Experiment 09

9.1) Write python programs to understand designing Graphical user interface (GUI) using built-in tools in python (Tkinter)
9.2) Write a python program to implement GUI Canvas Application using Tkinter

Roll No.	01
Name	Aamir Ansari
Class	D10-A
Subject	Python Lab
LO Mapped	LO1: Understand the structure, syntax, and semantics of the Python language LO5: Gain proficiency in creating GUI applications

Aim:

9.1) Write python programs to understand designing Graphical user interface (GUI) using built-in tools in python (Tkinter)
9.2) Write a python program to implement GUI Canvas Application using Tkinter.

Introduction:

GUI Programming in Python

Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

1. Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter
2. wxPython − This is an open-source Python interface for wxWindows
3. JPython − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine

Out of all the GUI methods, tkinter is the most commonly used method.

Tkinter Programming

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

1. Import the Tkinter module.
2. Create the GUI application main window.
3. Add one or more of the above-mentioned widgets to the GUI application.
4. Enter the main event loop to take action against each event triggered by the user.

Importing tkinter is same as importing any other module in the Python code. Note that the name of the module in Python 2.x is ‘Tkinter’ and in Python 3.x it is ‘tkinter’.

```
import tkinter
```

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

1. 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

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()

Example:

```
import tkinter
m = tkinter.Tk()
# widgets are added here
m.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 class.

1. pack() method:It organizes the widgets in blocks before placing in the parent widget.
2. grid() method:It organizes the widgets in grid (table-like structure) before placing in the parent widget.
3. place() method:It organizes the widgets by placing them on specific positions directed by the programmer.

Tkinter Widgets

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

Button

To add a button in your application, this widget is used.
The general syntax is:

w=Button(master, option=value)

where, master is the parameter used to represent the parent window.
There are 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.

1. activebackground: to set the background color when button is under the cursor.
2. activeforeground: to set the foreground color when button is under the cursor.
3. bg: to set the normal background color.
4. command: to call a function.
5. font: to set the font on the button label.

6. image: to set the image on the button.
7. width: to set the width of the button.
8. height: to set the height of the button.

Example:

```
import tkinter as tk
r = tk.Tk()
r.title('Hello')
button=tk.Button(r, text='Click me to close window!', width=25, height=5, command=r.destroy)
button.pack()
r.mainloop()
```

CheckButton

To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is:

w = CheckButton(master, option=value)

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

1. title: To set the title of the widget.
2. activebackground: to set the background color when widget is under the cursor.
3. activeforeground: to set the foreground color when widget is under the cursor.
4. bg: to set he normal backgrouSteganography
5. command: to call a function.
6. font: to set the font on the button label.
7. image: to set the image on the widget.

Example:

```
from tkinter import *
master = Tk()
var1 = IntVar()
c1 = Checkbutton(master, text='Music', variable=var1)
var2 = IntVar()
c2 = Checkbutton(master, text='Video', variable=var2)
c1.pack()
c2.pack()
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 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.

1. bd: to set the border width in pixels.
2. bg: to set the normal background color.
3. cursor: to set the cursor used.
4. command: to call a function.
5. highlightcolor: to set the color shown in the focus highlight.
6. width: to set the width of the box.
7. height: to set the height of the box.

Example:

```
from tkinter import *
master = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```

Frame

It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

w = Frame(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. highlightcolor: To set the color of the focus highlight when widget has to be focused.
2. bd: to set the border width in pixels.
3. bg: to set the normal background color.

4. cursor: to set the cursor used.
5. width: to set the width of the widget.
6. height: to set the height of the widget.

Example:

```
from tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text = 'Red', fg ='red')
redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue', fg ='blue')
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
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 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.

1. bg: to set he normal background color.
2. bg to set he normal background color.
3. command: to call a function.
4. font: to set the font on the label.
5. image: to set the image on the widget.
6. width: to set the width of the widget.
7. height" to set the height of the widget.

Example:

```
from tkinter import *
```

```
root = Tk()
w = Label(root, text="THIS IS A LABEL", fg="#06a099", font=("Arial", 15, 'bold'))
w.pack()
root.mainloop()
```

Listbox

It offers a list to the user from which the user can accept any number of options.
The general syntax is:

w = Listbox(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. highlightcolor: To set the color of the focus highlight when widget has to be focused.
2. bg: to set he normal background color.
3. bd: to set the border width in pixels.
4. font: to set the font on the button label.
5. image: to set the image on the widget.
6. width: to set the width of the widget.
7. height: to set the height of the widget.

Example:

```
from tkinter import *
top = Tk()
Lb = Listbox(top)
Lb.insert(1, 'Python')
Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4, 'Any other')
Lb.pack()
top.mainloop()
```

MenuButton**:**

It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality. The general syntax is:

w = MenuButton(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. activebackground: To set the background when mouse is over the widget.
2. activeforeground: To set the foreground when mouse is over the widget.
3. bg: to set he normal background color.
4. bd: to set the size of border around the indicator.
5. cursor: To appear the cursor when the mouse over the menubutton.
6. image: to set the image on the widget.
7. width: to set the width of the widget.
8. height: to set the height of the widget.
9. highlightcolor: To set the color of the focus highlight when widget has to be focused.

