Tkinter

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

Tkinter:

Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.

wxPython:

This is an open-source Python interface for wxWidgets GUI toolkit.

PyQt:

This is also a Python interface for a popular cross-platform Qt GUI library.

JPython:

JPython is a Python port for Java, which gives Python scripts seamless access to the Java class libraries on the local machine

There are many other interfaces available, which you can find them on the net.

Tkinter Programming

What is Tkinter?

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 â^'

- Import the Tkinter module.

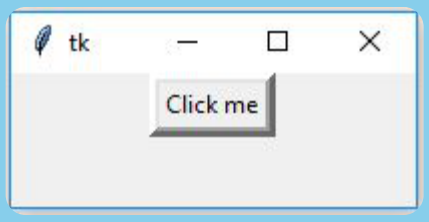
- Create the GUI application main window.
- Add one or more of the above-mentioned widgets to the GUI application.
- Enter the main event loop to take action against each event triggered by the user.

Tkinter widgets

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

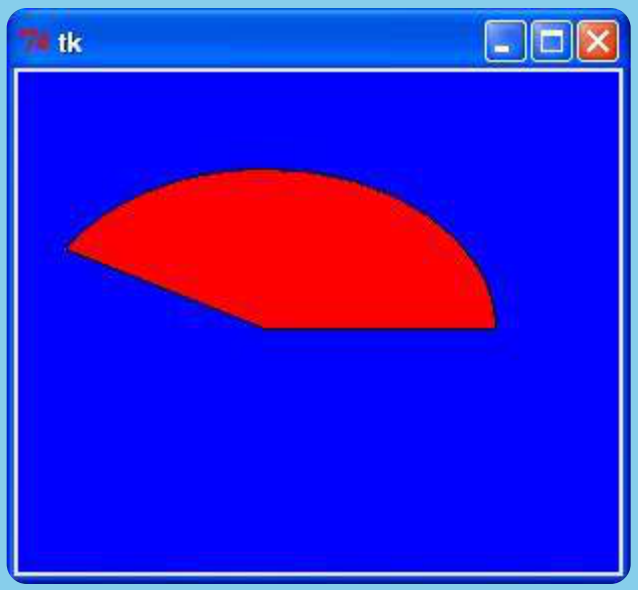
There are currently 15 types of widgets in Tkinter and 19 by further division. We present these widgets as well as a brief description in the following table-

1.Button:



The Button widget is used to display the buttons in your application.

2.Canvas

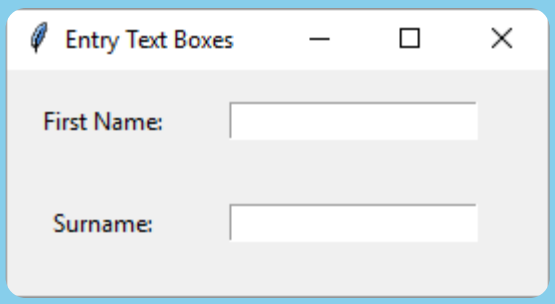


The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.

3.Checkbutton

The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.

4.Entry



The Entry widget is used to display a single-line text field for accepting values from a user.

5.Frame

The Frame widget is used as a container widget to organize other widgets.

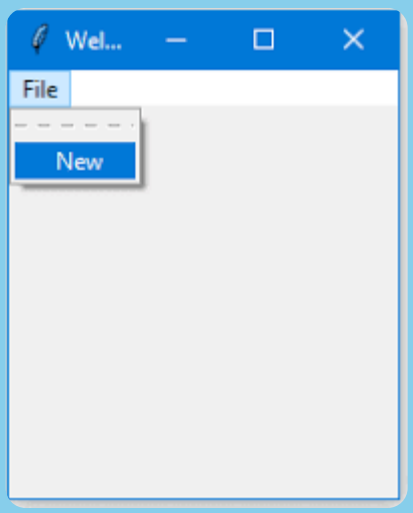
6.Label

The Label widget is used to provide a single-line caption for other widgets. It can also contain images.

7.Listbox

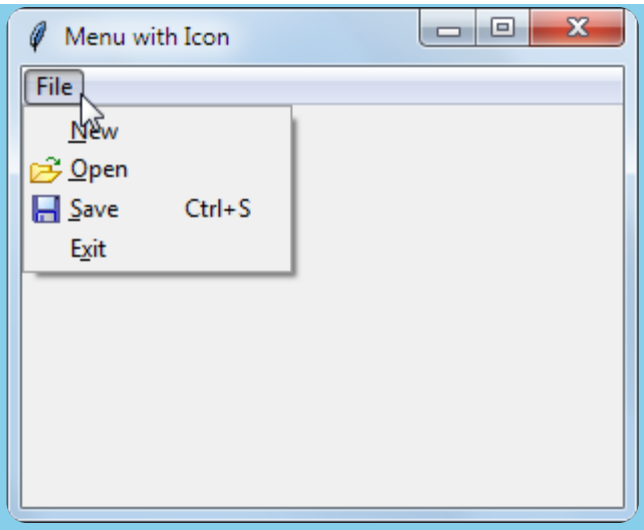
The Listbox widget is used to provide a list of options to a user.

