PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Prof. Sowmya Shree P, Prof. Sindhu R Pai
Department of Computer Science and Engineering

PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Unit – 3: Graphical User Interface with Tkinter package

Prof. Sowmya Shree P, Prof. Sindhu R Pai

Department of Computer Science and Engineering

GUI - Tkinter

Why do we need GUI?

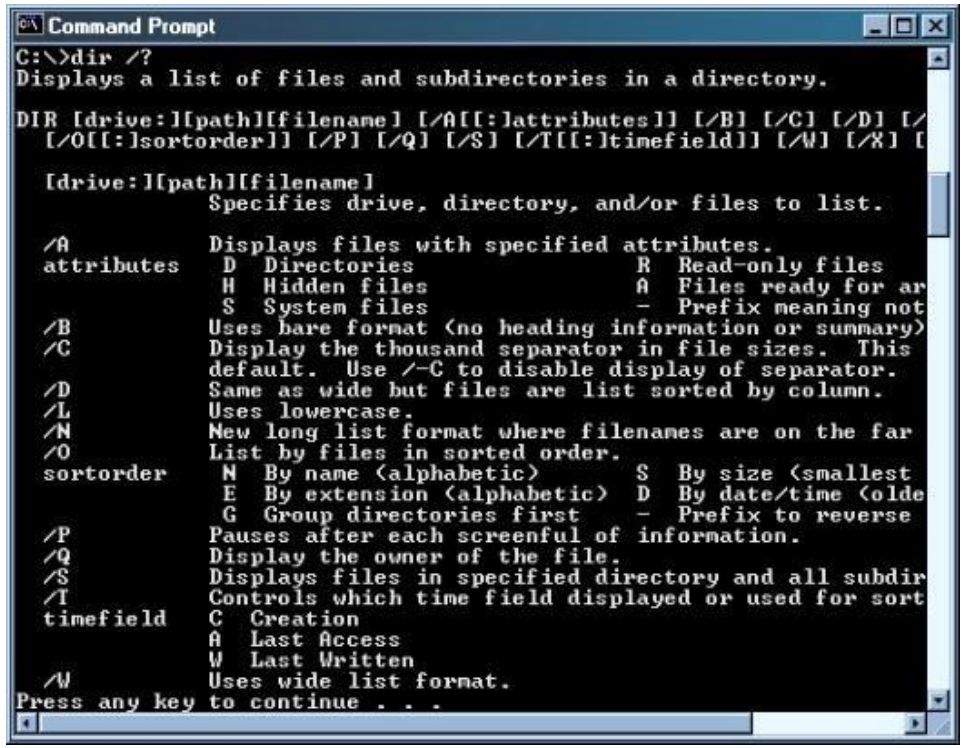
- A user with no computer knowledge can literally start learning about the machine because of GUI as it provides scope for users to explore and provides discoverability.

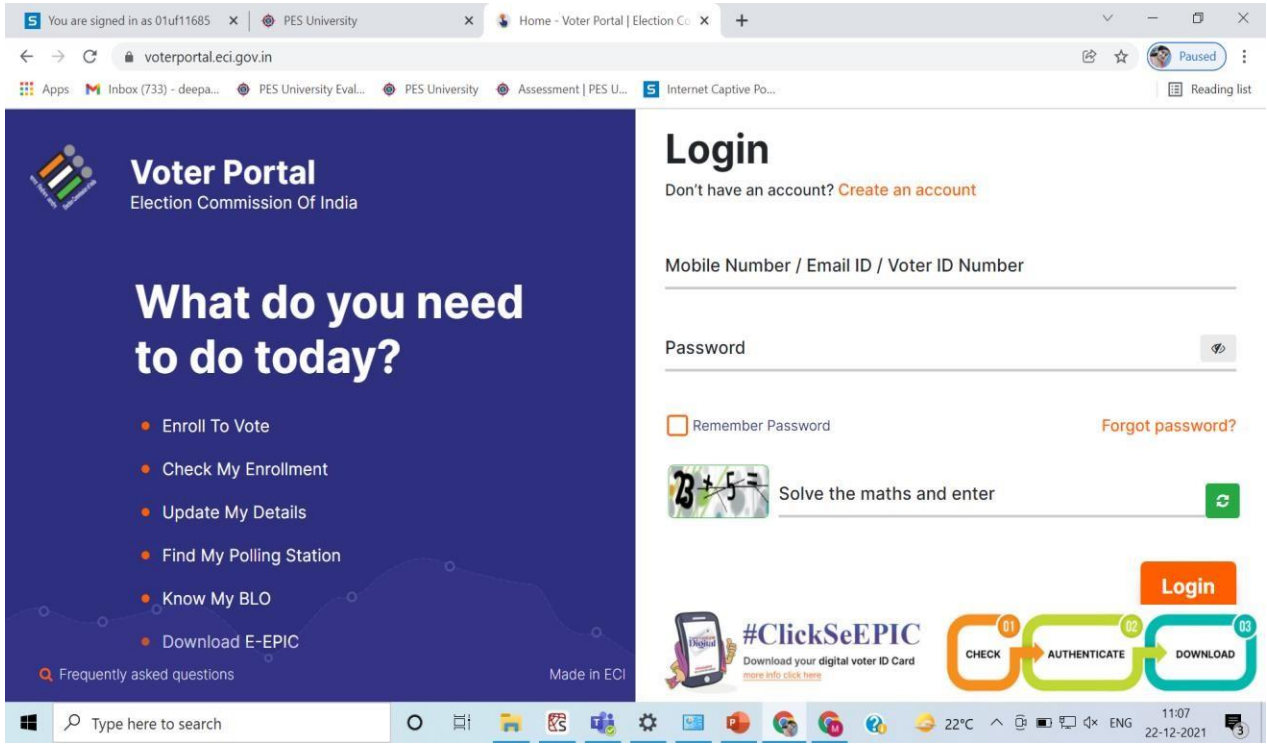
- For example, a user starts using a computer with no Interface, then he/she has to provide commands to the machine to execute each task. In a way, the user must have some kind of programming knowledge.

- In real time, you consider the system used in Retail store for billing purpose for command line interface and voting portal for GUI interface.

GUI - Tkinter

Observe the below screenshot, to get clear picture of Command Line Interface and Graphical User Interface.





