Unit-1 : Developing Python GUI with Tkinter

1.1 introduction

1.2 Import Tkinter Libraries

1.3 Tkinter Widgets

1.4 Widgets Attributes

What Is A GUI?

- Every time you interact with the screen on your computer or your phone you are more than liking using a graphical user interface, or GUI.
- Those dazzling rectangular windows with buttons, icons, and menus were created for us to have a much easier way to interact with our electronic devices.

What Is A GUI?

- Older methods, such as MS-DOS, used the command line and text.
- While Python has a number of toolkits for making your own GUIs, Tkinter is good for learning the basics about UI development.

What is Tkinter for Python?

- **Tkinter** is a standard Python GUI (Graphical User Interface) library that provides a set of tools and widgets to create desktop applications with graphical interfaces.
- Tkinter is included with most Python installations, making it easily accessible for developers who want to build GUI applications without requiring additional installations or libraries.

Full Form of Tkinter

- The name “**Tkinter**” comes from “**Tk interface**“, referring to the Tk GUI toolkit that Tkinter is based on.
- Tkinter provides a way to create windows, buttons, labels, text boxes, and other GUI components to build interactive applications.

Significance of 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.
- You don’t need to worry about the installation of the Tkinter module separately as it comes with Python already.
- It gives an object-oriented interface to the Tk GUI toolkit.
- Among all, Tkinter is most widely used

Where is Python Tkinter used?

Here are some common use cases for Tkinter:

1.Creating windows and dialog boxes: Tkinter can be used to create windows and dialog boxes that allow users to interact with your program. These can be used to display information, gather input, or present options to the user.

Where is Python Tkinter used?

2.Building a GUI for a desktop application: Tkinter can be used to create the interface for a desktop application, including buttons, menus, and other interactive elements.

3.Adding a GUI to a command-line program: Tkinter can be used to add a GUI to a command-line program, making it easier for users to interact with the program and input arguments.

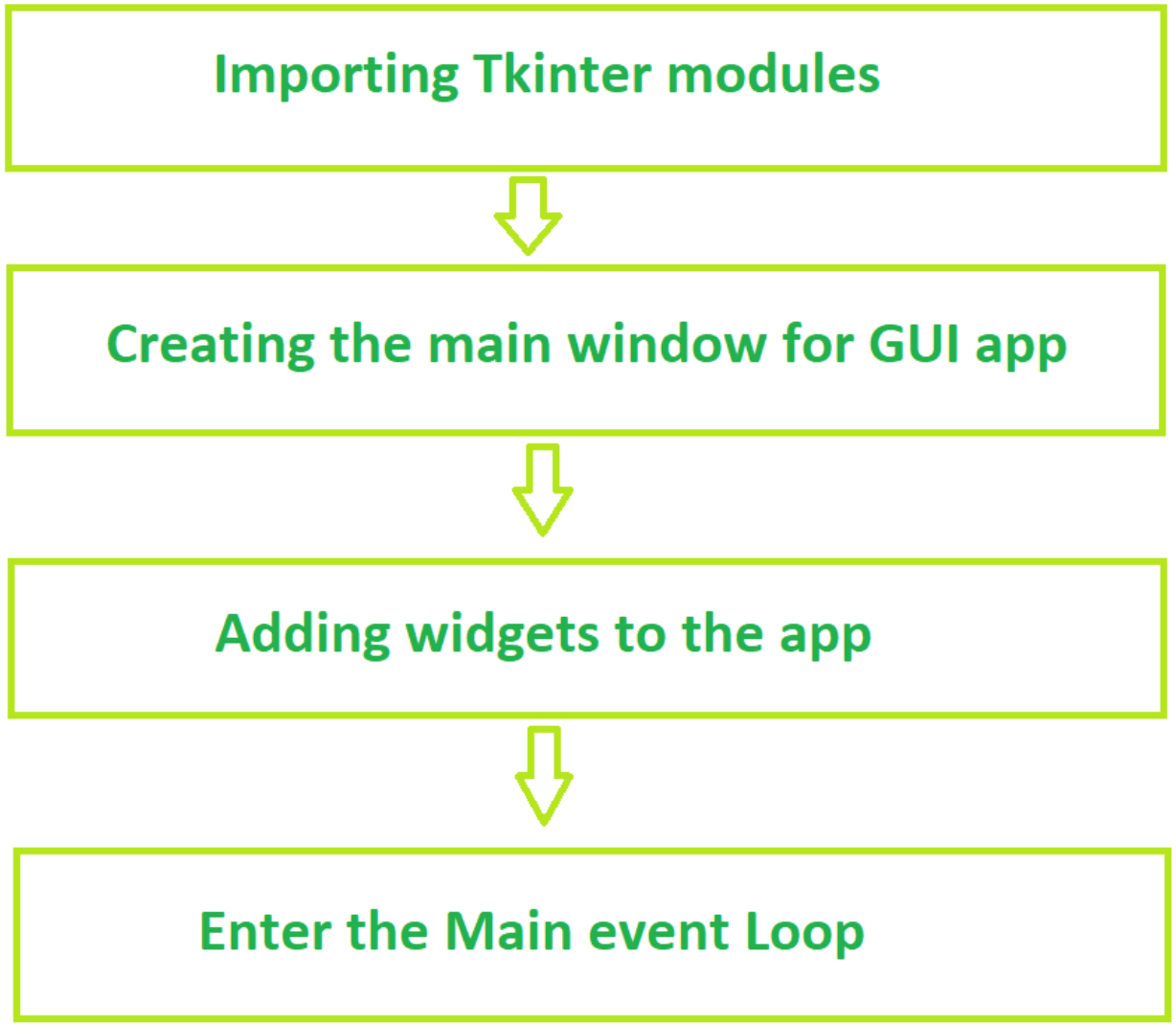
Where is Python Tkinter used?