Example:

```
from tkinter import *
top = Tk()
mb =  Menubutton ( top, text = "Menu")
mb.grid()
mb.menu  =  Menu ( mb, tearoff = 0 )
mb["menu"]  =  mb.menu
cVar  = IntVar()
aVar = IntVar()
mb.menu.add_checkbutton ( label = 'Contact', variable = cVar )
mb.menu.add_checkbutton ( label = 'About', variable = aVar )
mb.pack()
top.mainloop()
```

Menu

It is used to create all kinds of menus used by the application.
The general syntax is:

w = Menu(master, option=value)

master is the parameter used to represent the parent window.

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

1. title: To set the title of the widget.
2. activebackground: to set the background color when widget is under the cursor.
3. activeforeground: to set the foreground color when widget is under the cursor.
4. bg: to set he normal background color.
5. command: to call a function.

6. font: to set the font on the button label.
7. image: to set the image on the widget.

Example:

```
from tkinter import *
root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About')
mainloop()
```

Message

It refers to the multi-line and non-editable text. It works same as that of Label.
The general syntax is:

w = Message(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. bd: to set the border around the indicator.
2. bg: to set he normal background color.
3. font: to set the font on the button label.
4. image: to set the image on the widget.
5. width: to set the width of the widget.
6. height: to set the height of the widget.

Example:

```
from tkinter import *
main = Tk()
ourMessage ='Your transaction was a success!'
messageVar = Message(main, text = ourMessage)
messageVar.config(bg='lightgreen')
```

```
messageVar.pack( )
main.mainloop( )
```

RadioButton

It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.
The general syntax is:

w = RadioButton(master, option=value)

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

1. activebackground: to set the background color when widget is under the cursor.
2. activeforeground: to set the foreground color when widget is under the cursor.
3. bg: to set he normal background color.
4. command: to call a function.
5. font: to set the font on the button label.
6. image: to set the image on the widget.
7. width: to set the width of the label in characters.
8. height: to set the height of the label in characters.

Example:

```
from tkinter import *
def sel():
   selection = "You selected the option " + str(var.get())
   label.config(text = selection)
root = Tk()
var = IntVar()
R1 = Radiobutton(root, text="Option 1", variable=var, value=1,command=sel)
R1.pack( anchor = W )
R2 = Radiobutton(root, text="Option 2", variable=var, value=2,command=sel)
R2.pack( anchor = W )
R3 = Radiobutton(root, text="Option 3", variable=var, value=3,command=sel)
R3.pack( anchor = W)
label = Label(root)
label.pack()
root.mainloop()
```

Scale

It is used to provide a graphical slider that allows to select any value from that scale. The general syntax is:

w = Scale(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. cursor: To change the cursor pattern when the mouse is over the widget.
2. activebackground: To set the background of the widget when mouse is over the widget.
3. bg: to set he normal background color.
4. orient: Set it to HORIZONTAL or VERTICAL according to the requirement.
5. from_: To set the value of one end of the scale range.
6. to: To set the value of the other end of the scale range.
7. image: to set the image on the widget.
8. width: to set the width of the widget.

Example:

```
from tkinter import *
master = Tk()
w = Scale(master, from_=0, to=20)
w.pack()
w = Scale(master, from_=0, to=100, orient=HORIZONTAL)
w.pack()
mainloop()
```

Scrollbar

It refers to the slide controller which will be used to implement listed widgets.
The general syntax is:

w = Scrollbar(master, option=value)

master is the parameter used to represent the parent window.

There are 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.

1. width: to set the width of the widget.
2. activebackground: To set the background when mouse is over the widget.
3. bg: to set he normal background color.
4. bd: to set the size of border around the indicator.

5. cursor: To appear the cursor when the mouse over the menubutton.

Example:

```
from tkinter import *
root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill = Y )
mylist = Listbox(root, width=50, yscrollcommand = scrollbar.set )
for line in range(100):
   mylist.insert(END, 'You are on line number ' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )
mainloop()
```

Text

To edit a multi-line text and format the way it has to be displayed.
The general syntax is:

w =Text(master, option=value)

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

1. highlightcolor: To set the color of the focus highlight when widget has to be focused.
2. insertbackground: To set the background of the widget.
3. bg: to set he normal background color.
4. font: to set the font on the button label.
5. image: to set the image on the widget.
6. width: to set the width of the widget.
7. height: to set the height of the widget.

Example:

```
from tkinter import *
def onclick():
   pass
root = Tk()
text = Text(root)
text.insert(INSERT, "Hello, this is a test of Text widget.....")
text.insert(END, "\n\nBye, the test is over.....")
text.pack()
text.tag_add("here", "1.0", "1.8")
text.tag_add("start", "1.13", "1.16")
text.tag_config("here", background="yellow", foreground="blue")
```

```
text.tag_config("start", background="black", foreground="white")
root.mainloop()
```

TopLevel

This widget is directly controlled by the window manager. It don't need any parent window to work on.The general syntax is:

w = TopLevel(master, option=value)

There are 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.