8.Menubutton



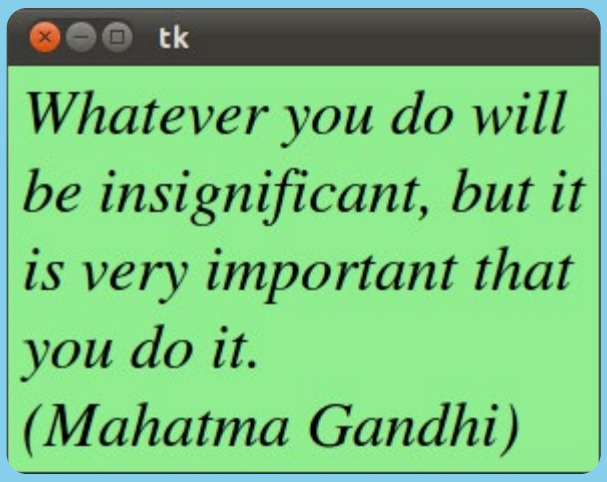
The Menubutton widget is used to display menus in your application.

9.Menu



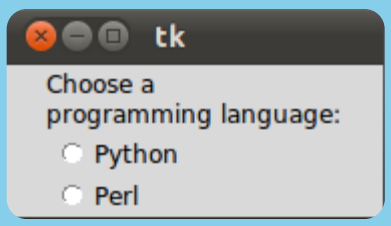
The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.

10.Message



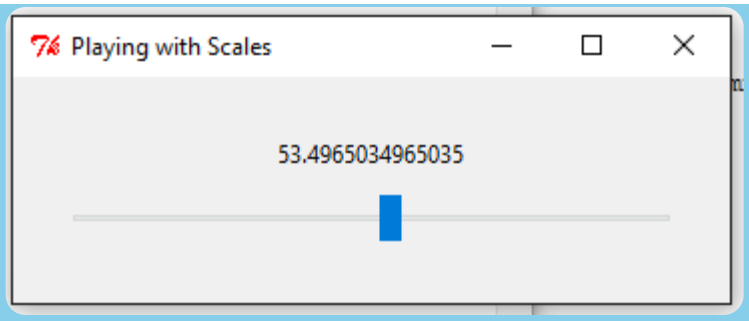
The Message widget is used to display multiline text fields for accepting values from a user.

11.Radiobutton



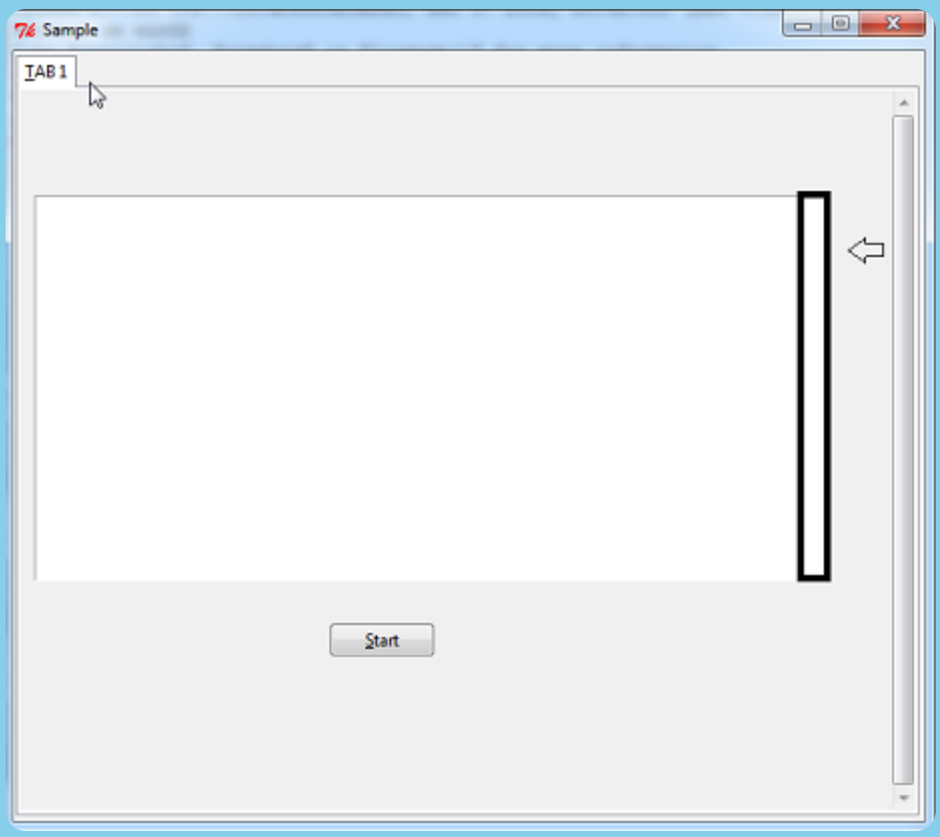
The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.

12.Scale



The Scale widget is used to provide a slider widget.

13.Scrollbar

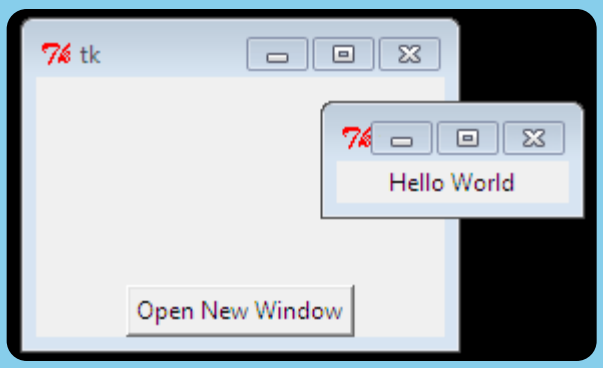


The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.

14.Text

The Text widget is used to display text in multiple lines.

