Python Tkinter

GUI Environment For Python

TKInter

- Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python.
- Python with tkinter is the fastest and easiest way to create the GUI applications.

To create a tkinter app:

1. Importing the module – tkinter
2. Create the main window (container)
3. Add any number of widgets to the main window
4. Apply the event Trigger on the widgets

Tkinter

- Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library.
- **Steen Lumholt and Guido van Rossum** are the founder of tkinter.
- Tkinter has several strengths. It's **cross-platform**, so the same code works on Windows, macOS, and Linux.

List of Other Python GUI Frameworks

- PyQt5
- Tkinter
- WxPython
- Pyside2
- Kivy
- Libvag

Install Tkinter

- Open command prompt and check
  - python –version
- (it will check whether python Is installed or not)
- Also check pip is installed or not using following command
  - Pip –v
- Install tkinter
  - Pip install tk

TKInter

- 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

Tkinter

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

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

To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'.

To change the name of the window, you can change the className to the desired one.

The basic code used to create the main window of the application is:

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

Tkinter

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.

```
Example : m.mainloop()
```

First Program to Create GUI

- import tkinter
- m = tkinter.Tk()
- '‘’ this is multiline comment
- widgets are added here
- '''
- m.mainloop()

First Program to Create GUI

window.mainloop() tells Python to run the Tkinter **event loop**.
This method listens for events, such as button clicks or keypresses, and blocks any code that comes after it from running until you close the window where you called the method.

Resizing GUI

After importing, setup the application object by calling the Tk() function.

This will create a top-level window (root) having a frame with a title bar, control box with the minimize and close buttons, and a client area to hold other widgets.

The geometry() method defines the width, height and coordinates of the top left corner of the frame as below
(all values are in pixels):
window.geometry("widthxheight+XPOS+YPOS")

Resizing GUI

The application object then enters an event listening loop by calling the mainloop() method.

The application is now constantly waiting for any event generated on the elements in it.

The event could be text entered in a text field, a selection made from the dropdown or radio button, single/double click actions of mouse, etc.

Geometry

```
import tkinter
window=tkinter.Tk()
window.title('Hello Python’)
window.geometry("300x200+10+10")
window.mainloop()
```

Minsize, Maxsize

When a window is resizable, you can specify the minimum and maximum sizes using the minsize() and maxsize() methods:

window.minsize(min_width, min_height)
window.maxsize(min_height, max_height)

Transparency

Tkinter allows you to specify the transparency of a window by setting its alpha channel ranging from 0.0 (fully transparent) to 1.0 (fully opaque):

```
window.attributes('-alpha',0.5)
```

Change default icon

Events of Tkinter

- Tkinter is a Python library which is used to create GUI-based application.
- Tkinter Events are generally used to provide an interface that works as a bridge between the User and the application logic. We can use Events in any Tkinter application to make it operable and functional.
- Tkinter provides a mechanism to let the programmer deal with events. For each widget, it's possible to bind Python functions and methods to an event.
- *widget.bind(event, handler)*

Events of Tkinter

- **<Button>** – Use the Button event in a handler for binding the Mouse wheels and Buttons.
- **<ButtonRelease>** – Instead of clicking a Button, you can also trigger an event by releasing the mouse buttons.
- **Destroy** – Use this event to kill or terminate a particular widget.
- **<Expose>** – The event occurs whenever a widget or some part of the application becomes visible that covered by another window in the application.
- **<Focus In>** – This event is generally used to get the focus on a particular widget.
- **<Focus Out>** – To move the focus from the current widget.
- **<Key Press>** – Start the process or call the handler by pressing the key.
- **<KeyRelease>** – Start the process or call an event by releasing a key.
- **<Leave>** – Use this event to track the mouse pointer when user switches from one widget to another widget.
- **<Motion>** – Track the event whenever the mouse pointer moves entirely within the application.

Python Tkinter – Message

- The Message widget is used to show the message to the user regarding the behavior of the python application.
- The message text contains more than one line
- Syntax
- W=message(master,options)

- **master**: This parameter is used to represents the parent window.
- **options**:There are many options which are available and they can be used as key-value pairs separated by commas.