4.Creating custom widgets: Tkinter includes a variety of built-in widgets, such as buttons, labels, and text boxes, but it also allows you to create your own custom widgets.

5.Prototyping a GUI: Tkinter can be used to quickly prototype a GUI, allowing you to test and iterate on different design ideas before committing to a final implementation.

- In summary, Tkinter is a useful tool for creating a wide variety of graphical user interfaces, including windows, dialog boxes, and custom widgets.
- It is particularly well-suited for building desktop applications and adding a GUI to command-line programs.

Fundamental structure of Tkinter program



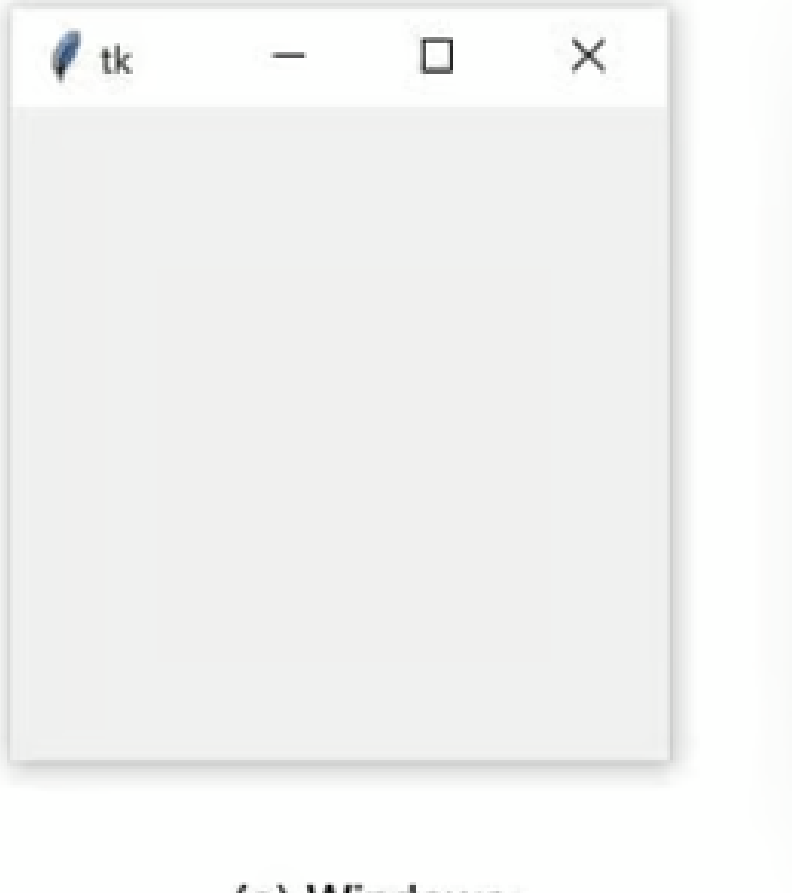
With your Python shell open, the first thing you need to do is import the Python GUI Tkinter module:

```Python
>>> import tkinter as tk
```

A **window** is an instance of Tkinter's Tk class. Go ahead and create a new window and assign it to the variable window:

```Python
>>> window = tk.Tk()
```

When you execute the above code, a new window pops up on your screen. How it looks depends on your operating system:



(a) Windows

1.3 Tkinter Widgets

Widget	Description
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Button	The Button widget is used to display buttons in your application.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.

1.3 Tkinter Widgets

Widget	Description
Menu	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

1.3 Tkinter Widgets

Widget	Description
Frame	The Frame widget is used as a container widget to organize other widgets.
Listbox	The Listbox widget is used to provide a list of options to a user.
Menubutton	The Menubutton widget is used to display menus in your application..

1.3 Tkinter Widgets

Widget	Description
Message	The Message widget is used to display multiline text fields for accepting values from a user..
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time..
Scale	The Scale widget is used to provide a slider widget.

1.3 Tkinter Widgets

Widget	Description
Scrollbar	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
Text	The Text widget is used to display text in multiple lines.
Toplevel	The Toplevel widget is used to provide a separate window container.

Widget	Description
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox	This module is used to display message boxes in your applications
Spinbox	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
PanedWindow	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

Adding a Widget

Now that you have a window, you can add a widget.

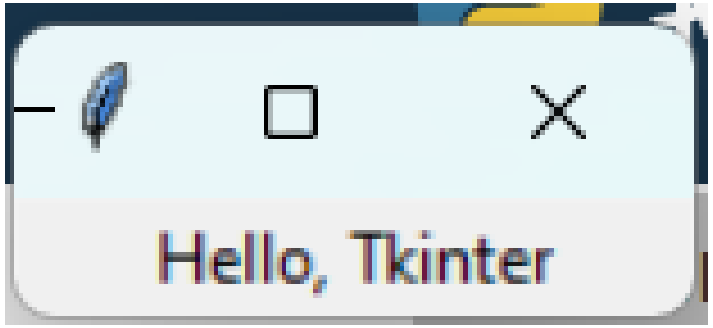
Use the tk.Label class to add some text to a window.

