Tkinter GUI

- *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:

  - Import the *Tkinter* module.

  - Create the GUI application main window.

  - Add one or more widgets to the GUI application.

  - Enter the main event loop to take action against each event triggered by the user.

Geometry Management

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

Button Widget
Canvas Widget
Checkbutton Widget
Entry Widget
Frame Widget
Label Widget
Listbox Widget
Menu Widget
Menubutton Widget
Radiobutton Widget
Scale Widget
Scrollbar Widget
Text Widget

Button widget

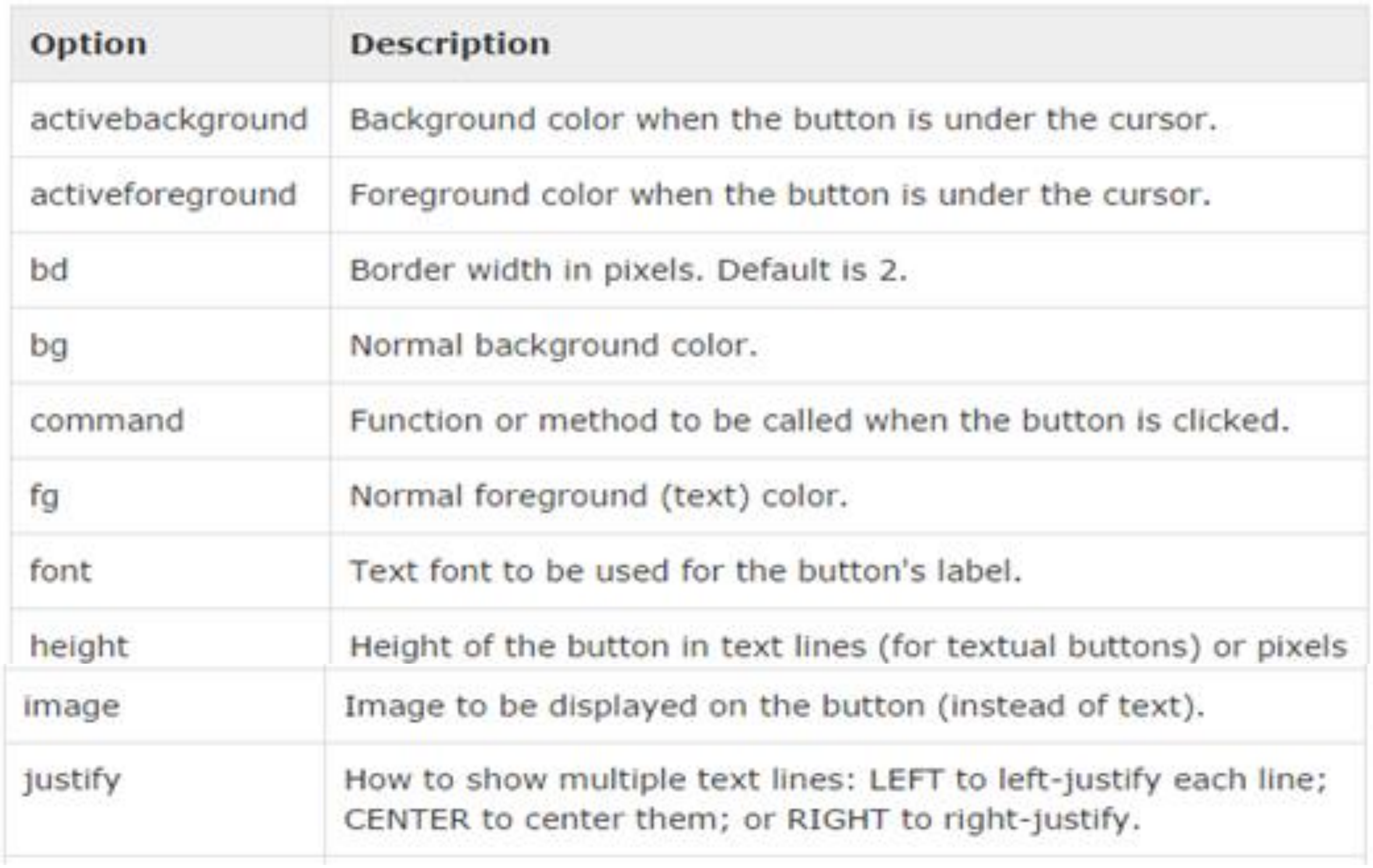
- The Button widget is used to add buttons in a Python application. These buttons can display text or images.
- You can attach a function or a method to a button which is called automatically when you click the button.

