10. Create GUI Apps in Python Using Tkinter

Tkinter is **Python's** de-facto standard GUI (**Graphical User Interface**) package.

Tkinter is most commonly used as $\boxed{\textbf{GUI Programming toolkit}}$ for Python.

Tkinter is a **Python** binding to **the** $\boxed{\textbf{Tk GUI toolkit.}}$ It is the standard **Python** interface to the Tk GUI toolkit.

It is a thin object-oriented layer **on top of Tcl/Tk.**

Tkinter is **included** with the standard Microsoft Windows and Mac OS X install of **Python**.

The name **Tkinter** comes from $\boxed{\textbf{Tk interface}}$**.**

What is TCL and Tk?
Tcl is a general purpose **multi-paradigm system programming language**. It is a scripting language that aims at providing the ability for applications to communicate with each other.
On the other hand, **Tk** is a cross platform **widget toolkit** **used for building GUI in many languages.**

What is the best GUI for Python?

What platform-independent GUI toolkits exist for Python?

- Tkinter. Standard builds of Python include an **object-oriented interface to the Tcl/Tk widget set,** called Tkinter. ...
- wxWidgets. ...
- Qt....
- Gtk+ ...
- Kivy. ...
- FLTK. ...
- OpenGL.

1. Label: **lebel1=Label (root, text=' ', bg=' ',fg=' ')**

2. Frame **frm=Frame (root)**

3. Button **b1=Button (root, text=' ', fg= ' ') //fg=text color**

#10.92 Tkinter Hello world Program

[Que1. Print hello world]

Python application using Tkinter with GUI contains dropdown menus, buttons, windows checkbox, radio buttons, and textboxes and so on.

Use

```
  import tkinter

from tkinter import *        …//Where * is going to import all the things from tkinter module
```

code:

```
from tkinter import *

root=Tk()                           // Tk () is tkinter class and root is  object of class

label1=Label (root, text="Hello world")   //create label

label1.pack ()                          //.pack () function will attach label to window

root.mainloop ()                       // to keep window running use object.mainloop () function
```

#10.93 Tkinter: Using frames

[Que2.Create two frames & add button on each frame with text message]

```
from tkinter import *
root=Tk()                                   # root is object of class Tk()
frm1=Frame(root)                            //frame1 will be placed at root/ windows
frm1.pack(side=TOP)                         // .pack() function will add frame on window by
default at top
frm2=Frame(root)                            //frame2 will be placed at root/ windows
frm2.pack(side=BOTTOM)                      // to add frame2 on windows @ bottom side
//to add button1 on  frame1 syntax is given… fg=foreground colour/text colour
b1=Button(frm1,text="Click Here",fg="RED")
b2=Button(frm2,text="Press on this",fg="BLUE")

// to add buttons on respective frames  .pack() function is used
b1.pack()
b2.pack()
root.mainloop()
```

#10.94 Tkinter: Grid layout

[Que.Create application with labels and corresponding entries. Using grid]

Grid Layout is another way to place elements/widgets on window .

Arranging widgets /elements in **grid format [rows number, Column number].**

Text field entries in tkinter is called as entries

Entry (root): in tkinter is used for test field entry

e.g

	column 0	1
Row 0	Label1	**Entry1**
1	Label2	Entry2
2		

```
from tkinter import *

root=Tk()

l1=Label (root, text="Frist Name")

l2=Label (root, text="Last Name")

entry1=Entry (root)                    //used for text entry…entry is input… cannot be packed

entry2=Entry (root)                   //used for another text entry

l1.grid (row=0, column=0)

l2.grid (row=1, column=0)

entry1.grid (row=0, column=1)

entry2.grid (row=1,column=1)

root.mainloop ()
```

#10.95 Tkinter: Self Adjusting Widgets

Creating widgets that are self-adjusting. That is they increase or grows with the size of our window to fit size.

[Que. Create label (self-adjusting to window) will enhance till size of window]

from tkinter import *

root=Tk()

label1=Label (root, text="First Name”, bg="YELLOW”, fg="RED")

// used for width of text to the window

label1.pack (side=TOP, fill=X)

label2=Label (root, text="Second Name”, fg="BLUE”, bg="GREEN")

//**increase the word to the length of window, side left will put label to left side of window**

label2.pack (side=LEFT, fill=Y)

root.mainloop ()

//bg=background color , fg=foreground color/text color

```
from tkinter import *

root=Tk()

label1=Label (root, text='First Name', bg='YELLOW', fg='RED')


# used for width of text to the window

label1.pack (side=TOP, fill=X)

label2=Label (root, text='Second Name', fg='BLUE', bg='GREEN')


#increase the word to the length of window, side left will put label to left side of window

label2.pack (side=LEFT, fill=Y)

root.mainloop ()


#bg=background color ,  fg=foreground color/text color
```

#10.96 Tkinter: Handling Button clicks

Handling button clicks means when we click on the button code written for that button gets executed.

Write a function with some line of codes and pass this function to the button **as a command** when we click on that button, then command will execute code written or that function

```
from tkinter import *                    // import all functions from tkinter

root=Tk()

def function1():

   print("Hello You just clicked here")

b1=Button (root, text="click”, command=function1)    // function1 is passed to command

b1.pack()

root.mainloop()
```

#10.97 Tkinter: Using classes

Create GUI application in python using class. Adding buttons,frames,entry using class.