Create a Label widget with the text "Hello, Tkinter" and assign it to a variable called greeting:

```
greeting = tk.Label(text="Hello, Tkinter")
```

The window you created earlier doesn't change. You just created a Label widget, but you haven't added it to the window yet. There are several ways to add widgets to a window. Right now, you can use the Label widget's .pack() method:

```
greeting.pack()
```

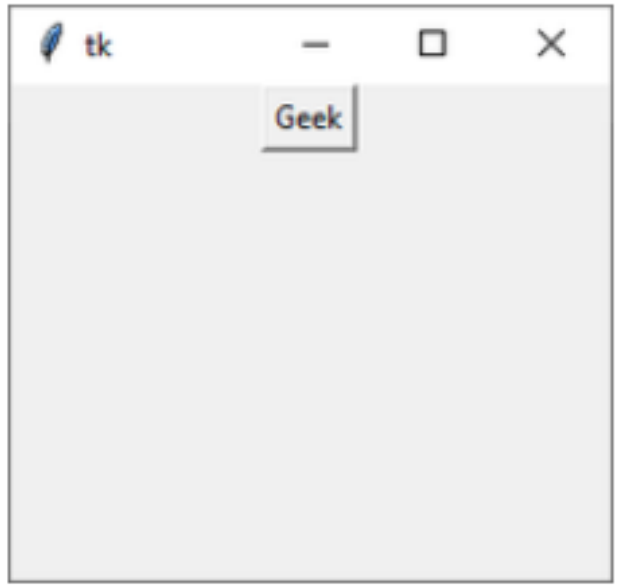


Geometry Management

- All Tkinter widgets have access to specific geometry management methods, which have the purpose of organizing widgets throughout the parent widget area.
- Tkinter exposes the following geometry manager classes: pack, grid, and place.

pack()	This geometry manager organizes widgets in blocks before placing them in the parent widget.
grid()	This geometry manager organizes widgets in a table-like structure in the parent widget.
place()	This geometry manager organizes widgets by placing them in a specific position in the parent widget.

```python
from tkinter import *

# create root window
root = Tk()

# frame inside root window
frame = Frame(root)

# geometry method
frame.pack()

# button inside frame which is
# inside root
button = Button(frame, text='Geek')
button.pack()

# Tkinter event loop
root.mainloop()
```



Tkinter Button Options

1. **activebackground**: Background color when the button is under the cursor.
2. **activeforeground**: Foreground color when the button is under the cursor.
3. **anchor**: Width of the border around the outside of the button
4. **bd or borderwidth**: Width of the border around the outside of the button
5. **bg or background:** Normal background color.
6. **command**: Function or method to be called when the button is clicked.
7. **cursor**: Selects the cursor to be shown when the mouse is over the button.
8. **text**: Text displayed on the button.
9. **disabledforeground**: Foreground color is used when the button is disabled.
10. **fg or foreground:** Normal foreground (text) color.

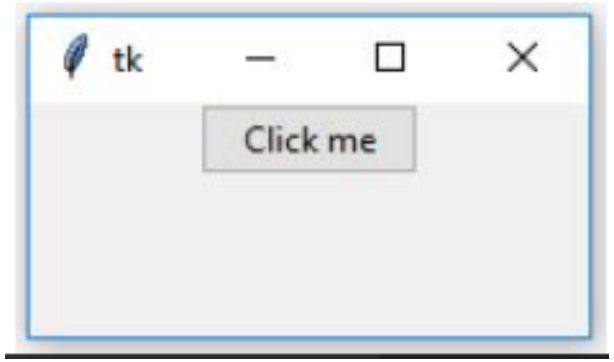
Tkinter Button Options

- **font**: Text font to be used for the button's label.
- **height**: Height of the button in text lines
- **highlightbackground**: Color of the focus highlight when the widget does not have focus.
- **highlightcolor**: The color of the focus highlight when the widget has focus.
- **highlightthickness**: Thickness of the focus highlight.
- **image**: Image to be displayed on the button (instead of text).
- **justify**: tk.LEFT to left-justify each line; tk.CENTER to center them; or tk.RIGHT to right-justify.
- **overrelief**: The relief style to be used while the mouse is on the button; default relief is tk.RAISED.
- **padx**, **pady**: padding left and right of the text. / padding above and below the text.
- **width**: Width of the button in letters (if displaying text) or pixels (if displaying an image).
- **underline**: Default is -1, underline=1 would underline the second character of the button's text.
- **width**: Width of the button in letters
- **wraplength**: If this value is set to a positive number, the text lines will be wrapped to fit within this length.