Python Tkinter – Message

Bg	The background color of the widget.
Bitmap	It is used to display the graphics on the widget. It can be set to any graphical or image object.
Bd	It represents the size of the border in the pixel. The default size is 2 pixel.
Cursor	The mouse pointer is changed to the specified cursor type. The cursor type can be an arrow, dot, etc.
Font, size, underline, bold	The font type of the widget text.
Fg	The font color of the widget text.
Height	The vertical dimension of the message.
Image	We can set this option to a static image to show that onto the widget.
Justify	This option is used to specify the alignment of multiple line of code with respect to each other. The possible values can be LEFT (left alignment), CENTER (default), and RIGHT (right alignment).

Python Tkinter – Message

Padx	The horizontal padding of the widget.
Pady	The vertical padding of the widget.
relief	It represents the type of the border. The default type is FLAT. flat, groove, raised, ridge, solid, or sunken
text	We can set this option to the string so that the widget can represent the specified text.
width	It specifies the horizontal dimension of the widget in the number of characters (not pixel).
wraplength	We can wrap the text to the number of lines by setting this option to the desired number so that each line contains only that number of characters.

Example

```
from tkinter import *
root = Tk()
root.geometry("300x200")
msg = Message( root, text = “Welcome to Dr Subhash University")
msg.pack()
root.mainloop()
```

PACK()

- *Pack* is the easiest Layout Manager to code with in Tkinter.
- **Instead of declaring the precise location of a widget, the *pack()* method declares the position of widgets in relation to each other.**
- For simple positioning of widgets vertically or horizontally in relation to each other, *pack()* is the Layout Manager of choice.

Pack.fill()

- **The *pack() fill* option is used to make a widget fill the entire frame.**
- The *pack() expand* option is used to expand the widget if the user expands the frame.
- ***fill* options:**
- *NONE* (default), which will keep the widget's original size.
- *X*, fill horizontally.
- *Y*, fill vertically.
- *BOTH*, fill horizontally and vertically.
- *expand* options:
- Numerical values.

PACK()

- *side* is the most basic option, and includes padding options for positioning widgets in relation to the left, right, top, bottom sides of the widget, and relative to each other.
- (side=left, top, right, bottom)
- *fill* packs a widget inside a container, filling the entire container.
- *expand* is an option for assigning additional space to the widget container.
- *padx*, which pads externally along the x axis.
- *pady,* which pads externally along the y axis.
- *ipadx*, which pads internally along the x axis.
- *ipady*, which pads internally along the y axis.
- Syntax of pack
- Widget.pack(options)

Tkinter Button

- **activebackground**
- Background color when the button is under the cursor. **activeforeground**
- Foreground color when the button is under the cursor
- **bd**
- Border width in pixels. Default is 2.
- **bg**
- Normal background color
- **command**
- Function or method to be called when the button is clicked.

Tkinter Button

- **fg**
- Normal foreground (text) color
- **font**
- Text font to be used for the button's labe
- **height**
- Height of the button in text lines (for textual buttons) or pixels (for images)
- **highlightcolor**
- The color of the focus highlight when the widget has focus.
- **image**
- Image to be displayed on the button (instead of text)

Tkinter Button

- **justify**
- How to show multiple text lines: LEFT to left-justify each line; CENTER to center them; or RIGHT to right-justify
- **state**
- Set this option to DISABLED to gray out the button and make it unresponsive. Has the value ACTIVE when the mouse is over it. Default is NORMAL.
- **underline**
- Default is -1, meaning that no character of the text on the button will be underlined. If nonnegative, the corresponding text character will be underline
- **Demo**

Tkinter Button Click

```
import Tkinter
import tk MessageBox
top = Tkinter.Tk()

def helloCallBack():
tkMessageBox.showinfo( "Hello Python", "Hello World")

B = Tkinter.Button(top, text ="Hello", command = helloCallBack)
B.pack()
top.mainloop()
```

Tkinter Button Destroy

```
from tkinter import *
from tkinter.ttk import *
root = Tk()
root.geometry('100x100')
btn = Button(root, text = 'Click me !',command = root.destroy)
btn.pack(side = 'top')
root.mainloop()
```

```
from tkinter import *
from tkinter.ttk import *
def left_click(event):
    print("click")
def left_double_click(event):
    print("double click")
def motion(event):
    print("motion")
def right_click(event):
    print("right click")
root = Tk()
root.geometry('100x100')
btn = Button(root, text = 'Click me !')
btn.pack(side = "top")
btn.bind("<Button-1>", left_click)
btn.bind("<Button-3>", right_click)
btn.bind("<Double-1>", left_double_click)
btn.bind('<Motion>', motion)
```

Tkinter MessageBox

- Python Tkinter – MessageBox Widget is used to display the message boxes in the python applications. This module is used to display a message using provides a number of functions.