Popular Python GUI frameworks

1. Tkinter
2. Qt for Python: PySide2 / Qt5
3. PySimpleGUI
4. PyGUI
5. Kivy
6. wxPython
7. Libavg
8. PyForms
9. Wax
10. PyGTK

Tkinter

- Built into the Python standard library
- It’s cross-platform, so the same code works on Windows, macOS, and Linux
- Lightweight and relatively easy to use compared to other frameworks

GUI - Tkinter

Tkinter is used

1. To create Windows and Dialog boxes
2. To build a GUI for Desktop Applications
3. To add a GUI to Command-Line Program
4. To create custom Widgets
5. In Prototyping a GUI

GUI - Tkinter

Let us learn how to create a GUI application in Python using Tkinter.

Import Tkinter package:

```
import tkinter
```

GUI - Tkinter

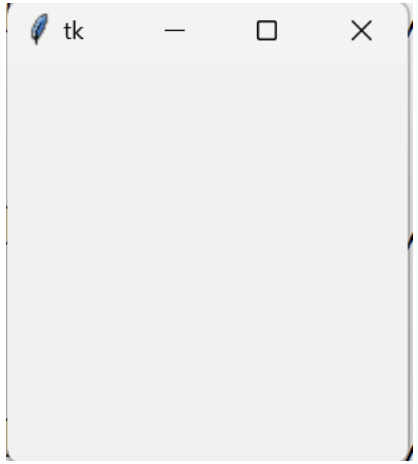
To create a Window

Step 1: Import tkinter package

Step 2: root=tkinter.Tk()

Step 3: root.mainloop()

Output




```
import tkinter
root = tkinter.Tk()        #creates window
root.mainloop()            #loops continuously until we close the  window
```

mainloop()

- A function that continuously loops and displays the window till we close it or an action closes the window.
- It will loop forever, waiting for events from the user, until the user exits the program (either by closing the window, or by terminating the program with a keyboard interrupt in the console)
- All windows that are created, work on this concept of constant looping to keep track of the interactions of the user with the Interface.
- It can track the movements of the mouse on the window because it constantly loops and has knowledge of where the mouse pointer is on the window at every frame.

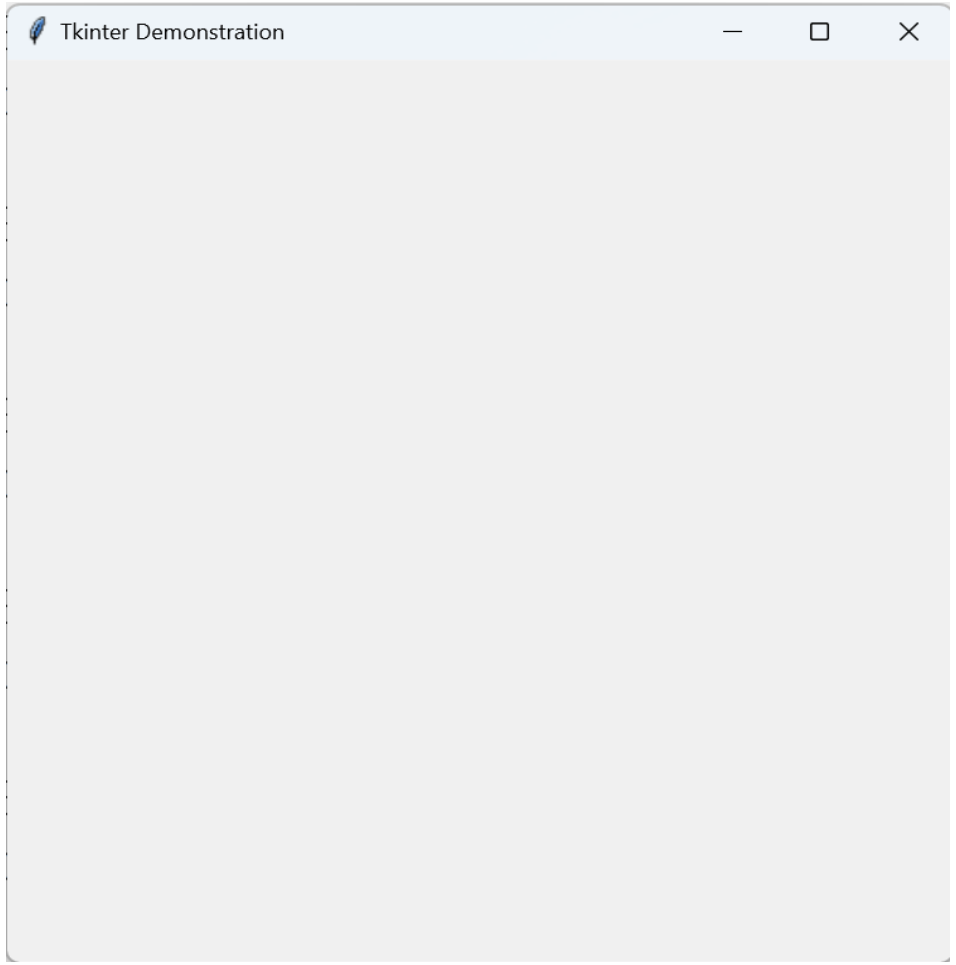
Adding title and geometry to the Window

root.title(Title Name)
root.geometry(Dimension in widthxheight)

Example:

```
import tkinter
root = tkinter.Tk()   #creates window
root.title("Tkinter Demonstration")   #Title
root.geometry('500x500’)  #Dimension
root.mainloop()
```