15.Toplevel

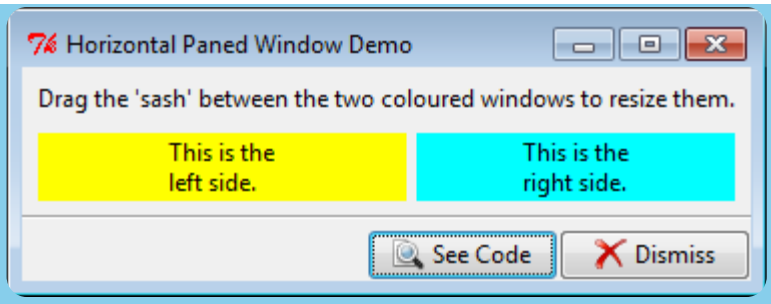


The Toplevel widget is used to provide a separate window container.

16.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.

17.PanedWindow

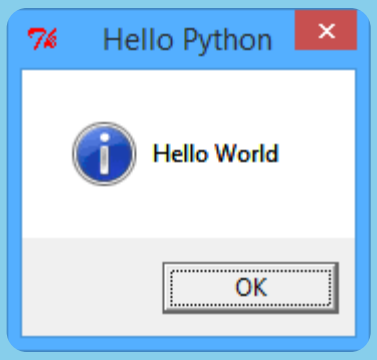


A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

18.LabelFrame

A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.

19.tkMessageBox



This module is used to display the message boxes in your applications.

Standard Attributes

Let us look at how some of the common attributes, such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Tkinter Button

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

Syntax:

Here is the simple syntax to create this widget-

b = Button (master, option=value, ...)

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground:

Background color when the button is under the cursor.

2.activeforeground:

Foreground color when the button is under the cursor.

3.bd:

Border width in pixels. Default is 2.

4.bg:

Normal background color.

5.command:

Function or method to be called when the button is clicked.

6.fg:

Normal foreground (text) color.

7.font:

Text font to be used for the button's label.

8.height:

Height of the button in text lines (for textual buttons) or pixels (for images).

9.highlightcolor:

The color of the focus highlight when the widget has focus.

10.image:

Image to be displayed on the button (instead of text).

11.justify:

How to show multiple text lines: LEFT to left-justify each line; CENTER to center them; or RIGHT to right-justify.

12.padx:

Additional padding left and right of the text.

13.pady:

Additional padding above and below the text.

14.relief:

Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.

15.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.

16.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 underlined.

17.width:

Width of the button in letters (if displaying text) or pixels (if displaying an image).

18.wraplength:

If this value is set to a positive number, the text lines will be wrapped to fit within this length.

Methods

---

flash():

Causes the button to flash several times between active and normal colors. Leaves the button in the state it was in originally. Ignored if the button is disabled.

invoke():

Calls the button's callback, and returns what that function returns. Has no effect if the button is disabled or there is no callback.

Tkinter Canvas

The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets or frames on a Canvas.

Syntax:

Here is the simple syntax to create this widget-

w = Canvas (master, option=value, ...)

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.confine:

If true (the default), the canvas cannot be scrolled outside of the scrollregion.

2.cursor:

Cursor used in the canvas like arrow, circle, dot etc.

3.bd:

Border width in pixels. Default is 2.

4.bg:

Normal background color.

5.relief:

Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.

6.scrollregion:

A tuple (w, n, e, s) that defines over how large an area the canvas can be scrolled, where w is the left side, n the top, e the right side, and s the bottom.

7.width

Size of the canvas in the X dimension.

8.height:

Size of the canvas in the Y dimension.

9.highlightcolor:

Color shown in the focus highlight.

10.xscrollincrement:

If you set this option to some positive dimension, the canvas can be positioned only on multiples of that distance, and the value will be used for scrolling by scrolling units, such as when the user clicks on the arrows at the ends of a scrollbar.

11.xscrollcommand:

If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar.

12.yscrollincrement:

Works like xscrollincrement, but governs vertical movement.

13.yscrollcommand:

If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar.

Standard items:

The Canvas widget can support the following standard items-

1.arc:

Creates an arc item, which can be a chord, a pieslice or a simple arc.

coord = 10, 50, 240, 210

arc = canvas.create_arc(coord, start=0, extent=150, fill="blue")

2.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)

3.line:

Creates a line item.

line = canvas.create_line(x0, y0, x1, y1, ..., xn, yn, options)

4.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.

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

5.polygen:

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

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

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.

You can also display images in place of text.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground

Background color when the checkbutton is under the cursor.

2.activeforeground

Foreground color when the checkbutton is under the cursor.

3.bg

The normal background color displayed behind the label and indicator.

4.bitmap

To display a monochrome image on a button.

5.bd

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

6.command

A procedure to be called every time the user changes the state of this checkbutton.

7.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.

8.disabledforeground

The foreground color used to render the text of a disabled checkbutton. The default is a stippled version of the default foreground color.

9.font

The font used for the text.

10.fg

The color used to render the text.

11.height

The number of lines of text on the checkbutton. Default is 1.

12.highlightcolor

The color of the focus highlight when the checkbutton has the focus.

13.image

To display a graphic image on the button.

14.justify

If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.

15.offvalue

Normally, a checkbutton's associated control variable will be set to 0 when it is cleared (off). You can supply an alternate value for the off state by setting offvalue to that value.

16.onvalue

Normally, a checkbutton's associated control variable will be set to 1 when it is set (on). You can supply an alternate value for the on state by setting onvalue to that value.

17.padx

How much space to leave to the left and right of the checkbutton and text. Default is 1 pixel.