- Sytnax
- `messagebox.Function_Name(title,message[, options])`

Tkinter MessageBox

- **Function_Name:** This parameter is used to represents an appropriate message box function.
- **title:** This parameter is a string which is shown as a title of a message box.
- **message:** This parameter is the string to be displayed as a message on the message box.
- **options:** There are two options that can be used are:
  - **default:** This option is used to specify the default button like ABORT, RETRY, or IGNORE in the message box.
  - **parent:** This option is used to specify the window on top of which the message box is to be displayed

Tkinter MessageBox

- **Function_Name:**
  There are functions or methods available in the messagebox widget.

1.showinfo(): Show some relevant information to the user.

2.showwarning(): Display the warning to the user.

3.showerror(): Display the error message to the user.

4.askquestion(): Ask question and user has to answered in yes or no.

5.askokcancel(): Confirm the user's action regarding some application activity.

6.askyesno(): User can answer in yes or no for some action.

7.askretrycancel(): Ask the user about doing a particular task again or not.

Tkinter MessageBox

```
from tkinter import *
from tkinter import messagebox
root = Tk()
root.geometry("300x200")
w = Label(root, text =Demo Messagebox', font = “90")
w.pack()
messagebox.showinfo("showinfo", "Information")
messagebox.showwarning("showwarning", "Warning")
messagebox.showerror("showerror", "Error")
messagebox.askquestion("askquestion", "Are you sure?")
messagebox.askokcancel("askokcancel", "Want to continue?")
messagebox.askyesno("askyesno", "Find the value?")
messagebox.askretrycancel("askretrycancel", "Try again?")
root.mainloop()
```

Same Program using Variable

Tkinter RadioButton

- This widget implements a multiple-choice button, which is a way to offer many possible selections to the user and lets user choose only one of them.
- In order to implement this functionality, each group of radiobuttons must be associated to the same variable and each one of the buttons must symbolize a single value

Tkinter RadioButton

- This widget implements a multiple-choice button, which is a way to offer many possible selections to the user and lets user choose only one of them.
- In order to implement this functionality, each group of radiobuttons must be associated to the same variable and each one of the buttons must symbolize a single value.
- **Syntax:**
- w = Radiobutton (master, option, ...)

Tkinter RadioButton

- **activebackground**
- The background color when the mouse is over the radiobutton.
- **activeforeground**
- The foreground color when the mouse is over the radiobutton.
- **anchor**
- If the widget inhabits a space larger than it needs, this option specifies where the radiobutton will sit in that space. The default is anchor=CENTER.
- **bg**
- The normal background color behind the indicator and label.
- **bitmap**
- To display a monochrome image on a radiobutton, set this option to a bitmap.
- **borderwidth**
- The size of the border around the indicator part itself. Default is 2 pixels.
- **command**
- A procedure to be called every time the user changes the state of this radiobutton

Tkinter RadioButton

- **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 radiobutton.
- **font**
- The font used for the text.
- **fg**
- The color used to render the text.
- **height**
- The number of lines (not pixels) of text on the radiobutton. Default is 1

Tkinter RadioButton

- **highlightbackground**
- The color of the focus highlight when the radiobutton does not have focus.
- **highlightcolor**
- The color of the focus highlight when the radiobutton has the focus.
- **image**
- To display a graphic image instead of text for this radiobutton, set this option to an image object
- **relief**
- Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.
- **selectcolor**
- The color of the radiobutton when it is set. Default is red

Tkinter RadioButton

- **state**
- The default is state=NORMAL, but you can set state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the radiobutton, the state is ACTIVE.
- **text**
- The label displayed next to the radiobutton. Use newlines ("\n") to display multiple lines of text.
- **textvariable**
- To slave the text displayed in a label widget to a control variable of class *StringVar*, set this option to that variable.