Classes->attributes->methods

```
from tkinter import *
root=Tk ()
class mybutton:
  def __init__(self, root):          //attributes
    frm=Frame(root)
    frm.pack ()

    self.butt1=Button (frm,text="click",fg="RED",command=self.printdata)
    self.butt1.pack (side=TOP)

    self.butt2=Button(frm,text="Exit",command=frm.quit)
    self.butt2.pack ()

  def printdata(self):               //function printdata
    print("Hello How r You?")

b=mybutton(root)
root.mainloop()
```

```
Hello How r You?
Hello How r You?
```

#10.98 Tkinter: Using drop downs

```
# [Que. Create Drop down menu]

from tkinter import *

root=Tk()

def fun1():

   print("You just clicked menu")

def fun2():

   print("Hi ")

mymenu=Menu(root)

submenu=Menu(mymenu)

root.config(menu=mymenu)
```

//They are attached to a parent menu (using **add_cascade*****), instead of a top-level window.***

```
mymenu.add_cascade(label="File",menu=submenu)

submenu.add_command(label="NewNotebook",command=fun1)

submenu.add_command(label="open",command=fun2)

submenu.add_separator ()

submenu.add_command(label="save",command=fun2)

newmenu=Menu(mymenu)

mymenu.add_cascade(label="edit",menu=newmenu)

newmenu.add_command(label="Undo",command=fun1)

root.mainloop()
```

#10.99 Tkinter: Toolbar

[Que: Create toolbar]

Toolbar is nothing but **essentially Frame.**

//padx pady is padding/spacing in x and y direction

```
from tkinter import *

root=Tk()

def fun1():

   print("You just clicked menu")

def fun2():

   print("Hi ")

toolbar1=Frame (root, bg="RED")

butt1=Button (toolbar1,text="print",command=fun1)

butt1.pack (side=LEFT,padx=2,pady=2)       /

butt2=Button(toolbar1,text="##",command=fun2)

butt2.pack(side=LEFT,padx=3,pady=3)

butt3=Button(toolbar1,text="++",command=fun2)

butt3.pack(side=LEFT,padx=3,pady=3)

toolbar1.pack(side=TOP,fill=X)    //fill=x will adjust toolbar with width of window

root.mainloop()
```

#10.100 Tkinter: Making Status Bar

[Que. Create status bar]

Most applications has a status **bar at the bottom** of each application window.

```
status = Label (root, text="This is status ", bd=1, relief=SUNKEN, anchor=W)
```

In the label, first, we will specify where we have to put the status bar.

Here, we have one window so we are using root.

The text is used to specify the text to be shown on the label.

Bd is used for the border. Whenever we create a label, we can add border to it.

Relief is a **parameter that allows us to manage how we want our label to appear**.

Here, we want it to appear **sunken** in our screen

. If we add just a label at the bottom of the screen, it looks kind of weird, but if we have border and relief, it looks like a part of the window.

Anchor allow us to decide where the label should be pinned. W is used for the West, N is for North, E is for East and S is for South. We want to show it on the left side of the screen so we have used W.

Now, to show this status bar in our main window, we need to pack it up.

```
from tkinter import *

root=Tk()

statusbar1=Label(root, text="Hi This is status", anchor=W,bd=1,relief=SUNKEN)

statusbar1.pack(side=BOTTOM, fill=X)

root.mainloop()
```

Here is list of possible constants which can be used for relief attribute.

- FLAT
- RAISED
- SUNKEN
- GROOVE
- RIDGE

#10.101 Tkinter: Message Box

[Que. Generate message box simple and with response message box**]**

Message box is nothing but simply **pop-up appears on screen.**

First import message as

```
import tkinter.messagebox

tkinter.messagebox.showinfo (“title”, “message want to display”)

res=tkinter.mesagebox.askquestion(“title”,” question”)
```

```
from tkinter import *

import tkinter.messagebox

root=Tk()

tkinter.messagebox.showinfo ("title”, “This is awesome")

res=tkinter.messagebox.askquestion("Que.1","Do you want Tea?")

if res=='yes':

   print("Here is tea")

else:

   print("Ok")

   root.mainloop()
```

#10.102 Tkinter Drawing

[Que. Drawing Graphics n particular canvas]

The **Canvas widget** supplies **graphics facilities** for Tkinter. Among these **graphical objects are lines, circles, images, and even other widgets.**

w = Canvas (master, option=value,)

 master − This represents the parent window.

 options − Here is the list of most commonly used options for this widget.

Sr. no.	Option & Description
1	**bd** Border width in pixels. Default is 2.
2	**bg** Normal background color.
3	**confine** If true (the default), the canvas cannot be scrolled outside of the scroll region.
4	**cursor** Cursor used in the canvas like *arrow, circle, dot etc.*
5	**height** Size of the canvas in the Y dimension.
6	**highlightcolor** Color shown in the focus highlight.
7	**relief** Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.
8	**scroll region** A tuple (w, n, e, s) that defines over how large an area the canvas can be scrolled, where w is the left side, n the top, e the right side, and s the bottom.
9	**width** Size of the canvas in the X dimension.

10	**xscrollincrement** If you set this option to some positive dimension, the canvas can be positioned only on multiples of that distance, and the value will be used for scrolling by scrolling units, such as when the user clicks on the arrows at the ends of a scrollbar.
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.

```
from tkinter import *
master=Tk()
root=Tk()
cn1=Canvas (master, width=300, height=300, bg=’YELLOW’, bd=3)
cn1.pack ()
n1=cn1.create_line (0, 0,100,100)
n2=cn1.create_line (50, 50, 50, 180, fill=’RED’)
n3=cn1.create_rectangle (100, 100, 200, 200, fill=’PINK’)
root.mainloop ()
```