18.pady

How much space to leave above and below the checkbutton and text. Default is 1 pixel.

19.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

20.selectcolor

The color of the checkbutton when it is set. Default is selectcolor="red".

21.selectimage

If you set this option to an image, that image will appear in the checkbutton when it is set.

22.state

The default is state=NORMAL, but you can use state=DISABLED to gray out the control and make it unresponsive. If the cursor is currently over the checkbutton, the state is ACTIVE.

23.text

The label displayed next to the checkbutton. Use newlines ("\n") to display multiple lines of text.

24.underline

With the default value of -1, none of the characters of the text label are underlined.

Set this option to the index of a character in the text (counting from zero) to underline that character.

25.variable

The control variable that tracks the current state of the checkbutton. Normally this variable is an IntVar, and 0 means cleared and 1 means set, but see the offvalue and onvalue options above.

26.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.

27.wraplength

Normally, lines are not wrapped. You can set this option to a number of characters and all lines will be broken into pieces no longer than that number.

Methods

---

deselect()

Clears (turns off) the checkbutton.

flash()

Flashes the checkbutton a few times between its active and normal colors, but leaves it the way it started.

invoke()

You can call this method to get the same actions that would occur if the user clicked on the checkbutton to change its state.

select()

Sets (turns on) the checkbutton.

toggle()

Clears the checkbutton if set, sets it if cleared.

Tkinter Radio Button

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. You can use the Tab key to switch from one radionbutton to another.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground

The background color when the mouse is over the radiobutton.

2.activeforeground

The foreground color when the mouse is over the radiobutton.

3.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.

4.bg

The normal background color behind the indicator and label.

5.bitmap

To display a monochrome image on a radiobutton, set this option to a bitmap.

6.borderwidth

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

7.command

A procedure to be called every time the user changes the state of this radiobutton.

8.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.

9.font

The font used for the text.

10.fg

The color used to render the text.

11.height

The number of lines (not pixels) of text on the radiobutton. Default is 1.

12.highlightbackground

The color of the focus highlight when the radiobutton does not have focus.

13.highlightcolor

The color of the focus highlight when the radiobutton has the focus.

14.image

To display a graphic image instead of text for this radiobutton, set this option to an

image object.

15.justify

If the text contains multiple lines, this option controls how the text is justified: CENTER (the default), LEFT, or RIGHT.

16.padx

How much space to leave to the left and right of the radiobutton and text. Default is 1.

17.pady

How much space to leave above and below the radiobutton and text. Default is 1.

18.relief

Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.

19.selectcolor

The color of the radiobutton when it is set. Default is red.

20.selectimage

If you are using the image option to display a graphic instead of text when the radiobutton is cleared, you can set the selectimage option to a different image that will be displayed when the radiobutton is set.

21.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.

22.text

The label displayed next to the radiobutton. Use newlines ("\n") to display multiple lines of text.

23.textvariable

To slave the text displayed in a label widget to a control variable of class StringVar, set this option to that variable.

24.underline

You can display an underline (_) below the nth letter of the text, counting from 0,

by setting this option to n. The default is underline=-1, which means no underlining.

25.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 aStringVar, give each radiobutton a different string value option.

26.variable

The control variable that this radiobutton shares with the other radiobuttons in the group. This can be either an IntVar or a StringVar.

27.width

Width of the label in characters (not pixels!). If this option is not set, the label will be sized to fit its contents.

28.wraplength

You can limit the number of characters in each line by setting this option to the desired number. The default value, 0, means that lines will be broken only at newlines.

Method

deselect()

Clears (turns off) the radiobutton

flash()

Flashes the radiobutton a few times between its active and normal colors, but leaves it the way it started.

invoke()

You can call this method to get the same actions that would occur if the user clicked on the radiobutton to change its state.

select()

Sets (turns on) the radiobutton.

Syntax:

Here is the simple syntax to create this widget-

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

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.command

A procedure to be called every time the user changes the state of this checkbutton.

4.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

5.font

The font used for the text.

6.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.

7.fg

The color used to render the text.

8.highlightcolor

The color of the focus highlight when the checkbutton has the focus.

9.justify

If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.

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.selectbackground

The background color to use displaying selected text.

12.selectborderwidth

The width of the border to use around selected text. The default is one pixel.

13.selectforeground

The foreground (text) color of selected text.

14.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="*".

15.state

The default is state=NORMAL, but you can use state=DISABLED to gray out the

control and make it unresponsive. If the cursor is currently over the checkbutton, the state is ACTIVE.

16.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.

17.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.

18.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.

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.

index (index)

Shift the contents of the entry so that the character at the given index is the leftmost visible character. Has no effect if the text fits entirely within the entry.

insert (index, s)

Inserts string s before the character at the given index.

select_adjust (index)

This method is used to make sure that the selection includes the character at the specified index.

select_clear()

Clears the selection. If there isn't currently a selection, has no effect.

select_from (index)

Sets the ANCHOR index position to the character selected by index, and selects that character.

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.

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 Text

Text widgets provide advanced capabilities that allow you to edit a multiline text and format the way it has to be displayed, such as changing its color and font.

You can also use elegant structures like tabs and marks to locate specific sections of the text, and apply changes to those areas. Moreover, you can embed windows and images in the text because this widget was designed to handle both plain and formatted text.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.bg

The default background color of the text widget.

2.bd

The width of the border around the text widget. Default is 2 pixels.

3.cursor

The cursor that will appear when the mouse is over the text widget.

4.exportselection

Normally, text selected within a text widget is exported to be the selection in the window manager. Set exportselection=0 if you don't want that behavior.