Tkinter RadioButton

- **value**
- When a radiobutton is turned on by the user, its control variable is set to its current value option.
- If the control variable is an *IntVar*, give each radiobutton in the group a different integer value option.
- If the control variable is a *StringVar*, give each radiobutton a different string value option.
- **variable**
- The control variable that this radiobutton shares with the other radiobuttons in the group. This can be either an IntVar or a StringVar

Tkinter RadioButton 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()
```

Tkinter RadioButton Example

```
from tkinter import *
master = Tk()
master.geometry("175x175")
v = StringVar(master, "1")
values = {"RadioButton 1" : "1",
                    "RadioButton 2" : "2",
                    "RadioButton 3" : "3",
                    "RadioButton 4" : "4",
                    "RadioButton 5" : "5"}
for (text, value) in values.items():
        Radiobutton(master, text = text, variable = v,
                                        value = value, indicator = 0,
                                        background = "light blue").pack(fill = X, ipady = 5)
mainloop()
```

Tkinter Canvas

- A tkinter canvas **can be used to draw in a window**. Use this widget to draw graphs or plots.
- You can even use it to create graphical editors. You can draw several widgets in the canvas: arc bitmap, images, lines, rectangles, text, ovals, polygons, ovals, and rectangles.
- Syntax:
- w = Canvas (master, option=value, ...)

Tkinter Canvas

bd Border width in pixels. Default is 2.
bg Normal background color.
confine If true (the default), the canvas cannot be scrolled outside of the scrollregion.
cursor Cursor used in the canvas like *arrow, circle, dot etc.*
height Size of the canvas in the Y dimension.
highlightcolor Color shown in the focus highlight

Examples

- **arc** – Creates an arc item, which can be a chord, a pieslice or a simple arc.
- Example:
- coord = 10, 50, 240, 210
- arc = canvas.create_arc(coord, start=0, extent=150, fill="blue")

Examples

- **image** – Creates an image item, which can be an instance of either the BitmapImage or the PhotoImage classes.

filename = PhotoImage(file = "sunshine.gif")

image = canvas.create_image(50, 50, anchor=NE, image=filename)

- **line** – Creates a line item.
- line = canvas.create_line(x0, y0, x1, y1, ..., xn, yn, options)

Examples

- **image** – Creates an image item, which can be an instance of either the BitmapImage or the PhotoImage classes.

filename = PhotoImage(file = "sunshine.gif")

image = canvas.create_image(50, 50, anchor=NE, image=filename)

- **line** – Creates a line item.
- line = canvas.create_line(x0, y0, x1, y1, ..., xn, yn, options)

Examples

- **oval** – Creates a circle or an ellipse at the given coordinates. It takes two pairs of coordinates; the top left and bottom right corners of the bounding rectangle for the oval
- Example:

oval = canvas.create_oval(x0, y0, x1, y1, options)

polygon – Creates a polygon item that must have at least three vertices.

- oval = canvas.create_polygon(x0, y0, x1, y1,...xn, yn, options)

All Examples

ARC, Oval, Image, Line, Polygon Practices:

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.
- Syntax:
- w = Scale (master, option, ...)

Tkinter Scale

activebackground

- The background color when the mouse is over the scale.

bg

- The background color of the parts of the widget that are outside the trough.
- **bd**
- Width of the 3-d border around the trough and slider. Default is 2 pixels.
- **command**
- A procedure to be called every time the slider is moved. This procedure will be passed one argument, the new scale value. If the slider is moved rapidly, you may not get a callback for every possible position, but you'll certainly get a callback when it settles.

Tkinter Scale

- **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 scale.
- **digits**
- The way your program reads the current value shown in a scale widget is through a control variable.
- The control variable for a scale can be an IntVar, a DoubleVar (float), or a StringVar. If it is a string variable, the digits option controls how many digits to use when the numeric scale value is converted to a string.

Tkinter Scale

- **font**
- The font used for the label and annotations.
- **fg**
- The color of the text used for the label and annotations.
- **from_**
- A float or integer value that defines one end of the scale's range.
- **highlightbackground**
- The color of the focus highlight when the scale does not have focus
- **highlightcolor**
- The color of the focus highlight when the scale has the focus

Tkinter Scale

- **label**
- You can display a label within the scale widget by setting this option to the label's text. The label appears in the top left corner if the scale is horizontal, or the top right corner if vertical. The default is no label.
- **length**
- The length of the scale widget. This is the x dimension if the scale is horizontal, or the y dimension if vertical. The default is 100 pixels
- **orient**
- Set orient=HORIZONTAL if you want the scale to run along the x dimension, or orient=VERTICAL to run parallel to the y-axis. Default is horizontal.
- **relief**
- Specifies the appearance of a decorative border around the label. The default is FLAT; for other values

Tkinter Scale

- **resolution**
- Normally, the user will only be able to change the scale in whole units. Set this option to some other value to change the smallest increment of the scale's value.
- For example, if from_=1 and to=10, and you set resolution=0.5, the scale will have possible values: 1.0, 1.5, 2, 2.5, 3...

Tkinter Scale

- **showvalue**
- Normally, the current value of the scale is displayed in text form by the slider (above it for horizontal scales, to the left for vertical scales). Set this option to 0 to suppress that label.
- **sliderlength**
- Normally the slider is 30 pixels along the length of the scale. You can change that length by setting the sliderlength option to your desired length
- **state**
- Normally, scale widgets respond to mouse events, and when they have the focus, also keyboard events. Set state=DISABLED to make the widget unresponsive.

Tkinter Scale

- **takefocus**
- Normally, the focus will cycle through scale widgets. Set this option to 0 if you don't want this behavior.
- **tickinterval**
- To display periodic scale values, set this option to a number, and ticks will be displayed on multiples of that value.
- For example, if from_=0.0, to=1.0, and tickinterval=0.25, labels will be displayed along the scale at values 0.0, 0.25, 0.50, 0.75, and 1.00. These labels appear below the scale if horizontal, to its left if vertical. Default is 0, which suppresses display of ticks.

Tkinter Scale

To:

A float or integer value that defines one end of the scale's range; the other end is defined by the from_ option, discussed above. The to value can be either greater than or less than the from_ value. For vertical scales, the to value defines the bottom of the scale; for horizontal scales, the right end.

variable

- The control variable for this scale, if any. Control variables may be from class IntVar, DoubleVar (float), or StringVar. In the latter case, the numerical value will be converted to a string.

Tkinter Scale Methods

- **get()**
- This method returns the current value of the scale.
- **set (value)**
- Sets the scale's value

- **Exercise/Demo Program**

Tkinter Checkbutton

- The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option.
- w = Checkbutton (master, option, ..)

Tkinter Checkbutton

- **offvalue:** The associated control variable is set to 0 by default if the button is unchecked.
- **onvalue:** The associated control variable is set to 1 by default if the button is checked.
- **variable:** This option used to represents the associated variable that tracks the state of the checkbutton.

Tkinter Checkbutton

- **deselect():** This method is called to turn off the checkbutton.
- **select():** This method is called to turn on the checkbutton.
- **toggle():** This method is used to toggle between the different Checkbuttons.

Tkinter Checkbutton