1. bg: to set he normal background color.
2. bd: to set the size of border around the indicator.
3. cursor: To appear the cursor when the mouse over the menubutton.
4. width: to set the width of the widget.
5. height: to set the height of the widget.

Example:

```
from tkinter import *
root = Tk()
root.title('First Window')
top = Toplevel()
top.title('Second Window')
top.mainloop()
```

SpinBox

It is an entry of 'Entry' widget. Here, value can be input by selecting a fixed value of numbers.The general syntax is:

w = SpinBox(master, option=value)

There are 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.

1. bg: to set he normal background color.
2. bd: to set the size of border around the indicator.
3. cursor: To appear the cursor when the mouse over the menubutton.
4. command: To call a function.
5. width: to set the width of the widget.
6. activebackground: To set the background when mouse is over the widget.

7. disabledbackground: To disable the background when mouse is over the widget.
8. from_: To set the value of one end of the range.
9. to: To set the value of the other end of the range.

Example:

```
from tkinter import *
master = Tk()
w = Spinbox(master, from_ = 0, to = 10)
w.pack()
mainloop()
```

LabelFrame

A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts. This widget has the features of a frame plus the ability to display a label. Here is the simple syntax to create this widget:

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

There are 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.

1. bg: The normal background color displayed behind the label and indicator.,
2. bd:The size of the border around the indicator. Default is 2 pixels.
3. cursor: If you set this option to a cursor name (arrow, dot etc.), the mouse cursor will change to that pattern when it is over the checkbutton.
4. font: The vertical dimension of the new frame.
5. height: The vertical dimension of the new frame.
6. labelAnchor: Specifies where to place the label.
7. highlightbackground; Color of the focus highlight when the frame does not have focus.
8. highlightcolor: Color shown in the focus highlight when the frame has the focus.
9. highlightthickness: Thickness of the focus highlight.
10. relief: With the default value, relief=FLAT, the checkbutton does not stand out from its background. You may set this option to any of the other styles
11. text: Specifies a string to be displayed inside the widget.
12. width: Specifies the desired width for the window.

Example:

```
from tkinter import *
root = Tk()
labelframe = LabelFrame(root, text="This is a LabelFrame")
labelframe.pack(fill="both", expand="yes")
left = Label(labelframe, text="Inside the LabelFrame")
left.pack()
```

```
root.mainloop()
```

tkinter.messagebox

The tkinter.messagebox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message. Some of these functions are showinfo, showwarning, showerror, askquestion, askokcancel, askyesno, and askretryignore. Here is the simple syntax to create this widget −

tkinter.messagebox.FunctionName(title, message [, options])

Parameters

1. FunctionName − This is the name of the appropriate message box function.
2. title − This is the text to be displayed in the title bar of a message box.
3. message − This is the text to be displayed as a message.
4. options − options are alternative choices that you may use to tailor a standard message box. Some of the options that you can use are default and parent. The default option is used to specify the default button, such as ABORT, RETRY, or IGNORE in the message box. The parent option is used to specify the window on top of which the message box is to be displayed.

You could use one of the following functions with dialogue box −

1. showinfo()
2. showwarning()
3. showerror ()
4. askquestion()
5. askokcancel()
6. askyesno ()
7. askretrycancel ()

Example:

```
import tkinter
import tkinter.messagebox
top = tkinter.Tk()
def hello():
   tkinter.messagebox.showinfo("Say Hello", "Hello World")
B1 = tkinter.Button(top, text = "Say Hello", command = hello)
B1.pack()
top.mainloop()
```

Tkinter Canvas Widget

The Canvas widget lets us display various graphics on the application. It can be used to draw simple shapes to complicated graphs. We can also display various kinds of custom widgets according to our needs. You can draw several widgets in the canvas: arc bitmap, images, lines, rectangles, text, pieslices, ovals, polygons, ovals, polygons, and rectangles. Rectangles can be both outline and interior.

The canvas has two coordinate systems: the window system (left top corner x=0,y=0) and the canvas coordinate system that defines where items are drawn.

Syntax:
C = Canvas(root, height, width, bd, bg, ..)

Optional parameters:

1. root = root window.
2. height = height of the canvas widget.
3. width = width of the canvas widget.
4. bg = background colour for canvas.
5. bd = border of the canvas window.
6. scrollregion (w, n, e, s)tuple defined as a region for scrolling left, top, bottom and right
7. highlightcolor colour shown in the focus highlight.
8. cursor It can defind as a cursor for the canvas which can be a circle, a do, an arrow etc.
9. confine decides if canvas can be accessed outside the scroll region.
10. relief type of the border which can be SUNKEN, RAISED, GROOVE and RIDGE.