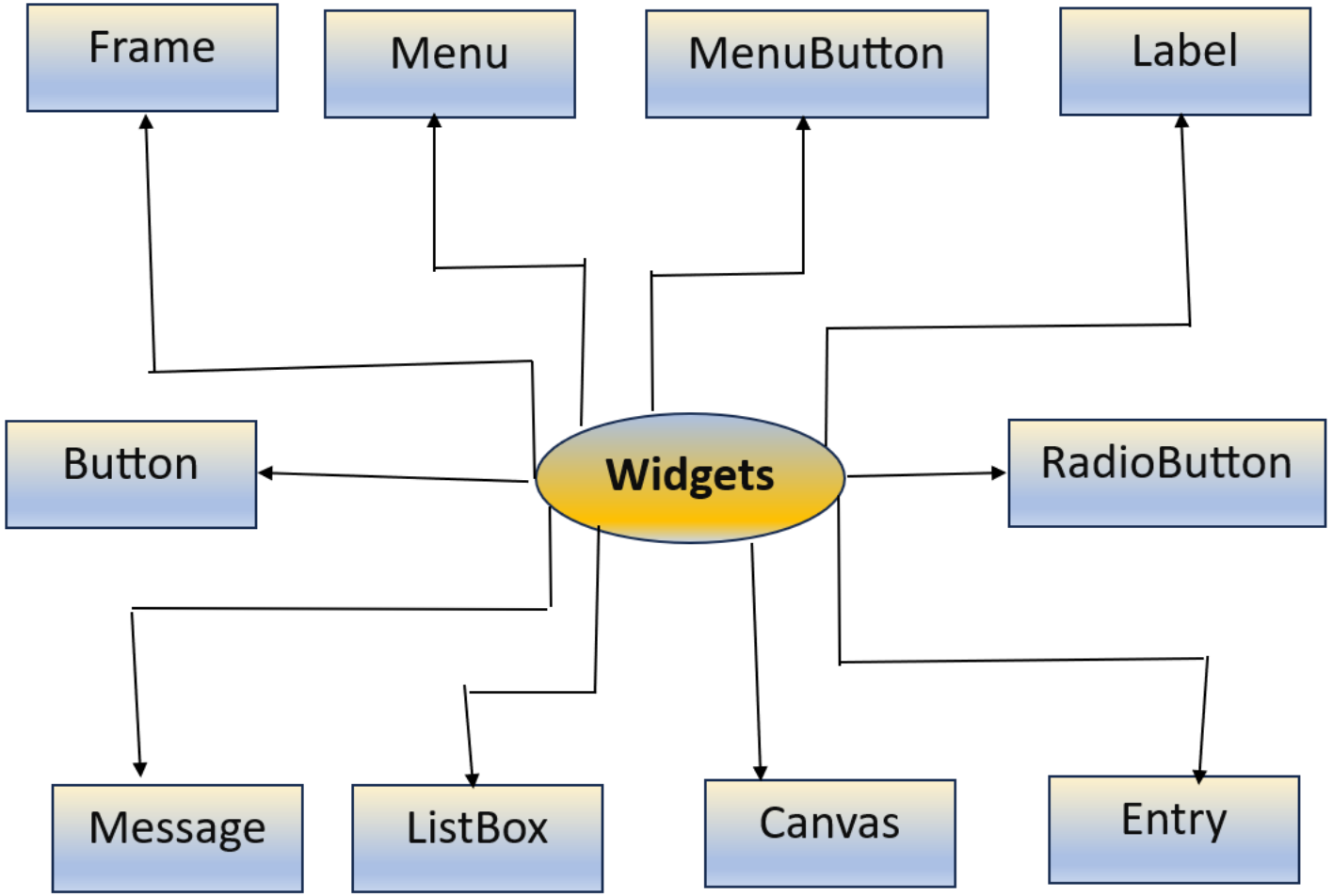
Output



Widgets

- After creating window, we need to add elements to make it more interactive.
- Each element in Tkinter is Widget.
- In Tkinter , Widgets are objects.
- Each separate widget is a Python object.
- When creating a widget, we must pass its parent as a parameter to the widget creation function.
- Except “root” window, which is the top-level window that will contain everything else and it does not have a parent.

GUI - Tkinter


Frame
Menu
MenuButton
Label
Button
Widgets
RadioButton
Message
ListBox
Canvas
Entry

GUI - Tkinter

Widget Name	Description
Button	To add a button to the application
Canvas	To draw a complex layout and pictures (like graphics, text, etc.)
CheckButton	To display a number of options as checkboxes
Entry	To display a single-line text field that accepts values from the user
Frame	To group and organize other widgets
Label	To Provide a single-line caption, can contain images also.
Listbox	To provide a user with a list of options
Menu	Creates all kinds of Menus required in the application
Menubutton	To display the menu items to the user

GUI - Tkinter

Widget Name	Description
Message	Displays a message box to the user
Radiobutton	Number of options to be displayed as radio buttons
Scale	A graphical slider that allows to select values from the scale
Scrollbar	To scroll the window up and down
Text	A multi-line text field to the user where users enter or edit the text and it is different from Entry
Toplevel	Used to provide a separate window container
Spinbox	An entry to the "Entry widget" in which value can be input just by selecting a fixed value of numbers
PanedWindow	A container widget that is mainly used to handle different panes
MessageBox	Used to display messages in desktop applications

Widgets

- Steps to add widget to the Window
  1. Create widget
  2. Add it to the Window

- Creating a new widget doesn't mean that it will appear on the screen. To display it, we need to call a special method: either **grid, pack**, or **place**.

1. **pack()** - packs widgets in rows or columns
2. **grid()** - puts the widgets in a 2-dimensional table. The master widget is split into a number of rows and columns, and each "cell" in the resulting table can hold a widget.
3. **place()** - explicitly set the position and size of a window, either in absolute terms, or relative to another window.

Button Widget - To add a button to the application

- **Syntax**
  w=Button(parent,options)
- parent – parent window
- options – to change look of the buttons, written as comma-separated

Button widget options

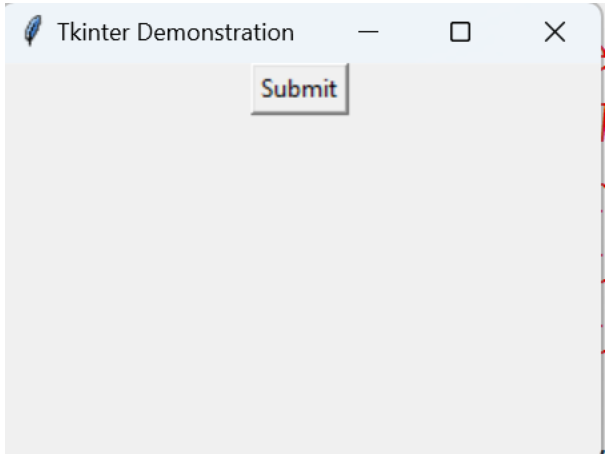
activebackground – background of button when the mouse hovers the button
activeforeground – represents the font color when the mouse hovers the button
bd – width of the border
bg – background color of button
fg – foreground colot of button
height – height of button
justify – with 3 values, LEFT, RIGHT, CENTER
underline – underline the text of button
width – width of the button

Button Widget

Example1

```
from tkinter import *
win =Tk()
win.title("Tkinter Demonstration")
win.geometry('300x200')
b=Button(win, text='Submit')
b.pack()
win.mainloop()
```