```
from tkinter import *
root = Tk()
root.geometry("300x200")
w = Label(root, text ='GeeksForGeeks', font = "50")
w.pack()
Checkbutton1 = IntVar()
Checkbutton2 = IntVar()
Checkbutton3 = IntVar()
Button1 = Checkbutton(root, text = "Tutorial",variable = Checkbutton1,onvalue = 1,offvalue = 0,height = 2,width = 10)
Button2 = Checkbutton(root, text = "Student",variable = Checkbutton2,onvalue = 1,offvalue = 0,height = 2,width = 10)
Button3 = Checkbutton(root, text = "Courses",variable = Checkbutton3,onvalue = 1,offvalue = 0,height = 2,width = 10)
Button1.pack()
Button2.pack()
Button3.pack()
mainloop()
```

Tkinter Scrollbar

Jump:

It is used to control the behavior of the scroll jump. If it set to 1, then the callback is called when the user releases the mouse button.

Orient:

It can be set to HORIZONTAL or VERTICAL depending upon the orientation of the scrollbar.

Tkinter Scrollbar

get()	It returns the two numbers a and b which represents the current position of the scrollbar.
Set (first, last)	It is used to connect the scrollbar to the other widget w. The yscrollcommand or xscrollcommand of the other widget to this method

Tkinter Scrollbar