Some common drawing methods:

1. Creating an Oval: oval = C.create_oval(x0, y0, x1, y1, options)
2. Creating an arc: arc = C.create_arc(20, 50, 190, 240, start=0, extent=110, fill="red")
3. Creating a Line: line = C.create_line(x0, y0, x1, y1, ..., xn, yn, options)
4. Creating a polygon: oval = C.create_polygon(x0, y0, x1, y1, ...xn, yn, options)
5. Creating text: C.create_text(x0, y0, x1, y1, ...xn, yn, options)

Example:

```
from tkinter import *
root = Tk()
C = Canvas(root, bg ="yellow", height = 250, width = 300)
line = C.create_line(108, 120, 320, 40, fill ="green")
arc = C.create_arc(180, 150, 80, 210, start = 0, extent = 220,  fill ="red")
oval = C.create_oval(80, 30, 140, 150, fill ="blue")
C.pack()
```

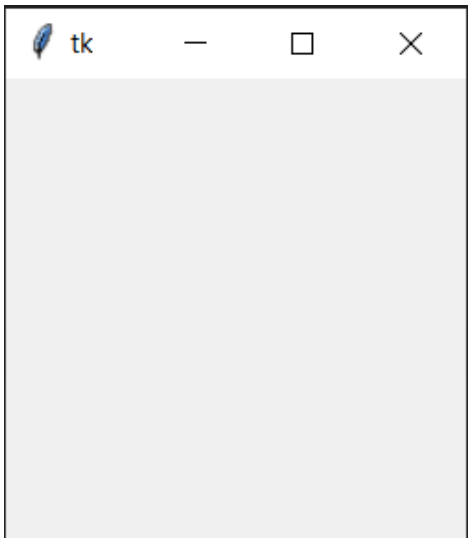
```
top.mainloop()
```

Results:

Program in tkinter_1.py:

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

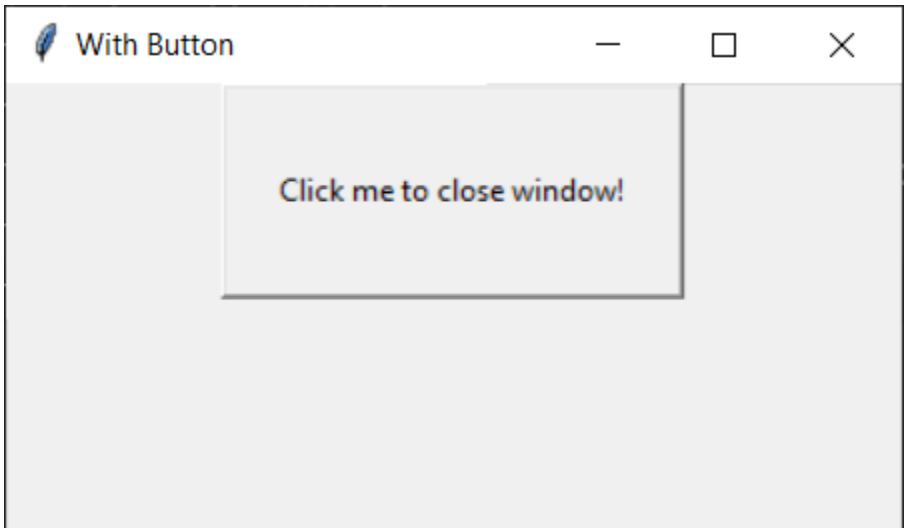
Output:



Program in tkinter_2.py:

```
import tkinter as tk
r = tk.Tk()
r.title('Hello')
button=tk.Button(r, text='Click me to close window!', width=25, height=5,
command=r.destroy)
button.pack()
r.mainloop()
```

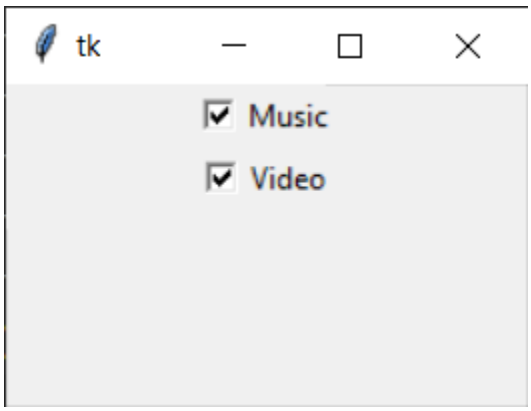
Output:



Program in tkinter_3.py:

```
from tkinter import *
master = Tk()
var1 = IntVar()
c1 = Checkbutton(master, text='Music', variable=var1)
var2 = IntVar()
c2 = Checkbutton(master, text='Video', variable=var2)
c1.pack()
c2.pack()
mainloop()
```

Output:

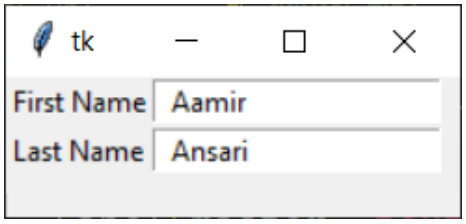


Program in tkinter_4.py:

```
from tkinter import *
```

```
master = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```

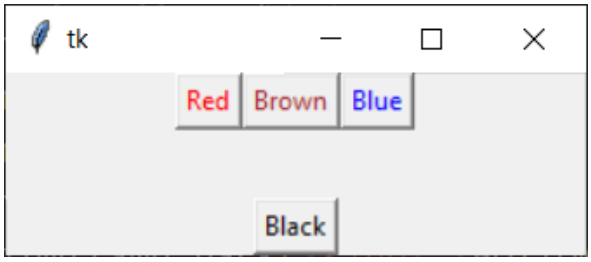
Output:



Program in tkinter_5.py:

```
from tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
bottomframe.pack( side = BOTTOM )
redbutton = Button(frame, text = 'Red', fg ='red')
redbutton.pack( side = LEFT)
greenbutton = Button(frame, text = 'Brown', fg='brown')
greenbutton.pack( side = LEFT )
bluebutton = Button(frame, text ='Blue', fg ='blue')
bluebutton.pack( side = LEFT )
blackbutton = Button(bottomframe, text ='Black', fg ='black')
blackbutton.pack( side = BOTTOM)
root.mainloop()
```

Output:



Program in tkinter_6.py:

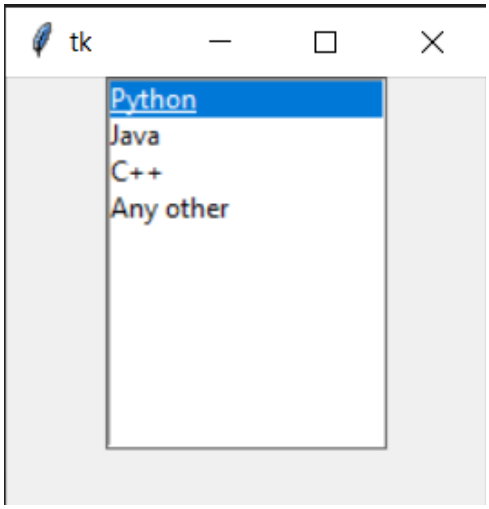
```
from tkinter import *
root = Tk()
w = Label(root, text="THIS IS A LABEL", fg="#06a099", font=("Arial", 15, 'bold'))
w.pack()
root.mainloop()
```

Output:

Program in tkinter_7.py:

```
from tkinter import *
top = Tk()
Lb = Listbox(top)
Lb.insert(1, 'Python')
Lb.insert(2, 'Java')
Lb.insert(3, 'C++')
Lb.insert(4, 'Any other')
Lb.pack()
top.mainloop()
```

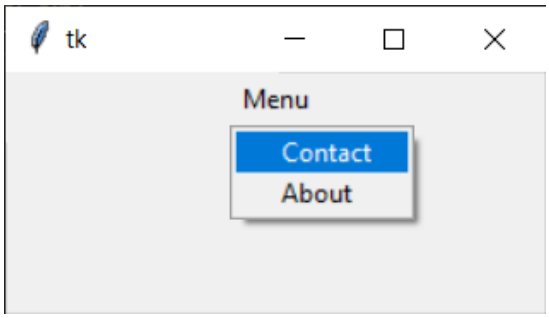
Output:



Program in tkinter_8.py:

```
from tkinter import *
top = Tk()
mb =  Menubutton ( top, text = "Menu")
mb.grid()
mb.menu  =  Menu ( mb, tearoff = 0 )
mb["menu"]  =  mb.menu
cVar  = IntVar()
aVar = IntVar()
mb.menu.add_checkbutton ( label = 'Contact', variable = cVar )
mb.menu.add_checkbutton ( label = 'About', variable = aVar )
mb.pack()
top.mainloop()
```

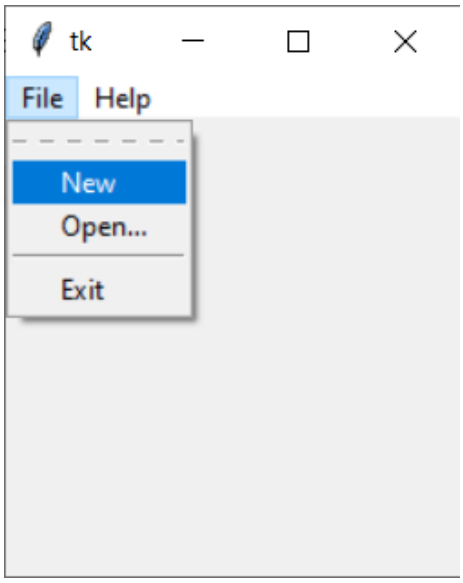
Output:



Program in tkinter_9.py:

```
from tkinter import *
root = Tk()
menu = Menu(root)
root.config(menu=menu)
filemenu = Menu(menu)
menu.add_cascade(label='File', menu=filemenu)
filemenu.add_command(label='New')
filemenu.add_command(label='Open...')
filemenu.add_separator()
filemenu.add_command(label='Exit', command=root.quit)
helpmenu = Menu(menu)
menu.add_cascade(label='Help', menu=helpmenu)
helpmenu.add_command(label='About')
mainloop()
```

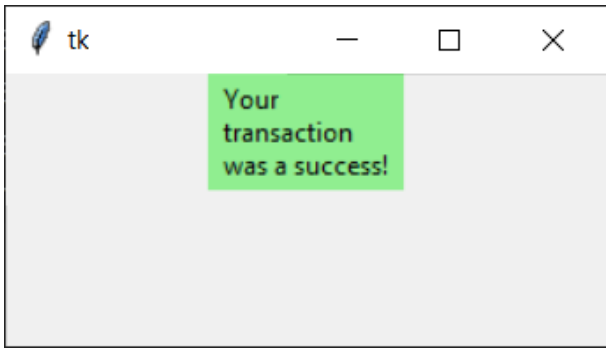
Output:



Program in tkinter_10.py:

```
from tkinter import *
main = Tk()
ourMessage ='Your transaction was a success!'
messageVar = Message(main, text = ourMessage)
messageVar.config(bg='lightgreen')
messageVar.pack( )
main.mainloop( )
```

Output:

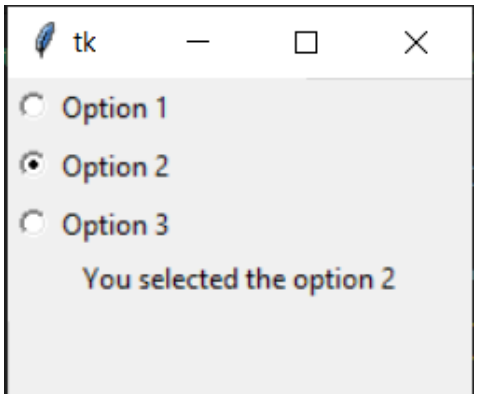


Program in tkinter_11.py:

```
from tkinter import *
def sel():
  selection = "You selected the option " + str(var.get())
```

```
    label.config(text = selection)
root = Tk()
var = IntVar()
R1 = Radiobutton(root, text="Option 1", variable=var, value=1,command=sel)
R1.pack( anchor = W )
R2 = Radiobutton(root, text="Option 2", variable=var, value=2,command=sel)
R2.pack( anchor = W )
R3 = Radiobutton(root, text="Option 3", variable=var, value=3,command=sel)
R3.pack( anchor = W)
label = Label(root)
label.pack()
root.mainloop()
```

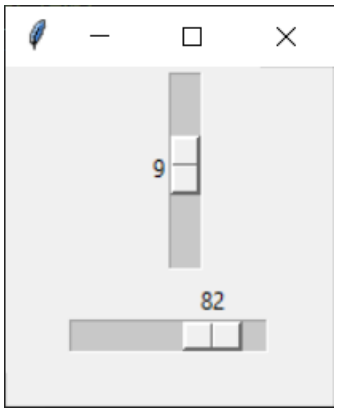
Output:



Program in tkinter_12.py:

```
from tkinter import *
master = Tk()
w = Scale(master, from_=0, to=20)
w.pack()
w = Scale(master, from_=0, to=100, orient=HORIZONTAL)
w.pack()
mainloop()
```

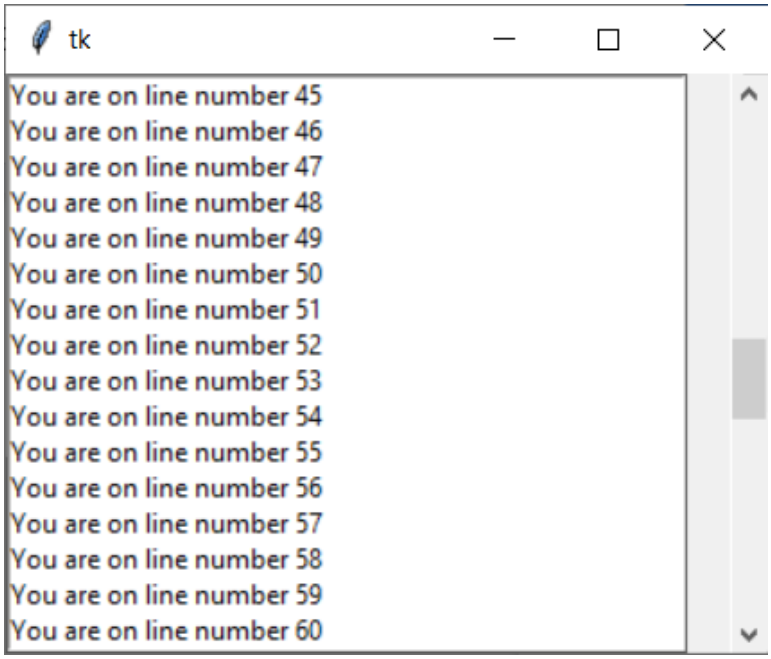
Output:



Program in tkinter_13.py:

```
from tkinter import *
root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill = Y )
mylist = Listbox(root, width=50, yscrollcommand = scrollbar.set )
for line in range(100):
   mylist.insert(END, 'You are on line number ' + str(line))
mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )
mainloop()
```

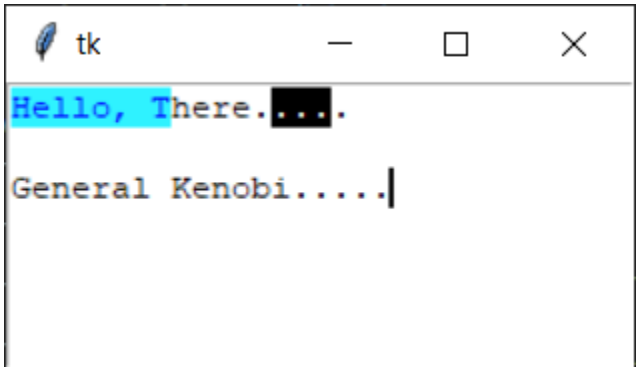
Output:



Program in tkinter_14.py:

```
from tkinter import *
def onclick():
   pass
root = Tk()
text = Text(root)
text.insert(INSERT, "Hello, this is a test of Text widget.....")
text.insert(END, "\n\nBye, the test is over.....")
text.pack()
text.tag_add("here", "1.0", "1.8")
text.tag_add("start", "1.13", "1.16")
text.tag_config("here", background="#2ff2fF", foreground="blue")
text.tag_config("start", background="black", foreground="white")
root.mainloop()
```

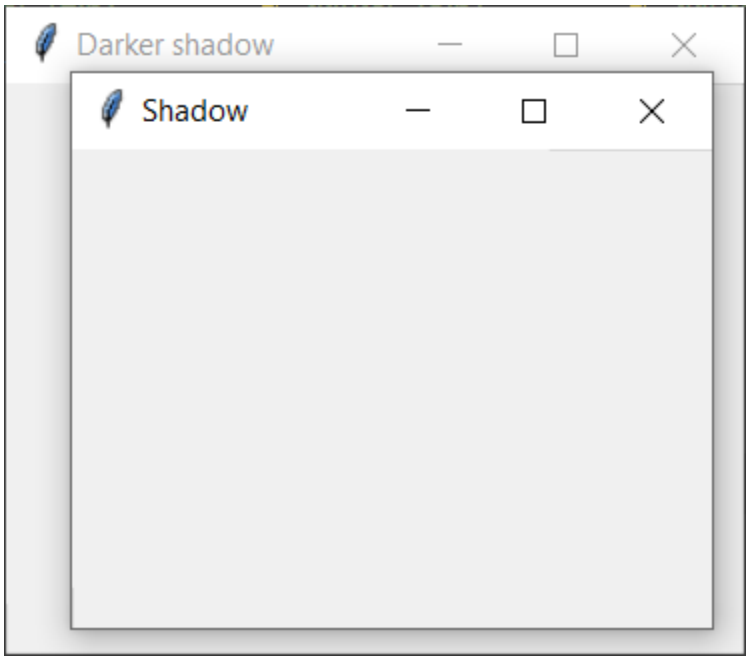
Output:



Program in tkinter_15.py:

```
from tkinter import *
root = Tk()
root.title('First Window')
top = Toplevel()
top.title('Second Window')
top.mainloop()
```

Output:



Program in tkinter_16.py:

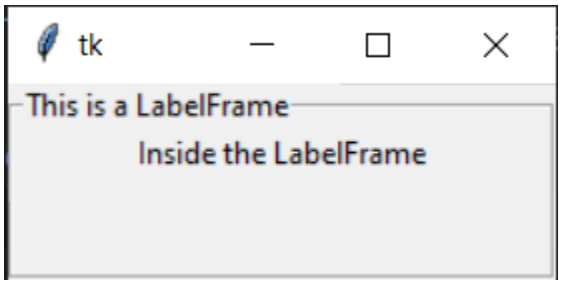
```
from tkinter import *
master = Tk()
w = Spinbox(master, from_ = 0, to = 10)
w.pack()
mainloop()
```

Output:

Program in tkinter_17.py:

```
from tkinter import *
root = Tk()
labelframe = LabelFrame(root, text="This is a LabelFrame")
labelframe.pack(fill="both", expand="yes")
left = Label(labelframe, text="Inside the LabelFrame")
left.pack()
root.mainloop()
```

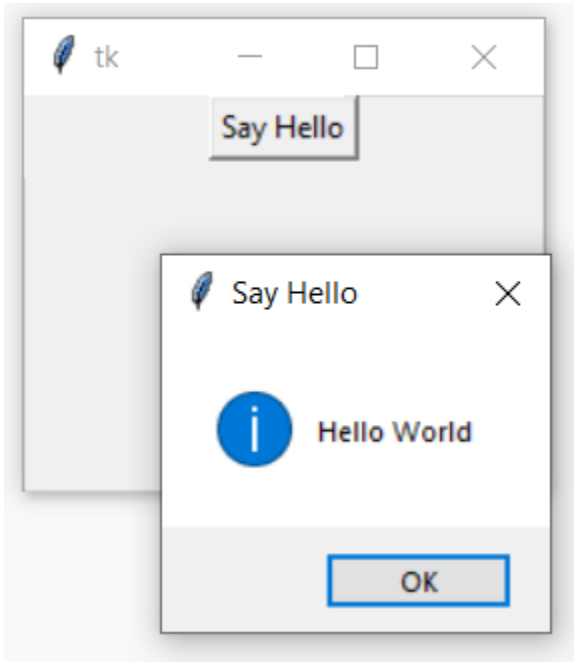
Output:



Program in tkinter_18.py:

```
import tkinter
import tkinter.messagebox
top = tkinter.Tk()
def hello():
   tkinter.messagebox.showinfo("Say Hello", "Hello World")
B1 = tkinter.Button(top, text = "Say Hello", command = hello)
B1.pack()
top.mainloop()
```

Output:



Program in canvas_1.py:

```
from tkinter import *
canvas_width = 600
canvas_height = 300
colours = ("#476042", "yellow")
box=[]
for ratio in ( 0.2, 0.35 ):
    box.append( (canvas_width * ratio, canvas_height * ratio, canvas_width * (1 - ratio),
canvas_height * (1 - ratio) ) )
master = Tk()
w = Canvas(master, width=canvas_width, height=canvas_height)
w.pack()
for i in range(2):
  w.create_rectangle(box[i][0], box[i][1],box[i][2],box[i][3], fill=colours[i])
w.create_line(0, 0, box[0][0], box[0][1],fill=colours[0], width=3)
w.create_line(0, canvas_height, box[0][0], box[0][3], fill=colours[0], width=3)
w.create_line(box[0][2],box[0][1], canvas_width, 0, fill=colours[0], width=3)
w.create_line(box[0][2], box[0][3], canvas_width, canvas_height, fill=colours[0],
width=3)
w.create_text(canvas_width / 2, canvas_height / 2, text="Python")
mainloop()
```

Output:

Program in canvas2.py:

```
from tkinter import *
canvas_width = 500
canvas_height = 150
def paint( event ):
   python_green = "#476042"
   x1, y1 = ( event.x - 1 ), ( event.y - 1 )
   x2, y2 = ( event.x + 1 ), ( event.y + 1 )
   w.create_oval( x1, y1, x2, y2, fill = python_green )
master = Tk()
master.title( "Painting using Ovals" )
w = Canvas(master,width=canvas_width,height=canvas_height)
w.pack(expand = YES, fill = BOTH)
w.bind( "<B1-Motion>", paint )
message = Label( master, text = "Press and Drag the mouse to draw" )
message.pack( side = BOTTOM )
mainloop()
```

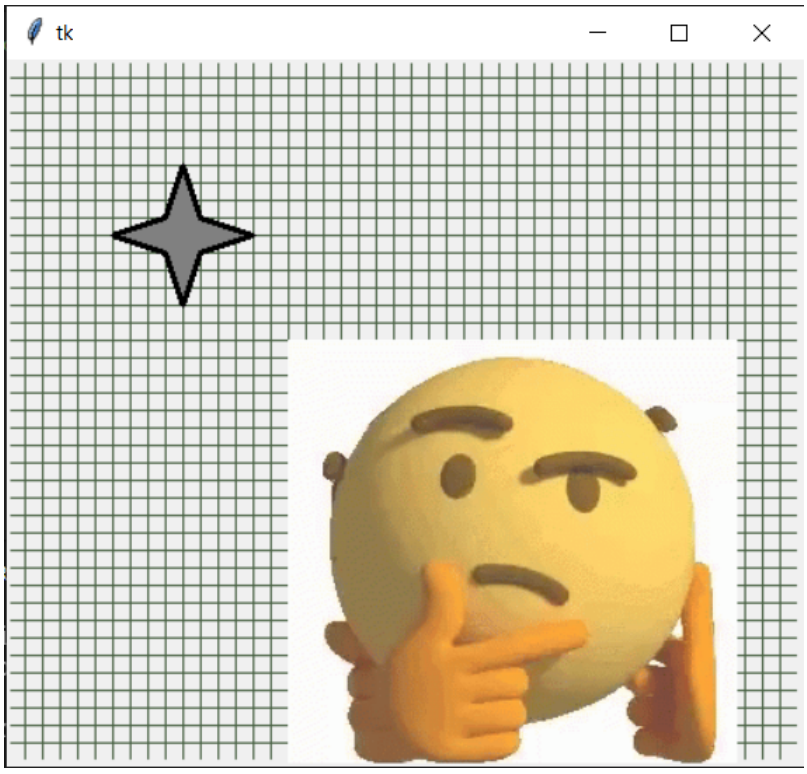
Output:

Program in canvas3.py:

```
from tkinter import *

def checkered(canvas, line_distance):
   for x in range(line_distance,canvas_width,line_distance):
```

```
    canvas.create_line(x, 0, x, canvas_height, fill="#476042")
  for y in range(line_distance,canvas_height,line_distance):
    canvas.create_line(0, y, canvas_width, y, fill="#476042")

canvas_width = 450
canvas_height =400
master = Tk()
canvas = Canvas(master, width=canvas_width, height=canvas_height)
canvas.pack()

#drawing a checkered pattern on the canvas
checkered(canvas,10)

#drawing a star-shaped polygon on the canvas
points = [100, 140, 110, 110, 140, 100, 110, 90, 100, 60, 90, 90, 60, 100, 90, 110]
canvas.create_polygon(points, outline='red', fill='yellow', width=3)

#draw gif iamge on the canvas
img = PhotoImage(file="nyan-cat.gif")
canvas.create_image(160,160, anchor=NW, image=img)
mainloop()
```

Output:



Conclusion:

Hence, we have successfully understood GUI Programming in Python by using tkinter, with hands-on experience with all the tkinter **widgets** and also canvas widget