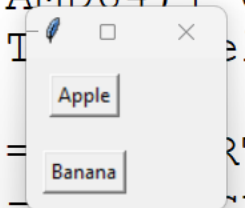
Creation of Button without using TK Themed Widget

- Creation of Button using **tk** themed widget (tkinter.ttk).
- This will give you the effects of modern graphics.
- Effects will change from one OS to another because it is basically for the appearance.
- As you can observe that BORDER is not present in 2nd output because tkinter.ttk does not support border.
- Also, when you hover the mouse over both the buttons ttk.Button will change its color and become light blue (effects may change from one OS to another) because it supports modern graphics while in the case of a simple Button it won't change color as it does not support modern graphics.

```python
# import tkinter module
from tkinter import *

# Following will import tkinter.ttk module and
# automatically override all the widgets
# which are present in tkinter module.
from tkinter.ttk import *

# Create Object
root = Tk()

# Initialize tkinter window with dimensions 100x100
root.geometry('100x100')

btn = Button(root, text = 'Click me !',
                command = root.destroy)

# Set the position of button on the top of window
btn.pack(side = 'top')

root.mainloop()
```



```python
# Import the library tkinter
from tkinter import *

# Create a GUI app
app = Tk()

# Create a function with one parameter to indicate which button was clicked
def which_button(button_text):
    # Printing the text when a button is clicked
    print(f"Button clicked: {button_text}")

# Creating and displaying of button b1
b1 = Button(app, text="Apple", command=lambda: which_button("Apple"))
b1.grid(padx=10, pady=10)

# Creating and displaying of button b2
b2 = Button(app, text="Banana", command=lambda: which_button("Banana"))
b2.grid(padx=10, pady=10)

# Make the infinite loop for displaying the app
app.mainloop()
```




```
AMD64)] on win32
T elp", "copyright

= RT: C:/Users/Len
-11/practical/button.py
Button clicked: Banana
```

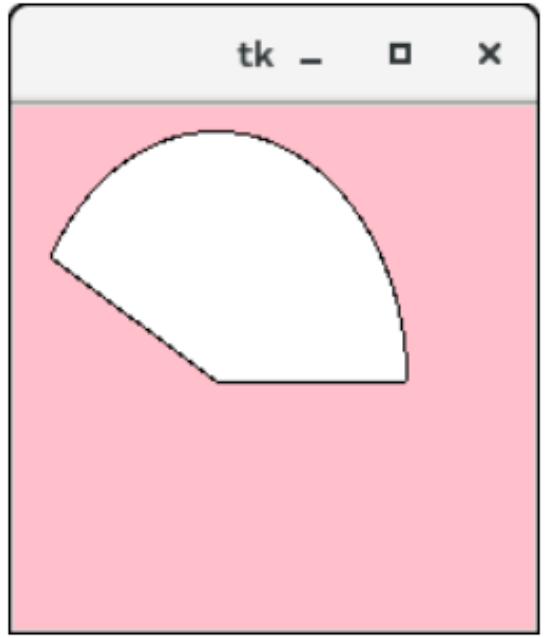
Python Tkinter Canvas

- 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:
- 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()
```

Explanation:

- **Coordinates (5, 10, 150, 200)**: Define the bounding box for the arc. The arc is drawn inside the rectangle defined by these coordinates.
- **start**: Specifies the starting angle in degrees, measured from the 3 o'clock position.
- **extent**: Defines the angle of the arc in degrees (swept counterclockwise).
- **fill**: Specifies the color to fill the arc. If omitted, the arc will not be filled.

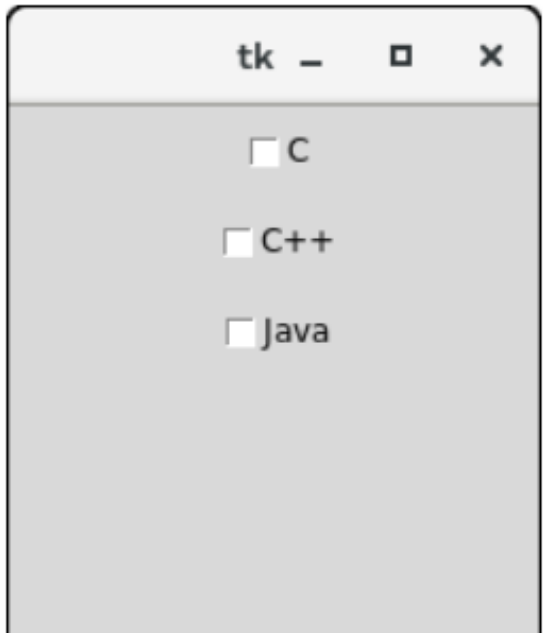
Python Tkinter Checkbutton

- 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.
- The syntax to use the checkbutton is given below

```
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, onval
ue = 1, offvalue = 0, height = 2, width = 10)
chkbtn2 = Checkbutton(top, text = "C++", variable = checkvar2, o
nvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn3 = Checkbutton(top, text = "Java", variable = checkvar3, o
nvalue = 1, offvalue = 0, height = 2, width = 10)
chkbtn1.pack()
chkbtn2.pack()
chkbtn3.pack()
top.mainloop()
```

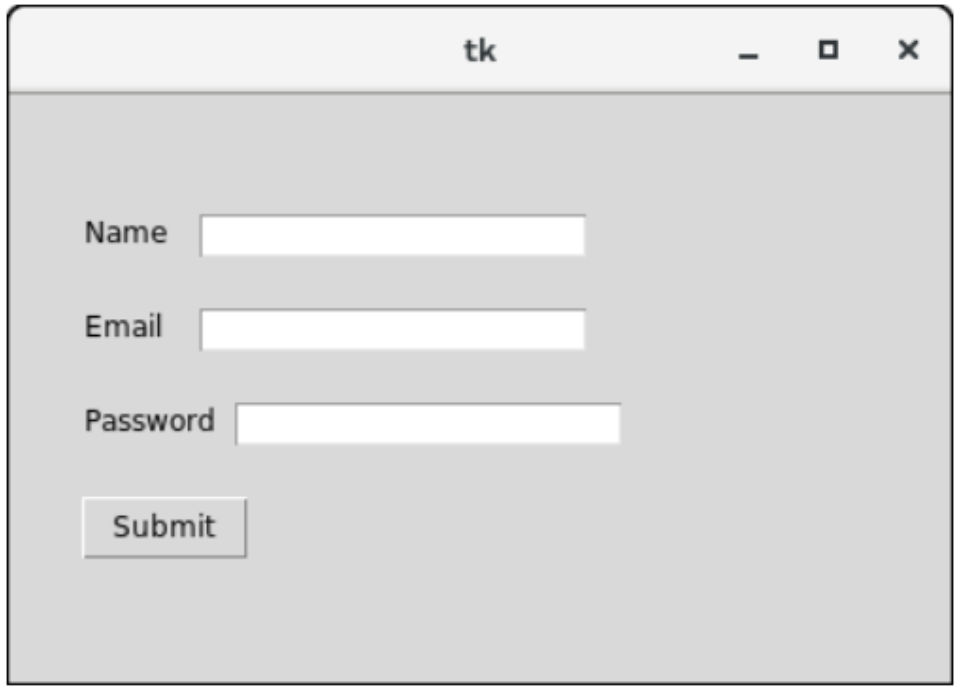
Python Tkinter Entry

- The Entry widget is used to provide 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.
- The syntax to use the Entry widget is given below.
- 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 = 1
30)
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_r
esult).grid(row=3, column=0)
root.mainloop()
```