```
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()
```

Tkinter Entry

- The Entry widget is used to accept single-line text strings from a user.
- If you want to display multiple lines of text that can be edited, then you should use the *Text* widget.
- If you want to display one or more lines of text that cannot be modified by the user, then you should use the *Label* widget
- Syntax:

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

Tkinter Entry

- **exportselection**
- By default, if you select text within an Entry widget, it is automatically exported to the clipboard. To avoid this exportation, use exportselection=0.
- **highlightcolor**
- The color of the focus highlight when the entry has the focus
- **justify**
- If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.
- **selectbackground**
- The background color to use displaying selected text

Tkinter Entry

- **selectborderwidth**
- The width of the border to use around selected text. The default is one pixel.
- **selectforeground**
- The foreground (text) color of selected text.
- **show**
- Normally, the characters that the user types appear in the entry.
- To make a password entry that echoes each character as an asterisk, set show="*"

Tkinter Entry

- **textvariable**
- In order to be able to retrieve the current text from your entry widget, you must set this option to an instance of the StringVar class
- **width**
- The default width of a checkbutton is determined by the size of the displayed image or text. You can set this option to a number of characters and the checkbutton will always have room for that many characters.
- **xscrollcommand**
- If you expect that users will often enter more text than the onscreen size of the widget, you can link your entry widget to a scrollbar.

Tkinter Entry Methods

- **delete (first, last=None)**
- Deletes characters from the widget, starting with the one at index first, up to but not including the character at position last. If the second argument is omitted, only the single character at position first is deleted
- **get()**
- Returns the entry's current text as a string
- **icursor (index)**
- Set the insertion cursor just before the character at the given index
- **insert (index, s)**
- Inserts string s before the character at the given index

Tkinter Entry Methods

- **select_clear()**
- Clears the selection. If there isn't currently a selection, has no effect.
- **select_present()**
- If there is a selection, returns true, else returns false.
- **select_range (start, end)**
- Sets the selection under program control. Selects the text starting at the start index, up to but not including the character at the end index. The start position must be before the end position
- **select_to (index)**
- Selects all the text from the ANCHOR position up to but not including the character at the given index.

Tkinter Entry Methods

- **xview (index)**
- This method is useful in linking the Entry widget to a horizontal scrollbar
- **xview_scroll (number, what)**
- Used to scroll the entry horizontally. The what argument must be either UNITS, to scroll by character widths, or PAGES, to scroll by chunks the size of the entry widget. The number is positive to scroll left to right, negative to scroll right to left.

Tkinter Entry Example

```
from Tkinter import *
top = Tk()
L1 = Label(top, text="User Name")
L1.pack( side = LEFT)
E1 = Entry(top, bd =5)
E1.pack(side = RIGHT)
top.mainloop()
```

Tkinter Frame

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way.
It works like a container, which is responsible for arranging the position of other widgets.

Syntax:

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

Tkinter Frame

- **bg**
- The normal background color displayed behind the label and indicator
- **bd**
- The size of the border around the indicator. Default is 2 pixels
- **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.
- **height**
- The vertical dimension of the new frame.

Tkinter Frame

- **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 style
- **width**
- The default width of a checkbutton is determined by the size of the displayed image or text.
- You can set this option to a number of characters and the checkbutton will always have room for that many characters

Tkinter Frame Example

How to Hide/show widget in python?