5.font

The default font for text inserted into the widget.

6.fg

The color used for text (and bitmaps) within the widget. You can change the color for tagged regions; this option is just the default.

7.height

The height of the widget in lines (not pixels!), measured according to the current font size.

8.highlightbackground

The color of the focus highlight when the text widget does not have focus.

9.highlightcolor

The color of the focus highlight when the text widget has the focus.

10.highlightthickness

The thickness of the focus highlight. Default is 1. Set highlightthickness=0 to suppress display of the focus highlight.

11.insertbackground

The color of the insertion cursor. Default is black.

12.insertborderwidth

Size of the 3-D border around the insertion cursor. Default is 0.

13.insertofftime

The number of milliseconds the insertion cursor is off during its blink cycle. Set this option to zero to suppress blinking. Default is 300.

14.insertontime

The number of milliseconds the insertion cursor is on during its blink cycle. Default is 600.

15.insertwidth

Width of the insertion cursor (its height is determined by the tallest item in its line). Default is 2 pixels.

16.padx

The size of the internal padding added to the left and right of the text area. Default is one pixel.

17.pady

The size of the internal padding added above and below the text area. Default is one pixel.

18.relief

The 3-D appearance of the text widget. Default is relief=SUNKEN.

19.selectbackground

The background color to use displaying selected text.

20.selectborderwidth

The width of the border to use around selected text.

21.spacing1

This option specifies how much extra vertical space is put above each line of text. If a line wraps, this space is added only before the first line it occupies on the display. Default is 0.

22.spacing2

This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. Default is 0.

23.spacing3

This option specifies how much extra vertical space is added below each line of text. If a line wraps, this space is added only after the last line it occupies on the display. Default is 0.

24.state

Normally, text widgets respond to keyboard and mouse events; set

state=NORMAL to get this behavior. If you set state=DISABLED, the text widget will not respond, and you won't be able to modify its contents programmatically either.

25.tabs

This option controls how tab characters position text.

26.width

The width of the widget in characters (not pixels!), measured according to the current font size.

27.wrap

This option controls the display of lines that are too wide. Set wrap=WORD and it will break the line after the last word that will fit. With the default behavior, wrap=CHAR, any line that gets too long will be broken at any character.

Methods

28.xscrollcommand

To make the text widget horizontally scrollable, set this option to the set() method of the horizontal scrollbar.

29.yscrollcommand

To make the text widget vertically scrollable, set this option to the set() method of the vertical scrollbar.

delete(startindex [,endindex])

This method deletes a specific character or a range of text.

get(startindex [,endindex])

This method returns a specific character or a range of text.

index(index)

Returns the absolute value of an index based on the given index.

insert(index [,string]...)

This method inserts strings at the specified index location.

see(index)

This method returns true if the text located at the index position is visible.

Text widgets support three distinct helper structures: Marks, Tabs, and Indexes-

Marks are used to bookmark positions between two characters within a given text. We have the following methods available when handling marks:

index(mark)

Returns the line and column location of a specific mark.

mark_gravity(mark [,gravity])

Returns the gravity of the given mark. If the second argument is provided, the gravity is set for the given mark.

mark_names()

Returns all marks from the Text widget.

mark_set(mark, index)

Informs a new position to the given mark.

mark_unset(mark)

Removes the given mark from the Text widget.

Tags are used to associate names to regions of text which makes easy the task of modifying the display settings of specific text areas. Tags are also used to bind event callbacks to specific ranges of text.

Following are the available methods for handling tabs-

tag_add(tagname, startindex[,endindex] ...)

This method tags either the position defined by startindex, or a range delimited by the positions startindex and endindex.

tag_config

You can use this method to configure the tag properties, which include, justify(center, left, or right), tabs(this property has the same functionality of the Text widget tabs's property), and underline(used to underline the tagged text).

tag_delete(tagname)

This method is used to delete and remove a given tag.

tag_remove(tagname [,startindex[.endindex]] ...)

After applying this method, the given tag is removed from the provided area without deleting the actual tag definition.

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.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

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.height

The vertical dimension of the new frame.

5.highlightbackground

Color of the focus highlight when the frame does not have focus.

6.highlightcolor

Color shown in the focus highlight when the frame has the focus.

7.highlightthickness

Thickness of the focus highlight.

8.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

9.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 Listbox

The Listbox widget is used to display a list of items from which a user can select a number of items

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

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

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

4.font

The font used for the text in the listbox.

5.fg

The color used for the text in the listbox.

6.height

Number of lines (not pixels!) shown in the listbox. Default is 10.

7.highlightcolor

Color shown in the focus highlight when the widget has the focus

8.highlightthickness

Thickness of the focus highlight.

9.relief

Selects three-dimensional border shading effects. The default is SUNKEN.

10.selectbackground

The background color to use displaying selected text.

11.width

The width of the widget in characters. The default is 20

12.xscrollcommand

If you want to allow the user to scroll the listbox horizontally, you can link your listbox widget to a horizontal scrollbar.

13.yscrollcommand

If you want to allow the user to scroll the listbox vertically, you can link your listbox widget to a vertical scrollbar.

14.selectmode

Determines how many items can be selected, and how mouse drags affect the selection:

BROWSE: Normally, you can only select one line out of a listbox. If you click on an item and then drag to a different line, the selection will follow the mouse. This is the default.

- **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.

Methods

---

activate (index)

Selects the line specifies by the given index.

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.

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.

index (i)

If possible, positions the visible part of the listbox so that the line containing index i is at the top of the widget.