Python Tkinter Frame

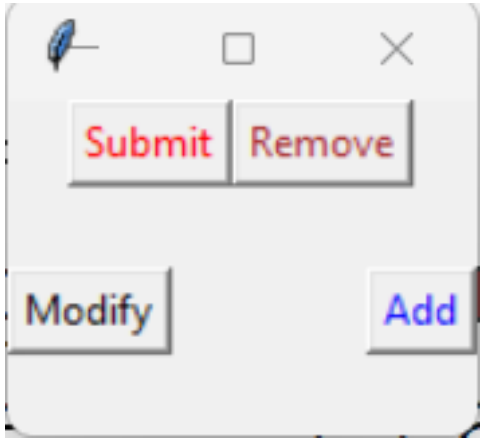
- Python Tkinter Frame widget is used to organize the group of widgets.
- It acts like a container which can be used to hold the other widgets.
- The rectangular areas of the screen are used to organize the widgets to the python application.
- The syntax to use the Frame widget is given below.
- w = Frame(parent, options)

Example

```
from tkinter import *

top = Tk()
top.geometry("140x100")
frame = Frame(top)
frame.pack()

leftframe = Frame(top)
leftframe.pack(side = LEFT)

rightframe = Frame(top)
rightframe.pack(side = RIGHT)
```

```
btn1 = Button(frame, text="Submit", fg="red",activebackg
round = "red")
btn1.pack(side = LEFT)

btn2 = Button(frame, text="Remove", fg="brown", activeb
ackground = "brown")
btn2.pack(side = RIGHT)

btn3 = Button(rightframe, text="Add", fg="blue", activeba
ckground = "blue")
btn3.pack(side = LEFT)
```

```
btn4 = Button(leftframe, text="Modify", fg="black", active
background = "white")
btn4.pack(side = RIGHT)

top.mainloop()
```



Python Tkinter Label

- The Label is used to specify the container box where we can place the text or images.
- This widget is used to provide the message to the user about other widgets used in the python application.
- There are the various options which can be specified to configure the text or the part of the text shown in the Label.
- The syntax to use the Label is given below.

w = Label (master, options)

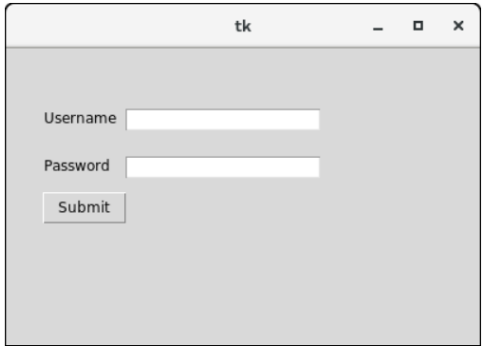
Example 1

```
from tkinter import *
top = Tk()  #Initialize Tkinter:create a main window
top.geometry("400x250")  #sets a window size
#creating label
uname = Label(top, text = "Username").place(x = 30,y = 50)
#top is parent widget
```

The place() method allows you to position widgets explicitly by specifying their coordinates

- **x**: The horizontal coordinate (distance from the left edge of the parent widget).
- **y**: The vertical coordinate (distance from the top edge of the parent widget).

```
#creating label
password = Label(top, text = "Password").place(x = 30, y = 9
0)
sbmitbtn = Button(top, text = "Submit",activebackground
= "pink", activeforeground = "blue").place(x = 30, y = 120)

e1 = Entry(top,width = 20).place(x = 100, y = 50)

e2 = Entry(top, width = 20).place(x = 100, y = 90)

top.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.
- The syntax to use the Listbox is given below.