Output



Button Widget

Example2

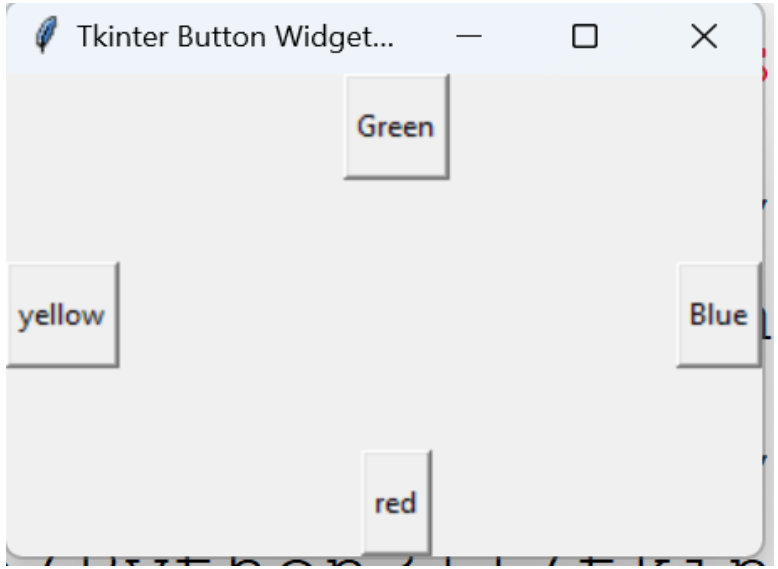
```
import tkinter
from tkinter import *
from tkinter import messagebox

win = Tk()
win.title("Tkinter Button Widget Demonstration")
win.geometry('300x200')

def click():
    messagebox.showinfo("Message", "Green Button clicked")

a=Button(win, text="yellow", activeforeground="yellow", activebackground="orange", pady=10)
```

Button Widget
Example2 (Contd…)

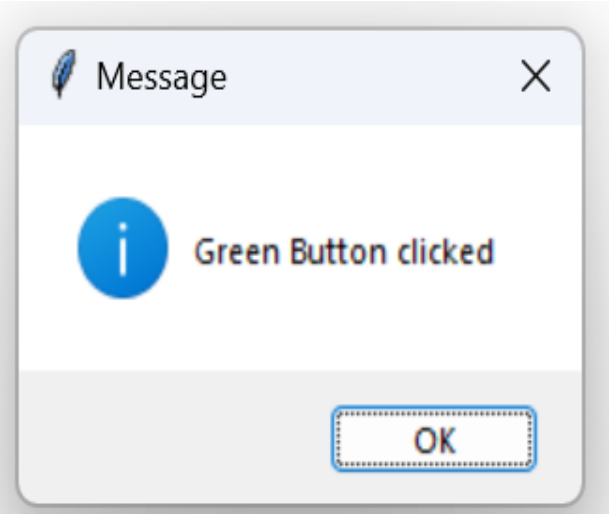
```
b=Button(win, text="Blue", activeforeground="blue", activebackground="orange", pady=10)
# adding click function to the below button
c=Button(win, text="Green", command=click, activeforeground = "green",
activebackground="orange", pady=10)
d=Button(win, text="red", activeforeground="red", activebackground="orange", pady=10)
a.pack(side=LEFT)
b.pack(side=RIGHT)
c.pack(side=TOP)
d.pack(side=BOTTOM)
win.mainloop()
```

Button Widget
Example2 (Contd…)

Output



After clicking Green button, Messagebox appears.



GUI - Tkinter

Canvas Widget - used to draw anything on the application window

- **Syntax**
  w=Canvas(parent,option=value)
- parent – parent window
- option – to change layout of the canvas, written as comma-separated-Key-values.

Canvas widget options

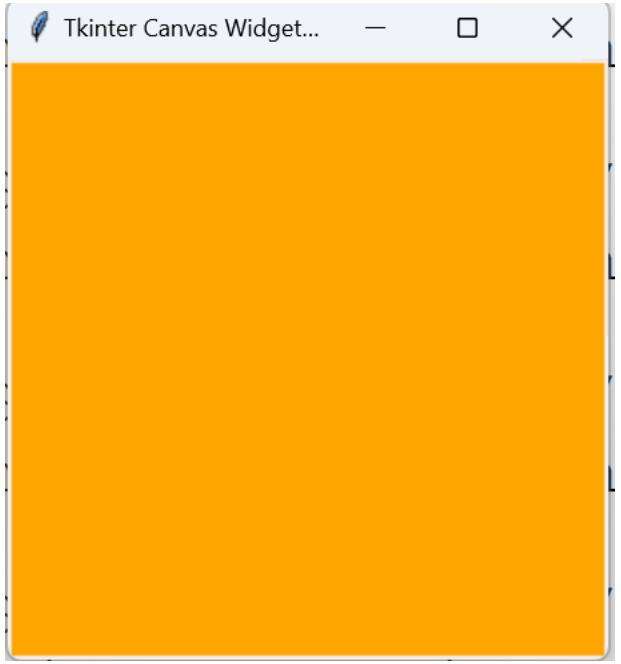
bd – width of the border
bg – background color
cursor – to use arrow, dot, or circle
height – height of canvas
xscrollcommand – horizontal scrollbar
yscrollcommand – vertical scrollbar
confine – non-scrollable outside the scroll region

Canvas Widget

Example1

```
from tkinter import *

win=Tk()
win.title("Tkinter Canvas Widget Demonstration")
win.geometry("300x300")

#creating canvas
cv=Canvas(win, bg = "orange", height = "300")

cv.pack()

win.mainloop()
```

