

Create UI in Python-Tkinter

Modern computer applications are user-friendly. User interaction is not restricted to console-based I/O. They have a more user-friendly graphical user interface (GUI). These applications can receive inputs through mouse clicks and can enable the user to choose from alternatives with the help of radio buttons, dropdown lists, and other GUI elements (or widgets).

Such applications are developed using one of various graphics libraries available. A graphics library is a software toolkit having a collection of classes that define the functionality of various GUI elements. These graphics libraries are generally written in C/C++. Many of them have been ported to Python in the form of importable modules. Some of them are listed below:

Tkinter is the Python port for the Tcl-Tk GUI toolkit developed by Fredrik Lundh. This module is bundled with standard distributions of Python for all platforms.

Tkinter is a standard library in Python which is used for GUI application. Tkinter has various controls which are used to build a GUI-based application.

To install Tkinter, we need Python pre-installed. Tkinter actually comes along when we install Python. While installing Python, we need to check the `td/tk` and `IDLE` checkbox. This will install the `tkinter` and we need not install it separately.

However, if we missed installing Tkinter while installing Python, we can do it later using the `pip` command.

Step 1 – Make sure Python and pip is preinstalled on your system

Type the following commands in the command prompt to check if **python** and **pip** is installed on your system.

To check Python

```
python --version
```

If python is successfully installed, the version of python installed on your system will be displayed.

To check pip

```
pip -v
```

The version of pip will be displayed, if it is successfully installed on your system. If not installed it can be installed with the command

```
apt install python3-pip
```

Step 2 – Install Tkinter

Tkinter can be installed using pip. The following command is run in the command prompt to install Tkinter.

```
pip install tk
```

This command will start downloading and installing packages related to the Tkinter library. Once done, the message of successful installation will be displayed.

It can also be installed by using the command

```
sudo apt-get install python3-tk
```

What is Tkinter?

Tkinter is the inbuilt python module that is used to create GUI applications. It is one of the most commonly used modules for creating GUI applications in Python as it is simple and easy to work with. It gives an object-oriented interface to the Tk GUI toolkit.

Some other Python Libraries available for creating our own GUI applications are

Kivy, Python Qt, wxPython. Among all Tkinter is the most widely used.

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 a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create GUI applications.

To create a tkinter app we need to:

- ❖ Importing the module – tkinter
- ❖ Create the main window (container)
- ❖ Add any number of widgets to the main window
- ❖ Apply the event Trigger on the widgets.

Importing tkinter is the same as importing any other module in the Python code.

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

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

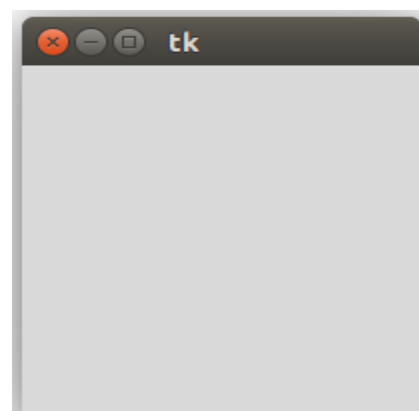
To create a main window, tkinter offers a method Tk(screenName=None, baseName=None, className='Tk', useTk=1). To change the name of the window, you can change the className to the desired one. The basic code used to create the main window of the application is:

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

mainloop(): There is a method known by the name **mainloop()** that 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()

```
from tkinter import *
root = Tk()
'''
widgets are added here
'''
root.mainloop()
```



tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes.

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

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

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

There are a number of widgets which you can put in your tkinter application. Some of the major widgets are explained below:

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

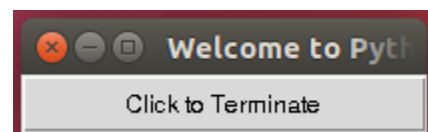
The general syntax is: `Val=Button(master, option=value)`

master is the parameter used to represent the parent window.

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

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

```
from tkinter import *
root = Tk()
root.title("Welcome to Python GUI ....")
my_button = Button(root, text='Stop', width=25, command=root.destroy)
my_button.pack()
root.mainloop()
```



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

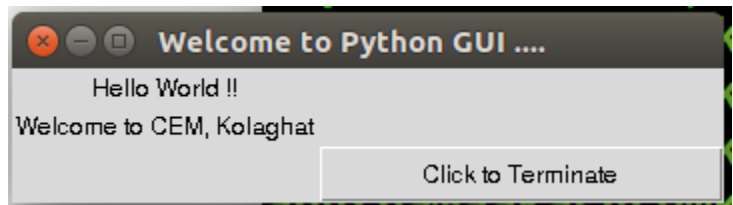
The general syntax is: `w=Label(master, option=value)`

master is the parameter used to represent the parent window.