```
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 listb
ox=listbox: listbox.delete(ANCHOR))

lbl.pack()
listbox.pack()

btn.pack()
top.mainloop()
```

lambda listbox=listbox:
listbox.delete(ANCHOR): Deletes the selected item from the Listbox.
•listbox.delete(ANCHOR): Removes the item at the position of the currently selected item (pointed to by ANCHOR)

After pressing the delete button.

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.
- The syntax to use the python tkinter Menubutton is given below.

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

Example

```
from tkinter import *

top = Tk()

top.geometry("200x250")

menubutton = Menubutton(top, text = "Language“)

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.
- The syntax to use the Menu widget is given below

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

Creating a top level menu

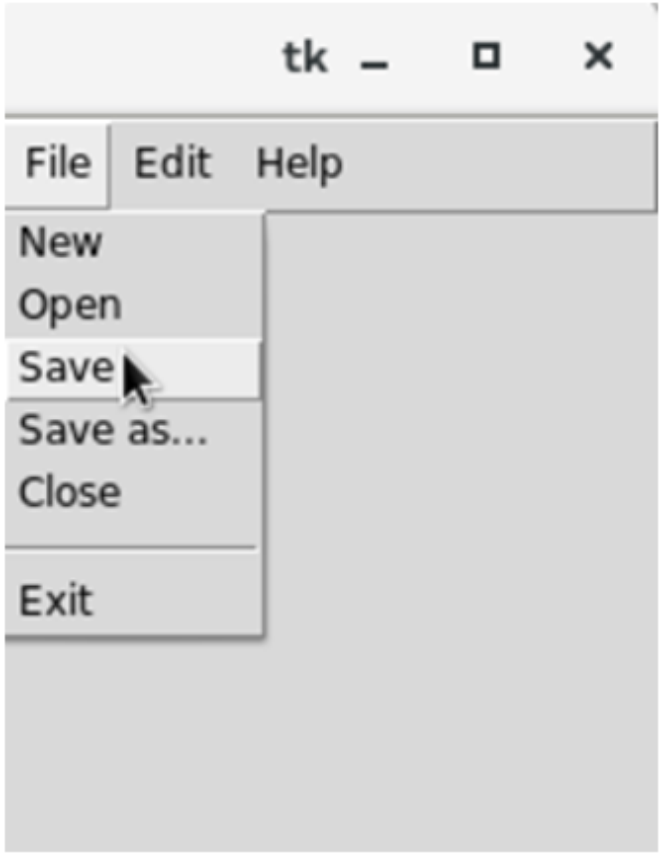
A top-level menu can be created by instantiating the Menu widget and adding the menu items to the menu

```
from tkinter import *

top = Tk()

def hello():
    print("hello!")

# create a toplevel menu
menubar = Menu(top)
menubar.add_command(label="Hello!", command=hello)
menubar.add_command(label="Quit!", command=top.quit)

# display the menu
top.config(menu=menubar)

top.mainloop()
```

Clicking the hello Menubutton will print the hello on the console while clicking the Quit Menubutton will make an exit from the python application.

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.
- The syntax to use the Message widget is given below.
- Syntax
- w = Message(parent, options)

Example

```
from tkinter import *

top = Tk()
top.geometry("100x100")
var = StringVar()
msg = Message( top, text = "Welcome to ASSC")

msg.pack()
top.mainloop()
```

Python Tkinter Radiobutton

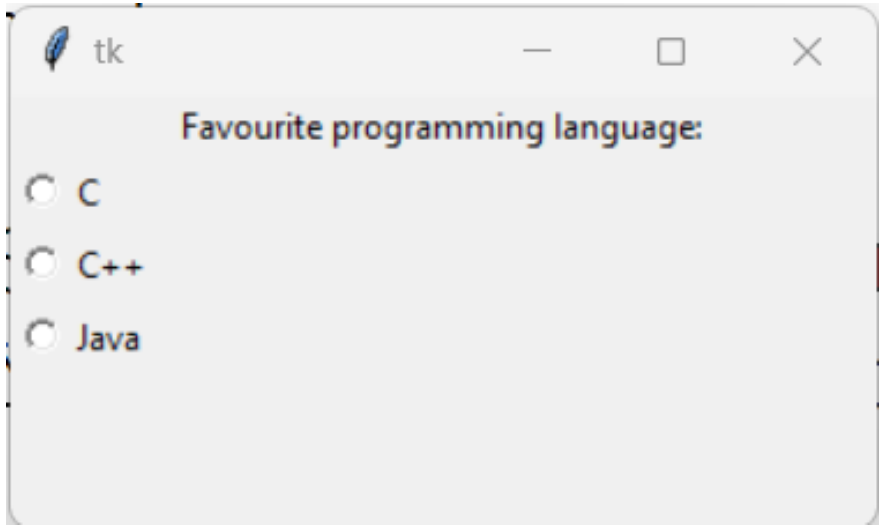
- The Radiobutton widget is used to implement one-of-many selection in the python application.
- It shows multiple choices to the user out of which, the user can select only one out of them.
- We can associate different methods with each of the radiobutton.
- We can display the multiple line text or images on the radiobuttons.
- To keep track the user's selection the radiobutton, it is associated with a single variable.
- Each button displays a single value for that particular variable.