- Syntax:

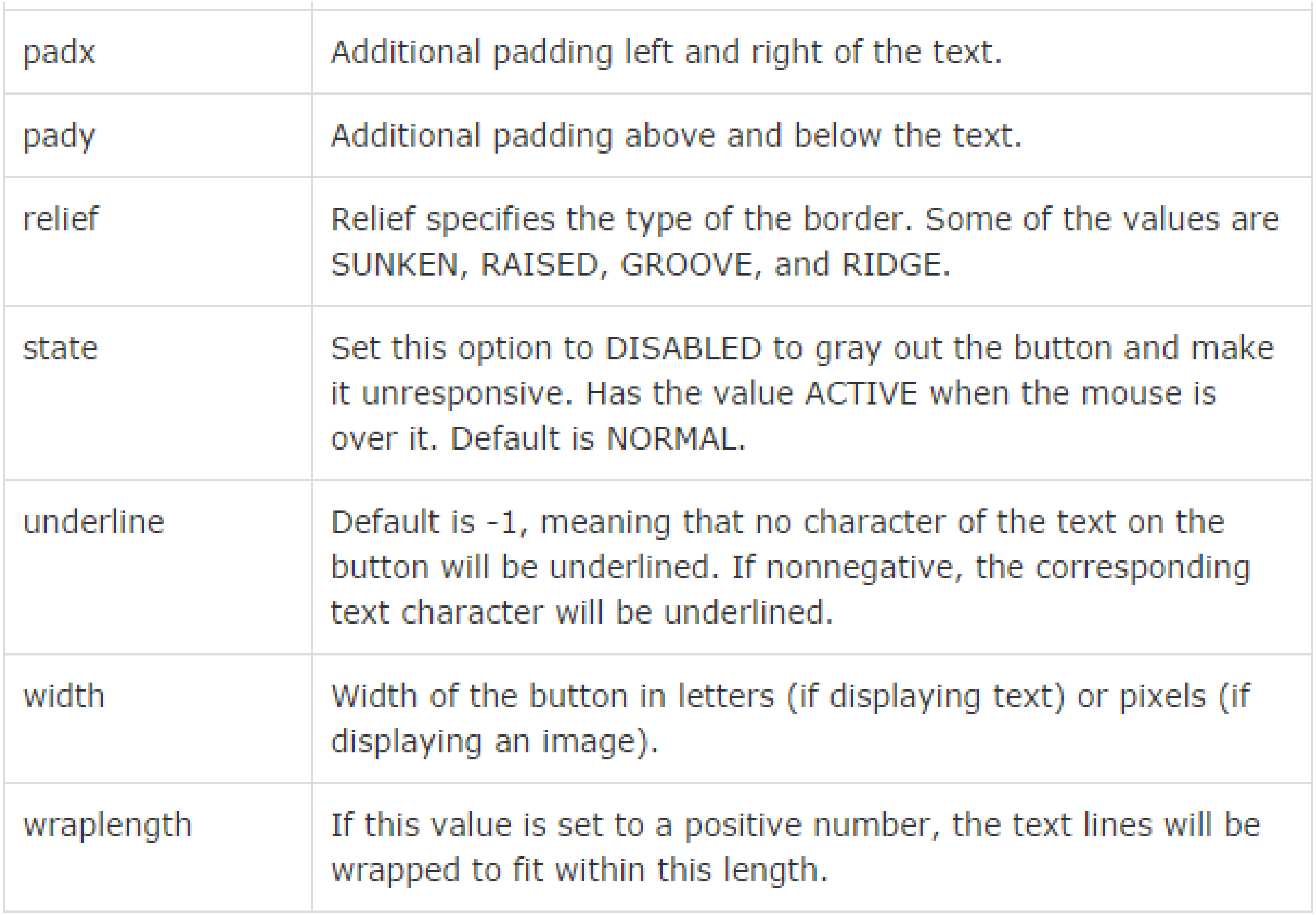
```
w = Button ( master, option=value, ... )
```

master: This represents the parent window.