Output



Canvas Widget

Example2

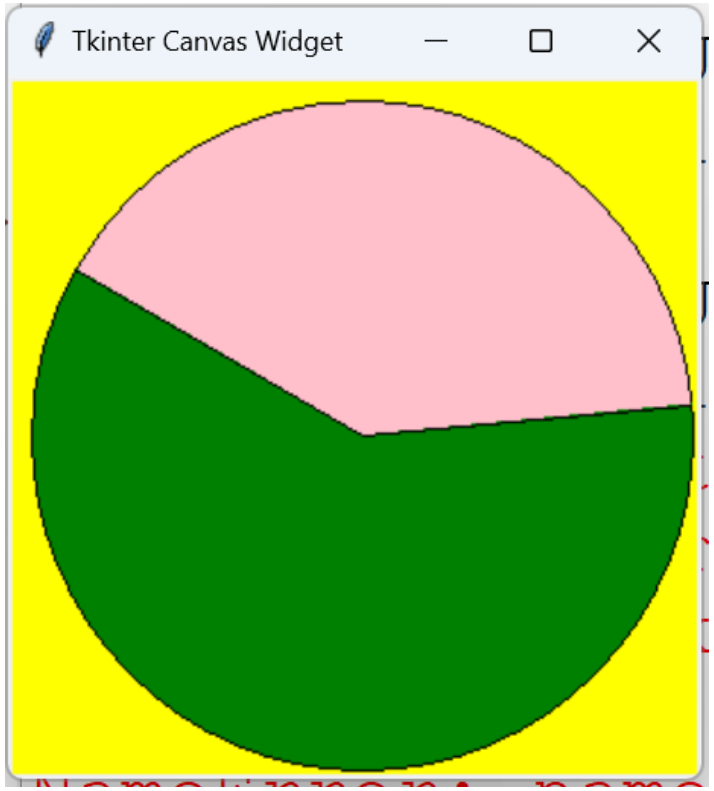
```
import tkinter

win=tkinter.Tk()
win.title("Tkinter Canvas Widget")

# creating canvas
cv=tkinter.Canvas(win, bg="yellow", height=300, width=300)

# drawing two arcs
coord = 10, 10, 300, 300
arc1=cv.create_arc(coord, start=0, extent=150, fill="pink")
arc2=cv.create_arc(coord, start=150, extent=215, fill="green")
```

Canvas Widget

Example2 (Contd…)

```
# adding canvas to window and display it
cv.pack()
win.mainloop()
```

Output



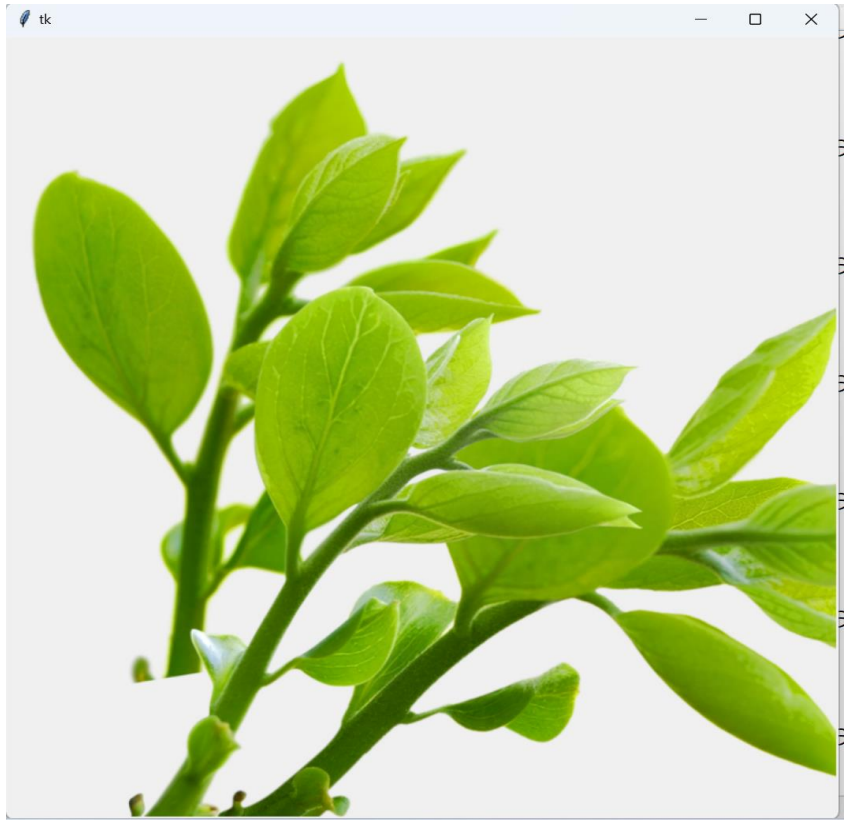
Canvas Widget

Example3

```
from tkinter import *

win=Tk()

cv=Canvas(win, height=700, width=700)
filename=PhotoImage(file="nature.png")

image=cv.create_image(20, 20, anchor=NW, image=filename)

cv.pack()
win.mainloop()
```

Canvas Widget

Example3 - Output



Checkbutton Widget - button to select from multiple options

- **Syntax**
  w= Checkbutton(parent,option=value)
- parent – parent window
- option – to configure checkbutton, written as comma-separated-Key-value pair.

Checkbutton widget options

bd – width of the border
bg – background color of button
bitmap – to display image in the button
command – function to be called on checking the button
height – height of widget
image – display generic image on the button
justify – with 3 values, LEFT, RIGHT, CENTER
padx – space to leave to the left and right of the checkbutton and text. Default value is 1 pixel
pady – space to leave to the above and below the checkbutton and text. Default value is 1 pixel

Checkbutton Widget

- **Functions**

1. deselect(): to turn off the checkbutton
2. flash(): The checkbutton is flashed between the active and normal colors.
3. invoke(): invoke the method associated with the checkbutton.
4. select(): to turn on the checkbutton.
5. toggle(): to toggle between the different Checkbuttons.

Checkbutton Widget

Example

```
from tkinter import *

win=Tk()
win.geometry("300x300")

w=Label(win, text ='Select Your Hobbies:', fg="Blue",font = "100")
w.pack()

Checkbutton1 = IntVar() # holds integer data passed to the checkbutton widget
Checkbutton2 = IntVar()
Checkbutton3 = IntVar()
```

Checkbutton Widget
Example (Contd…)