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

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

```
from tkinter import *
root = Tk()
root.title('Welcome to Python GUI ....')
#create a label widget
my_label1=Label(root, text="Hello World !!")
my_label2=Label(root, text="Welcome to CEM, Kolaghat")
my_button = Button(root, text=' Click to Terminate ', width=25, command=root.destroy)
#Place the Widget on the root window
my_label1.grid(row=0,column=0)
my_label2.grid(row=1,column=0)
my_button.grid(row=2,column=1)
root.mainloop()
```



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

The general syntax is: `w=Entry(master, option=value)`

master is the parameter used to represent the parent window.

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

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

```
from tkinter import *
root = Tk()
root.title('Welcome to Python GUI ....')
#create label widgets
my_label1=Label(root, text="Enter Your First Name ")
my_label2=Label(root, text="Enter Your Last Name ")
my_label1.grid(row=0,column=0)
my_label2.grid(row=1,column=0)

#create text widgets
text1=Entry(root)
text2=Entry(root)
text1.grid(row=0,column=1)
text2.grid(row=1,column=1)

def show():
    my_label3=Label(root,text="Hello "+text1.get() +" "+ text2.get()+"!!")
    my_label3.grid(row=2,column=1)

my_button1 = Button(root, text=' Click to Display ', width=25,command=show)
my_button1.grid(row=3,column=1)
my_button2 = Button(root, text=' Click to Terminate ', width=25,command=root.destroy)
my_button2.grid(row=3,column=0)

root.mainloop()
```

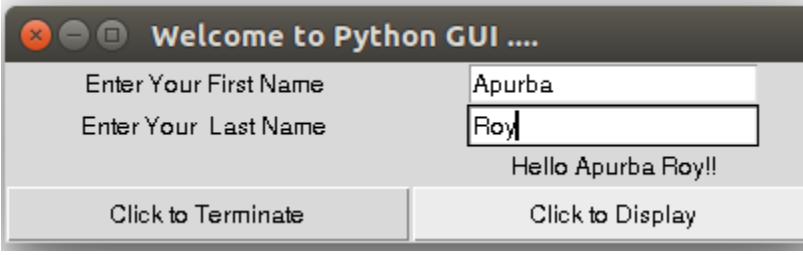


Image Handling

In order to do various operations and manipulations on images, we require the Python Pillow package. If the Pillow package is not present in the system then it can be installed using the below command.

```
pip install Pillow
```

```
from tkinter import *
from PIL import ImageTk,Image
root = Tk()
root.title("Welcome to Python GUI ....")
```

```
my_label1=Label(root,text='Click the ".. Load.." Button to Display the Image')
my_label1.grid(row=0,column=0)
```

```
def my_load():
```

```
    global my_img # If not global that will not be available in root window
```

```
    # loading the image
```

```
    my_img = ImageTk.PhotoImage(Image.open("./logo.jpg"))
```

```
    # Set on what image to be displayed
```

```
    my_label2 = Label(root,image = my_img)
```

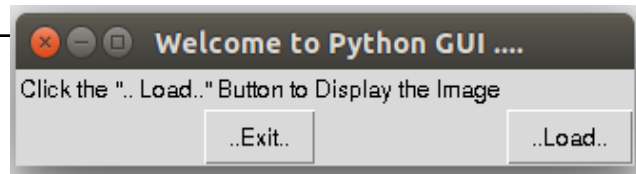
```
    # Place the container
```

```
    my_label2.grid(row=1,column=0)
```

```
my_button1=Button(root,text="..Load..",command=my_load)
my_button1.grid(row=2,column=1)
```

```
my_button2=Button(root,text="..Exit..",command=root.destroy)
my_button2.grid(row=2,column=0)
```

```
root.mainloop()
```



Frame:

```
from tkinter import *
from PIL import ImageTk, Image
root=Tk()
root.title("Welcome to CEMK!!")

#padx, pady provide space to place the frame in root window
# frame text can be ignored
my_frame=LabelFrame(root,text="This is1st Frame",padx=25,pady=25)
my_frame.pack(padx=20,pady=20)
my_frame2=LabelFrame(root,text="This is 2nd Frame",padx=15,pady=15)
my_frame2.pack(padx=40,pady=40)

my_label=Label(my_frame,text="Demonstration for placing a Frame")
my_label.pack()
my_button=Button(my_frame,text="Click to
Exit!!",command=root.destroy)
my_button.pack()

my_label2=Label(my_frame2,text="Demonstration for placing a Frame in
2")
my_label2.pack()
my_button2=Button(my_frame2,text="Click to
Exit!!",command=root.destroy)
my_button2.pack()

root.mainloop()
```



Radio Button: The Radiobutton is a standard Tkinter widget used to implement one-of-many selections. Radiobuttons can contain text or images, and you can associate a Python function or method with each button. When the button is pressed, Tkinter automatically calls that function or method.