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.

see (index)

Adjust the position of the listbox so that the line referred to by index is visible.

size()

Returns the number of lines in the listbox.

xview()

To make the listbox horizontally scrollable, set the command option of the associated horizontal scrollbar to this method.

xview_moveto (fraction)

Scroll the listbox so that the leftmost fraction of the width of its longest line is outside the left side of the listbox. Fraction is in the range [0,1].

xview_scroll (number, what)

Scrolls the listbox horizontally. For the what argument, use either UNITS to scroll by characters, or PAGES to scroll by pages, that is, by the width of the listbox. The number argument tells how many to scroll.

yview()

To make the listbox vertically scrollable, set the command option of the associated vertical scrollbar to this method.

yview_moveto (fraction)

Scroll the listbox so that the top fraction of the width of its longest line is outside the left side of the listbox. Fraction is in the range [0,1].

yview_scroll (number, what)

Scrolls the listbox vertically. For the what argument, use either UNITS to scroll by lines, or PAGES to scroll by pages, that is, by the height of the listbox. The number argument tells how many to scroll.

Tkinter tkMessageBox

The tkMessageBox 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.

Syntax:

Here is the simple syntax to create this widget-

tkMessageBox.FunctionName(title, message [, options])

Parameters:

FunctionName:

This is the name of the appropriate message box function.

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.

title:

This is the text to be displayed in the title bar of a message box.

message:

This is the text to be displayed as a message.

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

- showinfo()
- showwarning()
- showerror ()
- askquestion()
- askokcancel()
- askyesno ()

askretrycancel ()

Tkinter Menubutton

A menubutton is the part of a drop-down menu that stays on the screen all the time. Every menubutton is associated with a Menu widget that can display the choices for that menubutton when the user clicks on it.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground

The background color when the mouse is over the menubutton.

2.activeforeground

The foreground color when the mouse is over the menubutton.

3.anchor

This options controls where the text is positioned if the widget has more space than the text needs. The default is anchor=CENTER, which centers the text.

4.bg

The normal background color displayed behind the label and indicator.

5.bitmap

To display a bitmap on the menubutton, set this option to a bitmap name.

6.bd

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

7.cursor

The cursor that appears when the mouse is over this menubutton.

8.direction

Set direction=LEFT to display the menu to the left of the button; use direction=RIGHT to display the menu to the right of the button; or use direction='above' to place the menu above the button

9.disabledforeground

The foreground color shown on this menubutton when it is disabled.

10.fg

The foreground color when the mouse is not over the menubutton.

11.height

The height of the menubutton in lines of text (not pixels!). The default is to fit the menubutton's size to its contents.

12.highlightcolor

Color shown in the focus highlight when the widget has the focus.

13.image

To display an image on this menubutton,

14.justify

This option controls where the text is located when the text doesn't fill the

menubutton: use justify=LEFT to left-justify the text (this is the default); use justify=CENTER to center it, or justify=RIGHT to right-justify.

15.menu

To associate the menubutton with a set of choices, set this option to the Menu object containing those choices. That menu object must have been created by passing the associated menubutton to the constructor as its first argument.

16.padx

How much space to leave to the left and right of the text of the menubutton. Default is 1.

17.pady

How much space to leave above and below the text of the menubutton. Default is 1.

18.relief

Selects three-dimensional border shading effects. The default is RAISED.

19.state

Normally, menubuttons respond to the mouse. Set state=DISABLED to gray out the menubutton and make it unresponsive.

20.text

To display text on the menubutton, set this option to the string containing the desired text. Newlines ("\n") within the string will cause line breaks.

21.textvariable

You can associate a control variable of class StringVar with this menubutton. Setting that control variable will change the displayed text.

22.underline

Normally, no underline appears under the text on the menubutton. To underline one of the characters, set this option to the index of that character.

23.width

The width of the widget in characters. The default is 20.

24.wraplength

Normally, lines are not wrapped. You can set this option to a number of characters

and all lines will be broken into pieces no longer than that number.

Tkinter Menu

The goal of this widget is to allow us to create all kinds of menus that can be used by our applications. The core functionality provides ways to create three menu types: pop-up, toplevel and pull-down.

It is also possible to use other extended widgets to implement new types of menus, such as the OptionMenu widget, which implements a special type that generates a pop-up list of items within a selection.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground

The background color that will appear on a choice when it is under the mouse.

2.activeborderwidth

Specifies the width of a border drawn around a choice when it is under the mouse.

Default is 1 pixel.

3.activeforeground

The foreground color that will appear on a choice when it is under the mouse.

4.bg

The background color for choices not under the mouse.

5.bd

The width of the border around all the choices. Default is 1.

6.cursor

The cursor that appears when the mouse is over the choices, but only when the menu has been torn off.

7.disabledforeground

The color of the text for items whose state is DISABLED.

8.font

The default font for textual choices.

9.fg

The foreground color used for choices not under the mouse.

10.postcommand

You can set this option to a procedure, and that procedure will be called every time someone brings up this menu.

11.relief

The default 3-D effect for menus is relief=RAISED.

12.image

To display an image on this menubutton.

13.selectcolor

Specifies the color displayed in checkbuttons and radiobuttons when they are selected.

14.tearoff

Normally, a menu can be torn off, the first position (position 0) in the list of choices is occupied by the tear-off element, and the additional choices are added starting at position 1. If you set tearoff=0, the menu will not have a tear-off feature, and choices will be added starting at position 0.

15.title

Normally, the title of a tear-off menu window will be the same as the text of the menubutton or cascade that lead to this menu. If you want to change the title of that window, set the title option to that string.