- The syntax to use the Radiobutton is given below.
- Syntax
- w = Radiobutton(top, options)

```
from tkinter import *

def selection():
   selection = "You selected the option " + str(radio.get())
   label.config(text = selection)

top = Tk()
top.geometry("300x150")
radio = IntVar()
lbl = Label(text = "Favourite programming language:")
lbl.pack()
```

```
R1 = Radiobutton(top, text="C", variable=radio, value=1,
                 command=selection)
R1.pack( anchor = W )

R2 = Radiobutton(top, text="C++", variable=radio, value=2,
                 command=selection)
R2.pack( anchor = W )

R3 = Radiobutton(top, text="Java", variable=radio, value=3,

                 command=selection)
R3.pack( anchor = W)
```

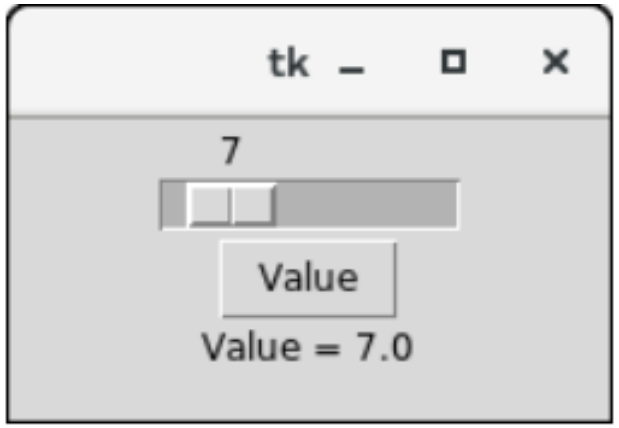
```
label = Label(top)
label.pack()
top.mainloop()
```



Python Tkinter Scale

- The Scale widget is used to implement the graphical slider to the python application so that the user can slide through the range of values shown on the slider and select the one among them.
- We can control the minimum and maximum values along with the resolution of the scale.
- It provides an alternative to the Entry widget when the user is forced to select only one value from the given range of values.
- The syntax to use the Scale widget is given below.
- Syntax
- w = Scale(top, options)

```
from tkinter import *

def select():
   sel = "Value = " + str(v.get())
   label.config(text = sel)

top = Tk()
top.geometry("200x100")
v = DoubleVar()
scale = Scale( top, variable = v, from_ = 1, to = 50, orient =
HORIZONTAL)
scale.pack(anchor=CENTER)
```

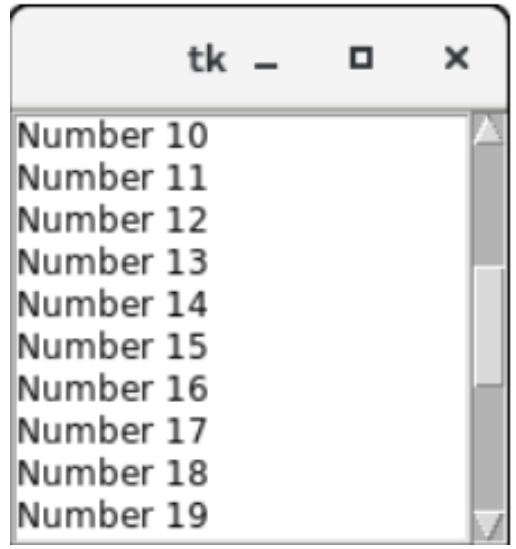


```
btn = Button(top, text="Value", command=select)
btn.pack(anchor=CENTER)

label = Label(top)
label.pack()

top.mainloop()
```

Python Tkinter Scrollbar

- The scrollbar widget is used to scroll down the content of the other widgets like listbox, text, and canvas.
- However, we can also create the horizontal scrollbars to the Entry widget.
- The syntax to use the Scrollbar widget is given below.
- Syntax
- w = Scrollbar(top, options)

```
from tkinter import *

top = Tk()
sb = Scrollbar(top)
sb.pack(side = RIGHT, fill = Y)

mylist = Listbox(top, yscrollcommand = sb.set )

for line in range(30):
    mylist.insert(END, "Number " + str(line))

mylist.pack( side = LEFT )
sb.config( command = mylist.yview )
mainloop()
```

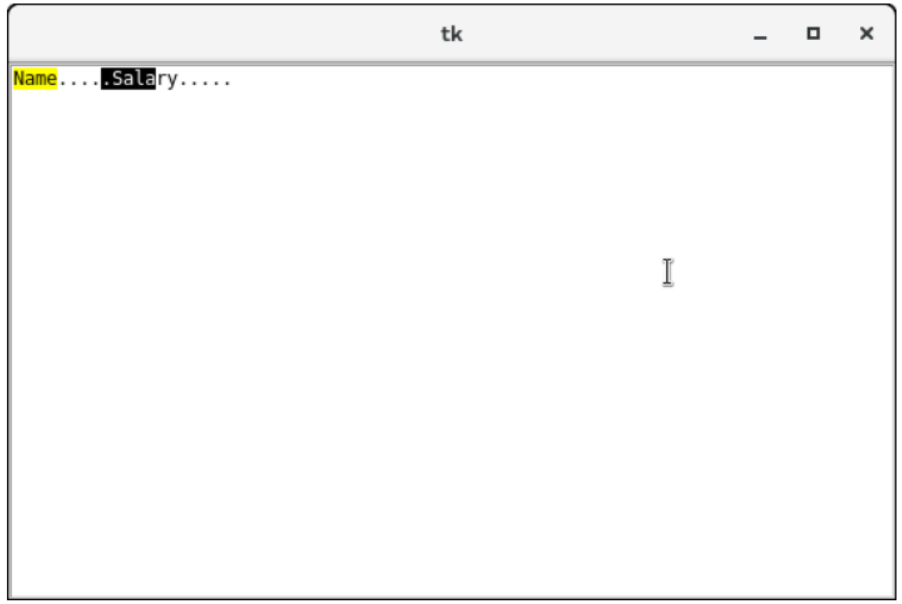


Python Tkinter Text

- The Text widget is used to show the text data on the Python application.
- However, Tkinter provides us the Entry widget which is used to implement the single line text box.
- The Text widget is used to display the multi-line formatted text with various styles and attributes.
- The Text widget is mostly used to provide the text editor to the user.
- The Text widget also facilitates us to use the marks and tabs to locate the specific sections of the Text.
- We can also use the windows and images with the Text as it can also be used to display the formatted text.