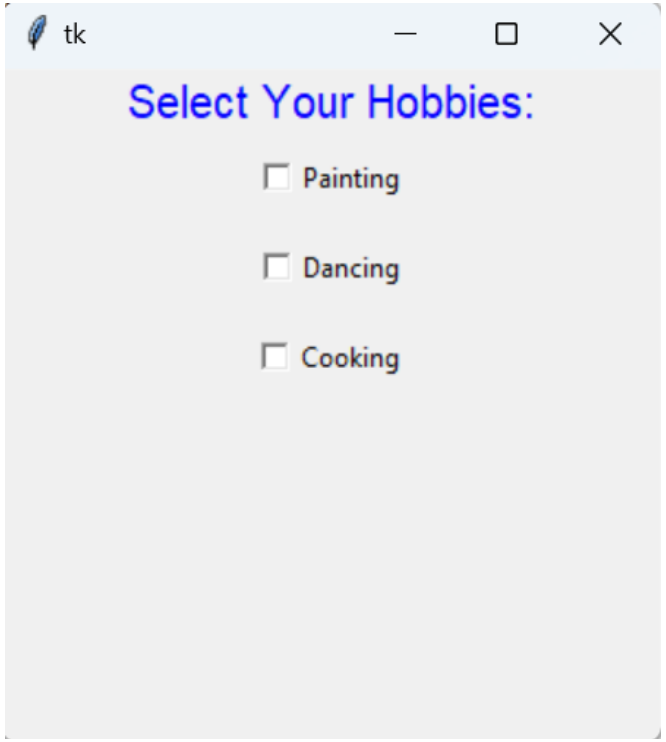
```
cb1=Checkbutton(win, text="Painting",variable = Checkbutton1,
                onvalue = 1,
                offvalue = 0,
                height = 2,
                width = 10)

cb2=Checkbutton(win, text = "Dancing", variable = Checkbutton2,
                onvalue = 1,
                offvalue = 0,
                height = 2,
                width = 10)
```

Checkbutton Widget
Example (Contd…)

```
cb3=Checkbutton(win, text = "Cooking", variable = Checkbutton3,
                                        onvalue = 1,
                                        offvalue = 0,
                                        height = 2,
                                        width = 10)

cb1.pack()
cb2.pack()
cb3.pack()

mainloop()
```

Checkbutton Widget
Example (Contd…)

Output



Label Widget - to provide a message about the other widgets

- **Syntax**
  w= Label(parent,options)
- parent – parent window
- option – to configure the text, written as comma-separated-Key-value pair.

Label widget options

anchor – to control the position of widget
bg – background color of widget
bitmap – to set the bitmap equals to the graphical object
cursor – type of cursor to show when the mouse is moved over the label
height – height of widget
image – indicates the image that is shown as label
justify – with 3 values, LEFT, RIGHT, CENTER
padx – Horizontal padding of text. Default value is 1.
pady – Vertical padding of text. Default value is 1.

Label Widget

Example

```
from tkinter import *

win=Tk()

win.geometry("400x250")


username=Label(win, text = "Username").place(x = 30,y = 50)

password=Label(win, text = "Password").place(x = 30, y = 90)
```

Label Widget

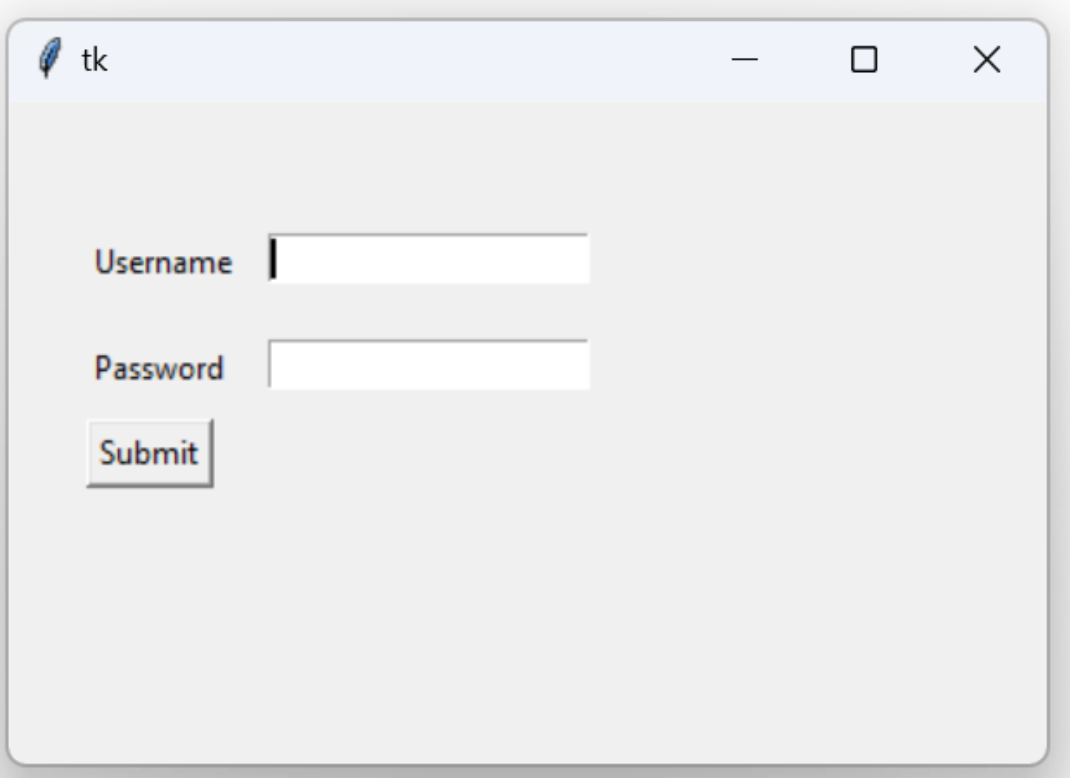
Example

```
submitbutton=Button(win, text = "Submit",activebackground = "red", activeforeground =
"blue").place(x = 30, y = 120)

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

win.mainloop()
```

Label Widget

Example

Output



Entry Widget - to enter or display single line of text

- **Syntax**
  w= Entry(parent,options)
- parent – parent window
- option – to configure the entry, written as comma-separated values.

Entry widget options

bg – background color of widget

font – font used for the text

fg – color to render the text

relief – default value, relief=FLAT. Other styles are : SUNKEN, RIGID, RAISED, GROOVE

show –to show the text while making an entry, Eg: for Password set Show="*"

textvariable – to retrieve the current text from your entry widget

Entry Widget

- **Functions**

1. get(): Returns the entry's current text as a string
2. delete(): Deletes characters from the widget
3. insert(index,name): Inserts string 'name' before the character at the given index

Entry Widget

Example

```
from tkinter import *

win=Tk()

win.geometry("400x250")

name=Label(win, text = "Name").place(x = 30,y = 50)
email=Label(win, text = "Email").place(x = 30, y = 90)
password=Label(win, text = "Password").place(x = 30, y = 130)
```