```
def show_widget():
label.pack()
def hide_widget():
label.pack_forget()
b1.configure(text="Show", command=show_widget)
b2.configure(text=“Hide", command=hide_widget)
```

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.
- Syntax:
- W=toplevel(options)

Tkinter Toplevel Properties

- Bg, bd, cursor, font, fg, height, relief, width

Tkinter Toplevel Example

```
from tkinter import *
root = Tk()
root.geometry("200x300")
root.title("main")
l = Label(root, text = "This is root window")
top = Toplevel()
top.geometry("180x100")
top.title("toplevel")
l2 = Label(top, text = "This is toplevel window")
l.pack()
l2.pack()
top.mainloop()
```

Tkinter SpinBox

- The Spinbox widget is **an alternative to the Entry widget**.
- It provides the range of values to the user, out of which, the user can select the one.
- It is used in the case where a user is given some fixed number of values to choose from.
- Syntax:
- W=Spinbox(master,options)

Tkinter SpinBox

- **command:** This option is associated with a function to be called when the state is changed.
- **cursor:** By using this option, the mouse cursor will change to that pattern when it is over the type.
- **disabledforeground:** This option used to represent the foreground color of the widget when it is disabled..
- **disabledbackground:** This option used to represent the background color of the widget when it is disabled

Tkinter SpinBox

- **font:** This option used to represent the font used for the text.
- **fg:** This option used to represent the color used to render the text.
- **format:** This option used to formatting the string and it’s has no default value.
- **from_:** This option used to represent the minimum value.
- **justify:** This option used to control how the text is justified: CENTER, LEFT, or RIGHT.
- **relief:** This option used to represent the type of the border and It’s default value is set to SUNKEN.
- increment

Tkinter SpinBox

- **state:** This option used to represent the represents the state of the widget and its default value is NORMAL.
- **textvariable:** This option used to control the behaviour of the widget text.
- **to:** It specify the maximum limit of the widget value. The other is specified by the from_ option.
- **validate:** This option is used to control how the widget value is validated.
- **validatecommand:** This option is associated to the function callback which is used for the validation of the widget content.
- **values:** This option used to represent the tuple containing the values for this widget.

Tkinter SpinBox Method

- **delete(startindex, endindex):** This method is used to delete the characters present at the specified range.
- **get(startindex, endindex):** This method is used to get the characters present in the specified range.
- **identify(x, y):** This method is used to identify the widget's element within the specified range.
- **index(index):** This method is used to get the absolute value of the given index.
- **insert(index, string):** This method is used to insert the string at the specified index.
- **invoke(element):** This method is used to invoke the callback associated with the widget.

Tkinter SpinBox Example

```
from tkinter import *

root = Tk()
root.geometry("300x200")
w = Label(root, text =‘Dr Subhash University’, font = "50")
w.pack()
sp = Spinbox(root, from_= 0, to = 20)
sp.pack()
root.mainloop()
```

Tkinter SpinBox Example

```
import tkinter as tk
from tkinter import *
my_w = tk.Tk()
my_w.geometry("300x150")
my_w.title(“Welcome")
sb1=Spinbox(my_w, from_= 0, to = 10,width=5)
sb1.grid(row=1,column=1,padx=20,pady=20)
my_list=['One', 'Two', 'Three', 'Four', 'Five’]
 sb2=Spinbox(my_w,values=my_list,width=10)
sb2.grid(row=1,column=2,padx=20,pady=20)
my_w.mainloop()
```

Tkinter SpinBox Example

Display two Spinboxes showing numbers and on click of a button display the sum of the selected numbers.

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.

ListBox

bg

The normal background color displayed behind the label and indicator

bd

The size of the border around the indicator. Default is 2 pixels

cursor

The cursor that appears when the mouse is over the listbox.

font

The font used for the text in the listbox.

fg

The color used for the text in the listbox.

ListBox

- **height**
- Number of lines (not pixels!) shown in the listbox. Default is 10
- **highlightcolor**
- Color shown in the focus highlight when the widget has the focus
- **highlightthickness**
- Thickness of the focus highlight.
- **relief**
- Selects three-dimensional border shading effects. The default is SUNKEN
- **selectbackground**
- The background color to use displaying selected text

ListBox

- **selectmode**
- Determines how many items can be selected, and how mouse drags affect the selection –
- **SINGLE** – You can only select one line, and you can't drag the mouse.wherever you click button 1, that line is selected.
- **MULTIPLE** – You can select any number of lines at once. Clicking on any line toggles whether or not it is selected.
- **EXTENDED** – You can select any adjacent group of lines at once by clicking on the first line and dragging to the last line.

ListBox Methods

- **curselection()**
- Returns a tuple containing the line numbers of the selected element or elements, counting from 0. If nothing is selected, returns an empty tuple
- **delete (first, last=None)**
- Deletes the lines whose indices are in the range [first, last]. If the second argument is omitted, the single line with index first is deleted.

ListBox Methods

- **get (first, last=None)**
- Returns a tuple containing the text of the lines with indices from first to last, inclusive. If the second argument is omitted, returns the text of the line closest to first.
- **insert (index, *elements)**
- Insert one or more new lines into the listbox before the line specified by index. Use END as the first argument if you want to add new lines to the end of the listbox.

- **nearest (y)**
- Return the index of the visible line closest to the y-coordinate y relative to the listbox widget.
- **size()**
- Returns the number of lines in the listbox.

ListBox Example

