is used to specify the position of the widget inside the cell. It may be the concatenation of the sticky letters representing the position of the widget. It may be N, E, W, S, NE, NW, NS, EW, ES.

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()
```

Python Tkinter place() 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.

Anchor: It represents the exact position of the widget within the container. The default value (direction) is NW (the upper left corner)

bordermode: The default value of the border type is INSIDE that refers to ignore the parent's inside the border. The other option is OUTSIDE.

height, width: It refers to the height and width in pixels.

relheight, relwidth: It is represented as the float between 0.0 and 1.0 indicating the fraction of the parent's height and width.

relx, rely: It is represented as the float between 0.0 and 1.0 that is the offset in the horizontal and vertical direction.

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()
```

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)
```

Example

#python application to create a simple button

```
from tkinter import *

top = Tk()

top.geometry("200x100")

b = Button(top,text = "Simple")

b.pack()

top.mainaloop()
```

Example

```
from tkinter import *

top = Tk()

top.geometry("200x100")

def fun():

   messagebox.showinfo("Hello", "Red Button clicked")

b1 = Button(top,text = "Red",command = fun,activeforeground =
"red",activebackground = "pink",pady=10)

b2 = Button(top, text = "Blue",activeforeground = "blue",activebackground =
"pink",pady=10)

b3 = Button(top, text = "Green",activeforeground = "green",activebackground
= "pink",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()
```

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)
```

Example

```
        from tkinter import *

        top = Tk()

        top.geometry("200x200")

        #creating a simple canvas

        c = Canvas(top,bg = "pink",height = "200")

        c.pack()

        top.mainloop()
```

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()
```

Python Tkinter Checkbutton

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.

Syntax

```
        w = checkbutton(master, options)
```

Example

```
from tkinter import *

top = Tk()

top.geometry("200x200")

checkvar1 = IntVar()

checkvar2 = IntVar()

checkvar3 = IntVar()

chkbtn1 = Checkbutton(top, text = "C", variable = checkvar1, onvalue = 1,
offvalue = 0, height = 2, width = 10)

chkbtn2 = Checkbutton(top, text = "C++", variable = checkvar2, onvalue = 1,
offvalue = 0, height = 2, width = 10)
```

```
chkbtn3 = Checkbutton(top, text = "Java", variable = checkvar3, onvalue = 1,
offvalue = 0, height = 2, width = 10)

chkbtn1.pack()

chkbtn2.pack()

chkbtn3.pack()

top.mainloop()
```

Python Tkinter Entry

The Entry widget is used to provde 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.

Syntax

```
w = Entry (parent, options)
```

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",
activeforeground = "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()
```

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, column=0)

root.mainloop()
```

Python Tkinter Listbox

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.

Syntax

```
w = Listbox(parent, options)
```

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()
```

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()
```

Python Tkinter Menubutton

The Menubutton widget can be defined as the drop-down menu that is shown to the user all the time. It is used to provide the user a option to select the appropriate choice exist within the application.

The Menubutton is used to implement various types of menus in the python application. A Menu is associated with the Menubutton that can display the choices of the Menubutton when clicked by the user.

Syntax

```
        w = Menubutton(Top, options)
```

Example

```
from tkinter import *

top = Tk()

top.geometry("200x250")

menubutton = Menubutton(top, text = "Language", relief = FLAT)
```

```
menubutton.grid()

menubutton.menu = Menu(menubutton)

menubutton["menu"]=menubutton.menu

menubutton.menu.add_checkbutton(label = "Hindi", variable=IntVar())

menubutton.menu.add_checkbutton(label = "English", variable = IntVar())

menubutton.pack()

top.mainloop()
```

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.

Syntax

```
w = Menu(top, options)
```

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(root)

menubar.add_command(label="Hello!", command=hello)

menubar.add_command(label="Quit!", command=top.quit)
```

```
# display the menu

top.config(menu=menubar)

top.mainloop()
```

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()
```

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.

Syntax

```
w = Message(parent, options)
```

Example

```
from tkinter import *

top = Tk()

top.geometry("100x100")

var = StringVar()

msg = Message( top, text = "Welcome to Javatpoint")

msg.pack()

top.mainloop()
```