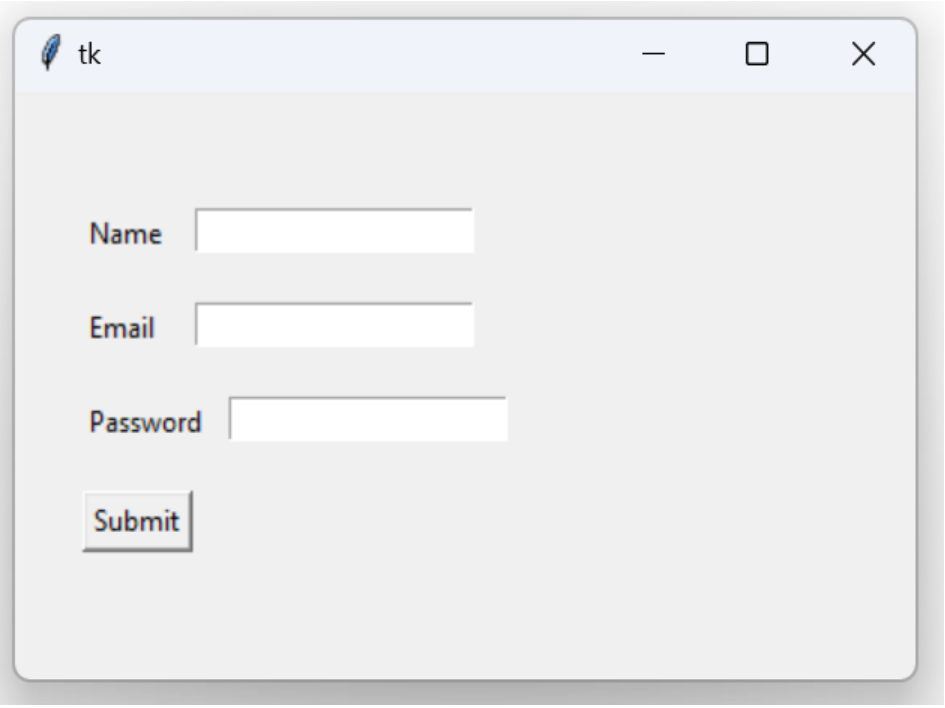
Entry Widget

Example (Contd…)

```
submitbtn=Button(win, text = "Submit",activebackground = "red", activeforeground =
"blue").place(x = 30, y = 170)

entry1=Entry(win).place(x = 80, y = 50)
entry2=Entry(win).place(x = 80, y = 90)
entry3=Entry(win).place(x = 95, y = 130)

win.mainloop()
```

Entry Widget
Example (Contd…)

Output



Dialogs in Tkinter

- A window which is used to "talk" to the application
- Used to input data, modify data, change the application settings etc.
- Communication between a user and a computer program

Tkinter Message Box Dialog

- Provide messages to the user of the application
- Message consists of text and image data
- Located in tkMessagebox module
- By using the message box library several Information is displayed, such as Error, Warning, Cancellation etc.

Message Box

- **Syntax**

  messagebox.function_name(Title, Message, [,options])

- function_name – Name of the function we want to use
- Title – Message box's Title
- Message – Message to be shown on the dialog
- options – to Configure the options

function_name

Name of the function	Significance
showinfo()	To display some important information
showwarning()	To display some type of Warning
showerror()	To display some Error Message
askquestion()	To display a dialog box that asks with two options YES or NO
askokcancel()	To display a dialog box that asks with two options OK or CANCEL
askretrycancel()	To display a dialog box that asks with two options RETRY or CANCEL
askyesnocancel()	To display a dialog box that asks with three options YES or NO or CANCEL

Messagebox – askquestion()

Example1

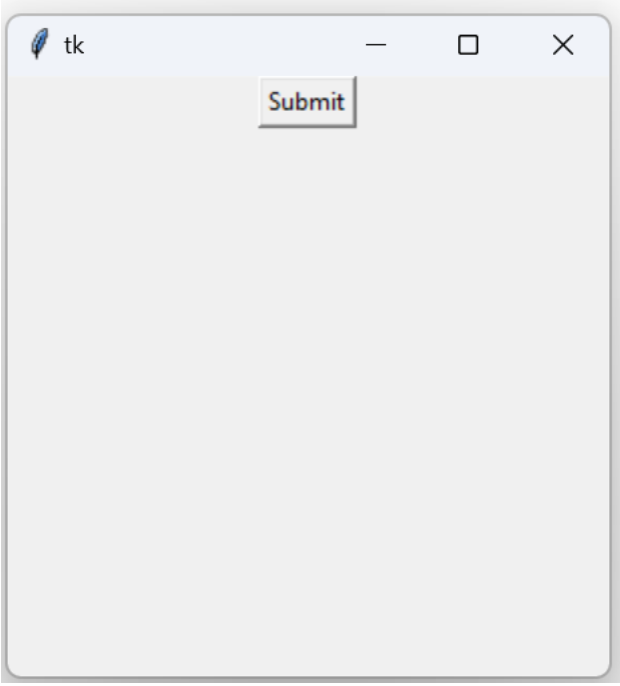
```
from tkinter import *
from tkinter import messagebox

win=Tk()

# function to use the askquestion() function
def Submit():
    messagebox.askquestion("Form", "Do you want to Submit")

win.geometry("300x300")
```

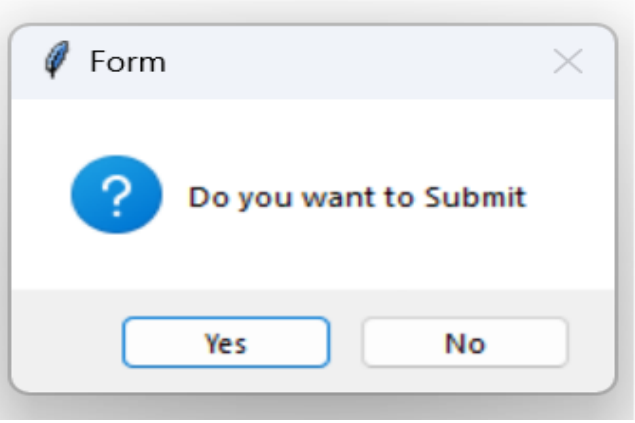
Example1 (Contd…)

```
# creating Submit Button
b=Button(win, text = "Submit", command = Submit)
b.pack()

win.mainloop()
```

Example1 (Contd…)

Output





After clicking Submit button in the 1st window, message box is displayed.

Frame widget in Tkinter

- A frame - rectangular region on the screen.
- Used to implement complex widgets.
- Organize a group of widgets.
- **Syntax**

  w=frame(parent,options)
- parent – parent window
- options – to configure frames, written as comma-separated-Key-value pair.