- The syntax to use the Text widget is given below.
- Syntax
- w = Text(top, options)

```
from tkinter import *

top = Tk()
text = Text(top)
text.insert(INSERT, "Name.....")
text.insert(END, "Salary.....")

text.pack()

text.tag_add("Write Here", "1.0", "1.4")
text.tag_add("Click Here", "1.8", "1.13")
```



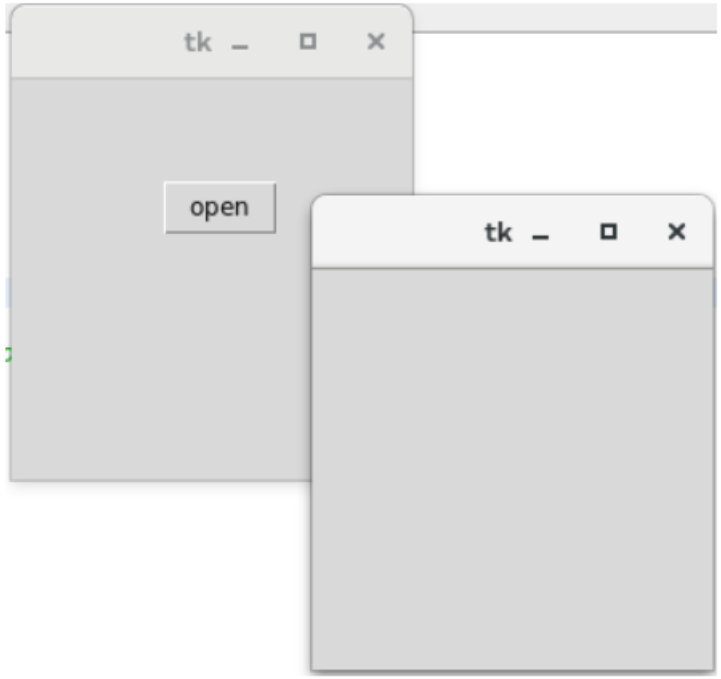
- **Syntax: tag_add(tag_name, start_index, end_index)**
- **start_index and end_index are in the format "line.column", where line is the line number (starting at 1) and column is the character index in that line (starting at 0).**
- **"1.0" means the first character of the first line.**
- **"1.4" means the 5th character (index 4, since counting starts from 0) in the first line.**

```
text.tag_config("Write Here", background="yellow", foregro
und="black")
text.tag_config("Click Here", background="black", foregrou
nd="white")

top.mainloop()
```

Tkinter Toplevel

- The Toplevel widget is used to create and display the toplevel windows which are directly managed by the window manager.
- The toplevel widget may or may not have the parent window on the top of them.
- The toplevel widget is used when a python application needs to represent some extra information, pop-up, or the group of widgets on the new window.
- The toplevel windows have the title bars, borders, and other window decorations.
- The syntax to use the Toplevel widget is given below.
- . Syntax
- w = Toplevel(options)

```
from tkinter import *

root = Tk()

root.geometry("200x200")

def open():
    top = Toplevel(root)
    top.mainloop()

btn = Button(root, text = "open", command = open)
btn.place(x=75,y=50)
root.mainloop()
```



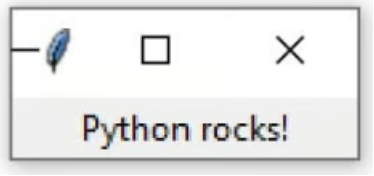
Assignment

1. What is Tkinter?
2. How do you import Tkinter in Python?
3. What are Tkinter widgets?
4. Name five commonly used Tkinter widgets.
5. How do you create a Button widget in Tkinter?
6. What is the purpose of the pack() method in Tkinter?
7. How can you change the background color of a Label widget?
8. What is a widget attribute in Tkinter?
9. How do you bind an event to a widget in Tkinter?
10. What does the mainloop() method do in Tkinter?

- https://www.javatpoint.com/python-tkinter-menubutton

Practical List

1.Write a full Python script that creates a Tkinter window with the text "Python rocks!".



2. Write a full python script that creates a simple button.

3. Write a full python script that **Check which Button was clicked in Tkinter.**

(if the text '**It is an apple**' is printed on the screen, we get to know '**Apple**' button is pressed, else when '**It is a banana**' is printed on the screen, we get to know '**Banana**' button is pressed.)

Important:
https://www.geeksforgeeks.org/how-to-check-which-button-was-clicked-in-tkinter/?ref=next_article
https://www.javatpoint.com/python-tkinter-button

For practicals:
https://www.w3resource.com/python-exercises/tkinter/index-basic.php

Extra:
https://www.pythonguis.com/tutorials/create-gui-tkinter/
https://www.geeksforgeeks.org/python-tkinter-tutorial/
https://www.geeksforgeeks.org/create-first-gui-application-using-python-tkinter/
https://www.geeksforgeeks.org/introduction-to-tkinter/?ref=lbp
https://realpython.com/python-gui-tkinter/