Method

---

add_command (options)

Adds a menu item to the menu.

add_radiobutton(options)

Creates a radio button menu item.

add_checkbutton(options)

Creates a check button menu item.

add_cascade(options)

Creates a new hierarchical menu by associating a given menu to a parent menu

add_separator()

Adds a separator line to the menu.

add(type, options)

Adds a specific type of menu item to the menu.

delete(startindex [, endindex])

Deletes the menu items ranging from startindex to endindex.

entryconfig(index, options)

Allows you to modify a menu item, which is identified by the index, and change its options.

index(item)

Returns the index number of the given menu item label.

insert_separator (index)

Insert a new separator at the position specified by index.

invoke (index)

Calls the command callback associated with the choice at position index. If a checkbutton, its state is toggled between set and cleared; if a radiobutton, that choice is set.

type (index)

Returns the type of the choice specified by index: either "cascade", "checkbutton", "command", "radiobutton", "separator", or "tearoff".

Tkinter 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.

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.activebackground

h4>

The color of the slider and arrowheads when the mouse is over them.

2.bg

The color of the slider and arrowheads when the mouse is not over them.

3.bd

The width of the 3-d borders around the entire perimeter of the trough, and also the width of the 3-d effects on the arrowheads and slider. Default is no border around the trough, and a 2-pixel border around the arrowheads and slider.

4.command

A procedure to be called whenever the scrollbar is moved.

5.cursor

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

6.disabledbackground

The background color to use when the widget is disabled.

7.disabledforeground

The text color to use when the widget is disabled.

8.fg

Text color.

9.font

The font to use in this widget.

10.format

Format string. No default value.

11.from

The minimum value. Used together with to to limit the spinbox range.

12.justify

Default is LEFT

13.relief

Default is SUNKEN.

14.repeatdelay

Together with repeatinterval, this option controls button auto- repeat. Both values are given in milliseconds.

15.repeatinterval

See repeatdelay.

16.state

One of NORMAL, DISABLED, or "readonly". Default is NORMAL.

17.textvariable

No default value.

18.to

See from.

19.validate

Validation mode. Default is NONE.

20.validatecommand

Validation callback. No default value.

21.values

A tuple containing valid values for this widget. Overrides from/to/increment.

22.vcmd

Same as validatecommand.

23.width

Widget width, in character units. Default is 20.

24.wrap

If true, the up and down buttons will wrap around.

25.xscrollcommand

Used to connect a spinbox field to a horizontal scrollbar. This option should be set to the set method of the corresponding scrollbar.

Methods

---

delete(startindex [,endindex])

This method deletes a specific character or a range of text.

get(startindex [,endindex])

This method returns a specific character or a range of text.

identify(x, y)

Identifies the widget element at the given location.

index(index)

Returns the absolute value of an index based on the given index.

insert(index [,string]...)

This method inserts strings at the specified index location.

invoke(element)

Invokes a spinbox button.

Tkinter PanedWindow

A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

Each pane contains one widget and each pair of panes is separated by a moveable (via mouse movements) sash. Moving a sash causes the widgets on either side of the sash to be resized

Syntax:

Here is the simple syntax to create this widget-

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

Parameters:

master:

This represents the parent window.

options:

Here is the list of most commonly used options for this widget in the next page given below. These options can be used as key-value pairs separated by commas.

Options available and their description

1.bg

The color of the slider and arrowheads when the mouse is not over them.

2.bd

The width of the 3-d borders around the entire perimeter of the trough, and also the width of the 3-d effects on the arrowheads and slider. Default is no border around the trough, and a 2-pixel border around the arrowheads and slider.

3.borderwidth

Default is 2.

4.cursor

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

5.handlepad

Default is 8.

6.handlesize

Default is 8.

7.height

No default value.

8.orient

Default is HORIZONTAL.

9.relief

Default is FLAT.

10.sashcursor

No default value.

11.sashrelief

Default is RAISED.

12.sashwidth

Default is 2.

13.showhandle

No default value

14.width

No default value

Methods

---

add(child, options)

Adds a child window to the paned window.

get(startindex [,endindex])

This method returns a specific character or a range of text.

config(options)

Modifies one or more widget options. If no options are given, the method returns a dictionary containing all current option values.

Standard Attributes

Let us look at how some of the common attributes, such as sizes, colors and fonts are specified.

- Dimensions
- Colors
- Fonts
- Anchors
- Relief styles
- Bitmaps
- Cursors

Tkinter Dimensions

Various lengths, widths, and other dimensions of widgets can be described in many different units.

- If you set a dimension to an integer, it is assumed to be in pixels.
- You can specify units by setting a dimension to a string containing a number followed by.

Character and their description

1.c

Centimeters

2.i

Inches

3.m

Millimeters

4.p

Printer's points (about 1/72")

Length options

Tkinter expresses a length as an integer number of pixels. Here is the list of common length options-

- **borderwidth:** Width of the border which gives a three-dimensional look to the widget.
- **highlightthickness:** Width of the highlight rectangle when the widget has focus .
- **padX padY:** Extra space the widget requests from its layout manager beyond the minimum the widget needs to display its contents in the x and y directions.
- **selectborderwidth:**Width of the three-dimentional border around selected items of the widget.
- **wraplength:**Maximum line length for widgets that perform word wrapping.
- **height:** Desired height of the widget; must be greater than or equal to 1.
- **underline:**Index of the character to underline in the widget's text (0 is the first character, 1 the second one, and so on).
- **width:**Desired width of the widget.