Syntax: `button = Radiobutton(root, text="Name on Button", variable = "shared variable", value = "values of each button", options = values, ...)`

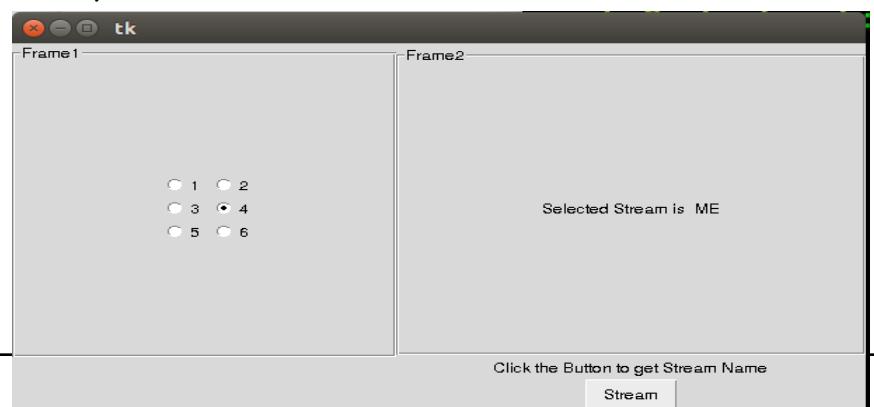
```
from tkinter import *
root=Tk()
root.title(" ")
frame1=LabelFrame(root,text="Frame1",padx=100,pady=100)
frame1.grid(row=0,column=0)
frame2=LabelFrame(root,text="Frame2",padx=100,pady=120)
frame2.grid(row=0,column=1)
r=StringVar()
r.set="6"
def myclick(value):
    mylabel2=Label(frame2,text="Selected Stream is" )
    mylabel2.grid(row=0,column=0)
    mylabel=Label(frame2,text=value)
    mylabel.grid(row=0,column=1)

r1=Radiobutton(frame1,text="1",variable=r,value="CSE")
Radiobutton(frame1,text="2",variable=r,value=" IT").grid(row=0,column=1)
Radiobutton(frame1,text="3",variable=r,value="ECE").grid(row=1,column=0)
Radiobutton(frame1,text="4",variable=r,value=" ME").grid(row=1,column=1)
Radiobutton(frame1,text="5",variable=r,value=" EE").grid(row=2,column=0)
Radiobutton(frame1,text="6",variable=r,value="AEIE").grid(row=2,column=1)
r1.grid(row=0,column=0)

mylabel2=Label(root,text="Click the Button to get Stream Name" )
mylabel2.grid(row=3,column=1)
my_button=Button(root,text="Stream",command=lambda:myclick(r.get()))
my_button.grid(row=4,column=1)

my_button1=Button(root,text="Exit",command=root.destroy)
my_button1.grid(row=4,column=0)

root.mainloop()
```



Checkbutton:

MessageBox:

```
from tkinter import *
from tkinter import messagebox
root=Tk()
root.title("Demonstration of popup() boxes ")

def mypopup():
# 6 kind of message box
# showinfo          showwarning          showerror
#askquestion        askokcancel          askyesno
    ans1=messagebox.showinfo("Welcome to GUI programming","This
is for message showing")
    Label(root,text=ans1).pack()
    if ans1 == 'ok':
        Label(root,text="You clicked OK in showinfo")
    ans2=messagebox.showwarning("Welcome to GUI
programming","This is showwarning display")
    Label(root,text=ans2).pack()
    if ans2=="ok":
        Label(root,text="You clicked OK in showWarning")
    ans3=messagebox.showerror("Welcome to GUI
programming","This is showerror display")
    Label(root,text=ans3).pack()
    if ans3=="ok":
        Label(root,text="You clicked OK in showerror")
    ans4=messagebox.askquestion("Welcome to GUI
programming","This is askquestion acceptance")
    #Label(root,text=ans4).pack()
    if ans4==1:
        Label(root,text="You Clicked YES within
(ASKQuestion)").pack()
    else:
        Label(root,text="You Clicked NO within
(ASKQuestion)").pack()
    ans5=messagebox.askokcancel("Welcome to GUI
```

```
programming","This is OK/Cancel acceptance")
    #Label(root,text=ans5).pack()
    if ans5==1:
        Label(root,text="You Clicked OK within
(OK/Cancel)").pack()
    else:
        Label(root,text="You Clicked Cancel within
(OK/Cancel)").pack()

    ans6=messagebox.askyesno("Welcome to GUI programming","This
is Yes/No acceptance")
    #Label(root,text=ans6).pack()
    if ans6==1:
        Label(root,text="You Clicked YES within
(YES/NO)").pack()
    else:
        Label(root,text="You Clicked No within
(YES/NO)").pack()

button1=Button(text="Popup",command=mypopup).pack()
button2=Button(text="Exit",command=root.destroy).pack()

root.mainloop()
```