Frame widget options

bg – background color displayed behind the label and indicator
bd – border size, default is 2 pixels
cursor – to change the mouse cursor pattern
height – vertical dimension of new frame
highlightcolor – color of focus highlight when the frame has focus
highlightthickness – color the focus when the frame does not have the focus
highlightbackground – thickness of focus highlight
relief – type of the border of the frame. default =FLAT
width – width of the frame

Frame widget

Example1

```
from tkinter import *

win = Tk()
win.geometry("300x150")

w=Label(win, text ='Frame Demonstration', font = "50")
w.pack()

frame=Frame(win)
frame.pack()
```

Example1 (Contd…)

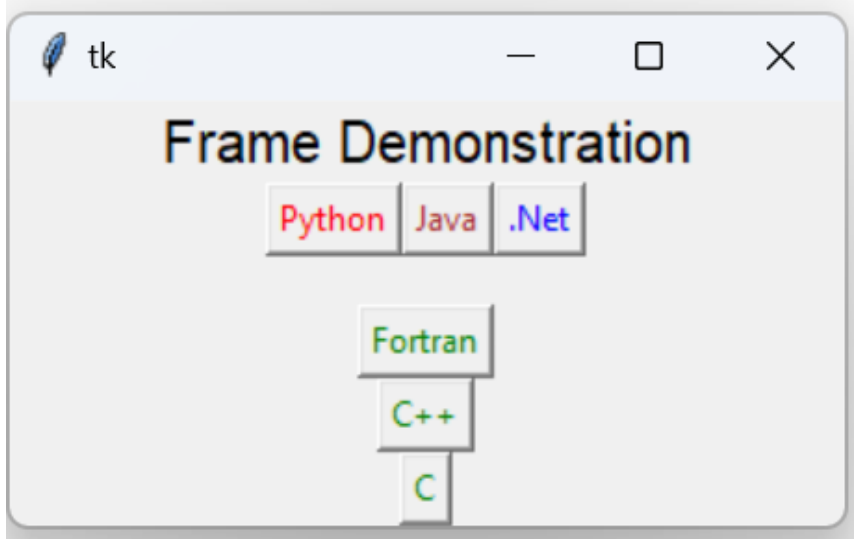
```
bottomframe=Frame(win)
bottomframe.pack( side = BOTTOM )

b1= Button(frame, text ="Python", fg ="red")
b1.pack( side = LEFT)

b2 = Button(frame, text ="Java", fg ="brown")
b2.pack( side = LEFT )

b3 = Button(frame, text =".Net", fg ="blue")
b3.pack( side = LEFT )
```

Example1 (Contd…)

```
b4 = Button(bottomframe, text ="C", fg ="green")
b4.pack( side = BOTTOM)

b5 = Button(bottomframe, text ="C++", fg ="green")
b5.pack( side = BOTTOM)

b6 = Button(bottomframe, text ="Fortran", fg ="green")
b6.pack( side = BOTTOM)

win.mainloop()
```

Example1 (Contd…)

Output



Frame widget –Nested Frames

- A frame within another frame

- Steps to create Nested Frames
  1. Create normal Tkinter window
  2. Create 1st Frame
  3. Create 2nd Frame
  4. Take the 1st frame as parent for 2nd Frame
  5. Execute code

- Syntax

```
frame(parent)
```

Frame widget – Nested Frames

Example 4

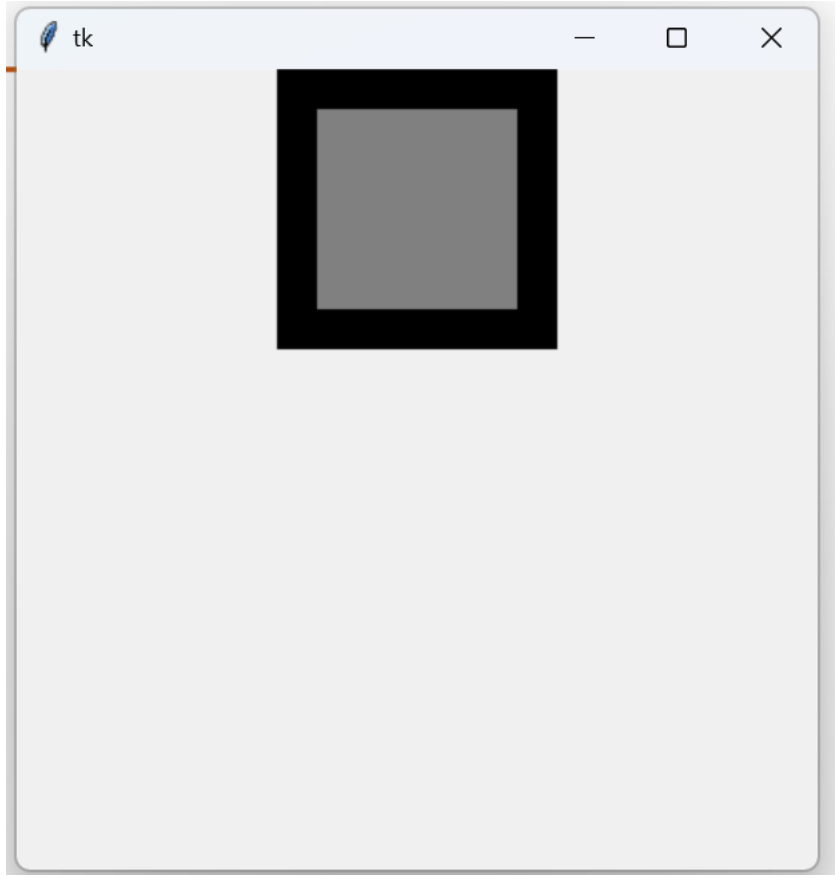
```
from tkinter import *

win=Tk()
win.geometry("400x400")

# Frame 1
frame1=Frame(win,bg="black",width=500,height=300)
frame1.pack()
```

Example 4 (Contd…)

```
# Frame 2 is created within Frame 1
frame2=Frame(frame1,bg="Grey",width=100,height=100)
frame2.pack(pady=20,padx=20)

win.mainloop()
```

Example 4 (Contd…)

Output



GUI - Tkinter

Explore on:

- Tkinter Color Chooser Dialog - colorchooser – askcolor()
- Tkinter file dialog - filedialog – askopenfile()
- Frame widget – Change width
- Frame widget – Change Color

THANK YOU

Department of Computer Science and Engineering

Contact Email ID's:

sindhurpai@pes.edu

sowmyashree@pes.edu