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

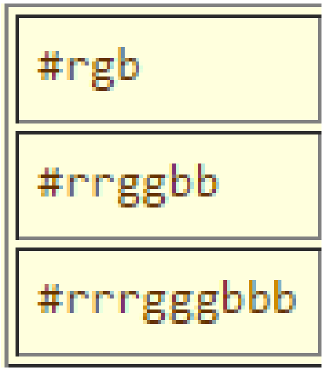


Option	Description
activebackground	Background color when the button is under the cursor.
activeforeground	Foreground color when the button is under the cursor.
bd	Border width in pixels. Default is 2.
bg	Normal background color.
command	Function or method to be called when the button is clicked.
fg	Normal foreground (text) color.
font	Text font to be used for the button's label.
height	Height of the button in text lines (for textual buttons) or pixels
image	Image to be displayed on the button (instead of text).
justify	How to show multiple text lines: LEFT to left-justify each line; CENTER to center them; or RIGHT to right-justify.



padx	Additional padding left and right of the text.
pady	Additional padding above and below the text.
relief	Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.
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 underlined.
width	Width of the button in letters (if displaying text) or pixels (if displaying an image).
wraplength	If this value is set to a positive number, the text lines will be wrapped to fit within this length.

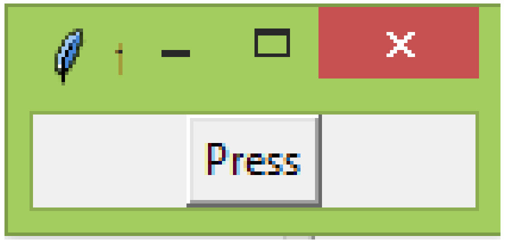
- There are two general ways to specify colors in *Tkinter*.
- You can use a string specifying the proportion of red, green, and blue in hexadecimal digits:



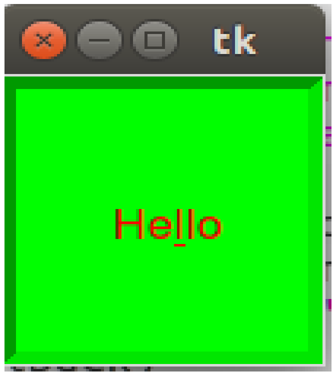
#rgb
#rrggbb
#rrrgggbbb

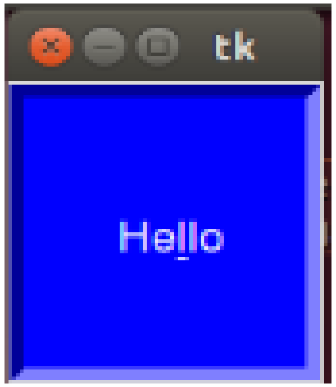
  For example, '#fff' is white, '#000000' is black, '#000fff000' is pure green, and '#00ffff' is pure cyan (green plus blue).
- You can also use any locally defined standard color name. The colors 'white', 'black', 'red', 'green', 'blue', 'cyan', 'yellow', and 'magenta' will always be available. Other names may work, depending on your local installation.

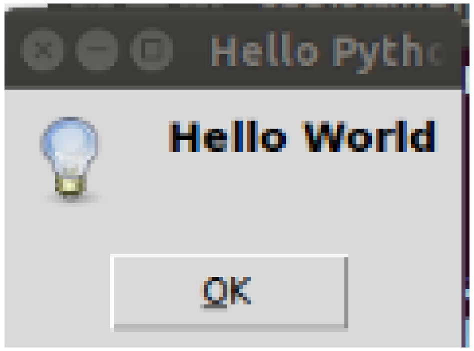
```
import tkinter as tk
root = tk.Tk()
btn = tk.Button(root, text="Press”)
btn.pack()
root.mainloop()
```



```
import tkinter
from tkinter import messagebox

root = tkinter.Tk()

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

B=tkinter.Button (root, text ="Hello", activebackground='blue',
    activeforeground='#fff', bd=5, fg='red', bg='#000fff000', height=3,
    font='helvetica', padx=30, pady=20, relief='sunken', underline=2,
    command = helloCallBack)

B.pack()
root.mainloop()
```