Tkinter Colors

Tkinter represents colors with strings. There are two general ways to specify colors in Tkinter-

- You can use a string specifying the proportion of red, green and blue in hexadecimal digits. For example, "#fff" is white, "#000000" is black, "#000fff000" is pure green, and "#00ffff" is pure cyan (green plus blue).
- You can also use any locally defined standard color name. The colors "white", "black", "red", "green", "blue", "cyan", "yellow", and "magenta" will always be available.

Color Options

- **activebackground:** Background color for the widget when the widget is active.
- **activeforeground:** Foreground color for the widget when the widget is active.
- **background:** Background color for the widget. This can also be represented as bg.
- **disabledforeground:**Foreground color for the widget when the widget is disabled.
- **foreground:**Foreground color for the widget. This can also be represented as fg.
- **highlightbackground:**Background color of the highlight region when the

widget has focus.
- **highlightcolor:** Foreground color of the highlight region when the widget has focus.
- **selectbackground:**Background color for the selected items of the widget.
- **selectforeground:** Foreground color for the selected items of the widget.

Tkinter Fonts

There may be up to three ways to specify type style.

Simple Tuple Fonts

As a tuple whose first element is the font family, followed by a size in points, optionally followed by a string containing one or more of the style modifiers bold, italic, underline and overstrike.

Example:

- ("Helvetica", "16") for a 16-point Helvetica regular.
- ("Times", "24", "bold italic") for a 24-point Times bold italic.

Font object Fonts

You can create a "font object" by importing the tkFont module and using its Font class constructor âˆ’

import tkFont

font = tkFont.Font (option, ...)

Here is the list of options-

- **family:** The font family name as a string.
- **size:** The font height as an integer in points. To get a font n pixels high, use -n.
- **weight:** "bold" for boldface, "normal" for regular weight.
- **slant:** "italic" for italic, "roman" for unslanted.
- **underline:** 1 for underlined text, 0 for normal.
- **overstrike:** 1 for overstruck text, 0 for normal.

Example

helv36 = tkFont.Font(family="Helvetica",size=36,weight="bold")

Tkinter Anchors

Anchors are used to define where text is positioned relative to a reference point. Here is list of possible constants, which can be used for Anchor attribute.

- NW
- N
- NE
- W
- CENTER
- E
- SW
- S
- SE

For example, if you use CENTER as a text anchor, the text will be centered horizontally and vertically around the reference point.

Anchor NW will position the text so that the reference point coincides with the northwest (top left) corner of the box containing the text.

Anchor W will center the text vertically around the reference point, with the left edge of the text box passing through that point, and so on.

If you create a small widget inside a large frame and use the anchor=SE option, the widget will be placed in the bottom right corner of the frame. If you used anchor=N instead, the widget would be centered along the top edge.

Tkinter Relief styles

The relief style of a widget refers to certain simulated 3-D effects around the outside of the widget.

Here is list of possible constants which can be used for relief attribute-

- FLAT
- RAISED
- SUNKEN
- GROOVE
- RIDGE

Tkinter Bitmaps

This attribute to displays a bitmap. There are following type of bitmaps available-

- "error"
- "gray75"
- "gray50"
- "gray12"
- "hourglass"
- "info"
- "questhead"
- "question"
- "warning"

Tkinter Cursors

Python Tkinter supports quite a number of different mouse cursors available. The exact graphic may vary according to your operating system.

Here is the list of interesting ones-

- "arrow"
- "circle"
- "clock"
- "cross"
- "dotbox"
- "exchange"
- "fleur"
- "heart"
- "man"
- "mouse"
- "pirate"
- "plus"
- "shuttle"
- "sizing"
- "spider"
- "spraycan"
- "star"
- "target"
- "tcross"
- "trek"
- "watch"

Geometry Management

All Tkinter widgets have access to the 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.

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

Tkinter pack() Method

This geometry manager organizes widgets in blocks before placing them in the parent widget.

Syntax

widget.pack(pack_options)

Here is the list of possible options-

- **expand:** When set to true, widget expands to fill any space not otherwise used in widget's parent.
- **fill:** Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).
- **side:** Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.

Tkinter grid() Method

This geometry manager organizes widgets in a table-like structure in the parent widget.

Syntax

widget.grid(grid_options)

Here is the list of possible options-

- **column :** The column to put widget in; default 0 (leftmost column).
- **columnspan:** How many columns widgetoccupies; default 1.
- **ipadx, ipady :**How many pixels to pad widget, horizontally and vertically, outside v's borders.
- **row:** The row to put widget in; default the first row that is still empty.
- **rowspan :** How many rowswidget occupies; default 1.
- **sticky :** What to do if the cell is larger than widget. By default, with sticky='', widget is centered in its cell. sticky may be the string concatenation of zero or more of N, E, S, W, NE, NW, SE, and SW, compass directions indicating the sides and corners of the cell to which widget sticks.
- **padx, pady :** How many pixels to pad widget, horizontally and vertically, outside v's borders.

Tkinter place() Method

This geometry manager organizes widgets by placing them in a specific position in the parent widget.

Syntax

widget.place(place_options)

Here is the list of possible options-

- **anchor :** The exact spot of widget other options refer to: may be N, E, S, W, NE, NW, SE, or SW, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)
- **bordermode :** INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.
- **height, width :** Height and width in pixels.
- **relheight, relwidth :**Height and width as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- **relx, rely :** Horizontal and vertical offset as a float between 0.0 and 1.0, as a fraction of the height and width of the parent widget.
- **x, y :** Horizontal and vertical offset in pixels.