```
from Tkinter import *
import tkMessageBox
import Tkinter
top = Tk()
Lb1 = Listbox(top)
Lb1.insert(1, "Python")
Lb1.insert(2, "Perl")
Lb1.insert(3, "C")
Lb1.insert(4, "PHP")
Lb1.insert(5, "JSP")
Lb1.insert(6, "Ruby")
Lb1.pack()
top.mainloop()
```

Tkinter Menubutton

- The Menubutton widget can be defined as the drop-down menu that is shown to the user all the time.
- The Menubutton is used to implement various types of menus in the python application.
- Syntax
- `w = Menubutton ( master, options )`

Tkinter Menubutton

- **activebackground:** This option used to represent the background color when the Menubutton is under the cursor.
- **activeforeground:** This option used to represent the foreground color when the Menubutton is under the cursor.
- **bg:** This option used to represent the normal background color displayed behind the label and indicator.

Tkinter Menubutton

- **bd:** This option used to represent the size of the border around the indicator and the default value is 2 pixels.
- **cursor:** By using this option, the mouse cursor will change to that pattern when it is over the Menubutton.
- **disabledforeground:** The foreground color used to render the text of a disabled Menubutton. The default is a stippled version of the default foreground color.
- **direction:** It direction can be specified so that menu can be displayed to the specified direction of the button.
- **fg:** This option used to represent the color used to render the text.

Tkinter Menubutton

highlightcolor: This option used to represent the color of the focus highlight when the Menubutton has the focus.

justify: This option used to control how the text is justified: CENTER, LEFT, or RIGHT.

menu: It represents the menu specified with the Menubutton.

padx: This option used to represent how much space to leave to the left and right of the Menubutton and text. It’s default value is 1 pixel.

pady: This option used to represent how much space to leave above and below the Menubutton and text. It’s default value is 1 pixel.

Tkinter Menubutton

text: This option used use newlines (“\n”) to display multiple lines of text.

textvariable: This option used to represents the associated variable that tracks the state of the Menubutton.

height: This option used to represent the number of lines of text on the Menubutton and it’s default value is 1.

relief: The type of the border of the Menubutton. It’s default value is set to FLAT.

state: It represents the state of the Menubutton. By default, it is set to normal. We can change it to DISABLED to make the Menubutton unresponsive. The state of the Menubutton is ACTIVE when it is under focus.

Tkinter Menu

Tkinter Menu

add_command(options)

It is used to add the Menu items to the menu.

add_radiobutton(options)

This method adds the radiobutton to the menu.

add_checkbutton(options)

This method is used to add the checkbuttons to the menu

add_cascade(options)

It is used to create a hierarchical menu to the parent menu by associating the given menu to the parent menu

add_seperator()

It is used to add the separator line to the menu

Tkinter Menu

delete(startindex, endindex)

It is used to delete the menu items exist in the specified range

index(item)

It is used to get the index of the specified menu item

insert_seperator(index)

It is used to insert a seperator at the specified index

Tkinter Menu

```
from Tkinter import *
root = Tk()
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Close", command=donothing)
filemenu.add_separator() filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)
editmenu.add_separator()
editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Select All", command=donothing)
menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)
root.config(menu=menubar)
root.mainloop()
```

```
from tkinter import *
top = Tk()
menubar = Menu(root)
menubar.add_command(label="Hello!", command=hello)
menubar.add_command(label="Quit!", command=top.quit)
top.config(menu=menubar)
top.mainloop()
```