Hello World

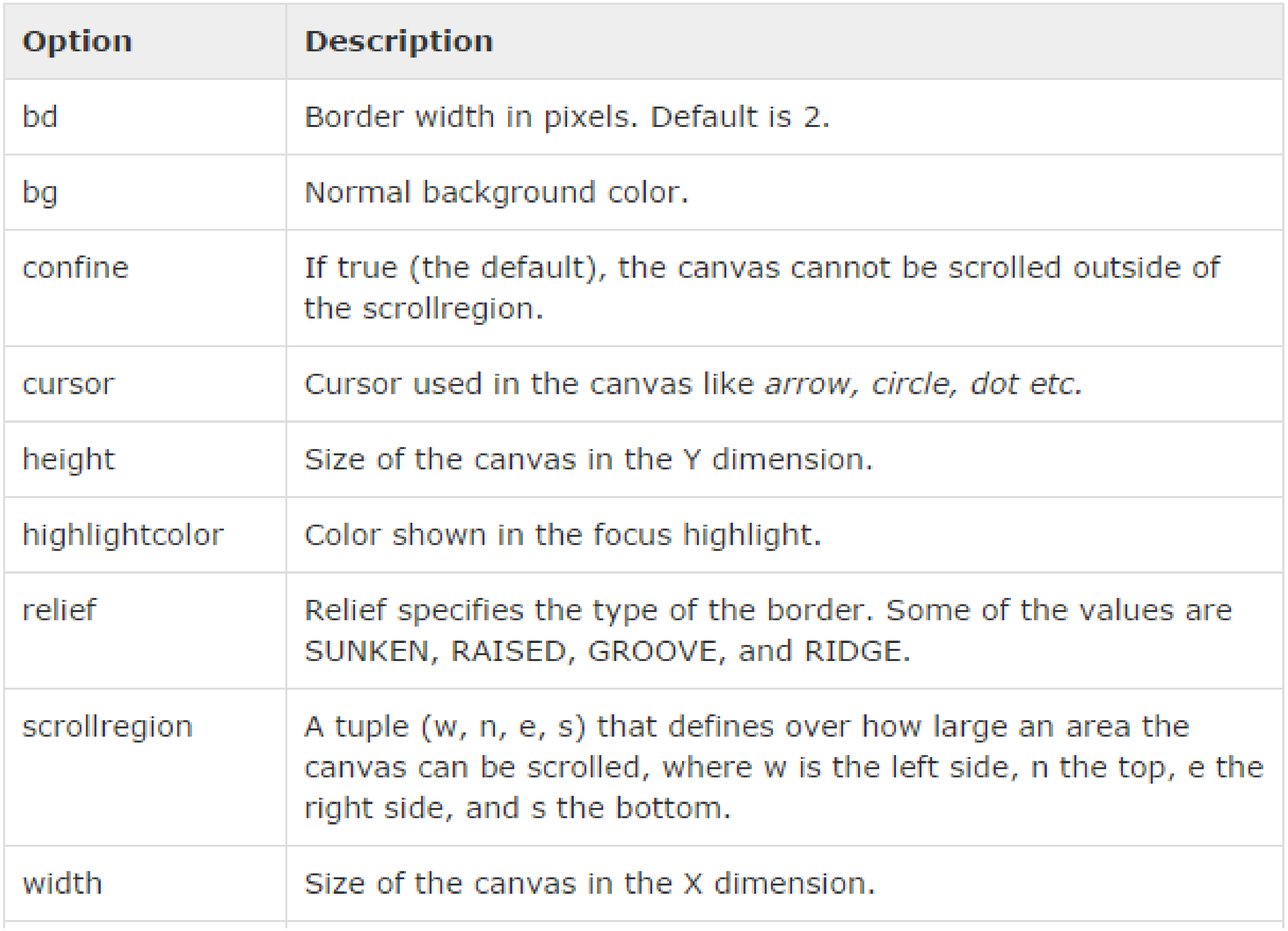
If you want an image button, use the PhotoImage class. We set the size of the window with the function geometry()

```
from tkinter import *
root= Tk()
root.geometry("320x100")  #use x not *
def callback():
   print("click!")
photo=PhotoImage(file="C:\\Users\\Mithila\\Deskt
op\\bitmap.png")
b = Button(root,image=photo, command=callback)
b.pack()
root.mainloop()
```

Canvas widget

- 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

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



Option	Description
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.
relief	Relief specifies the type of the border. Some of the values are SUNKEN, RAISED, GROOVE, and RIDGE.
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.
width	Size of the canvas in the X dimension.

- The Canvas widget can support the following standard items:

  line: Creates a line item.

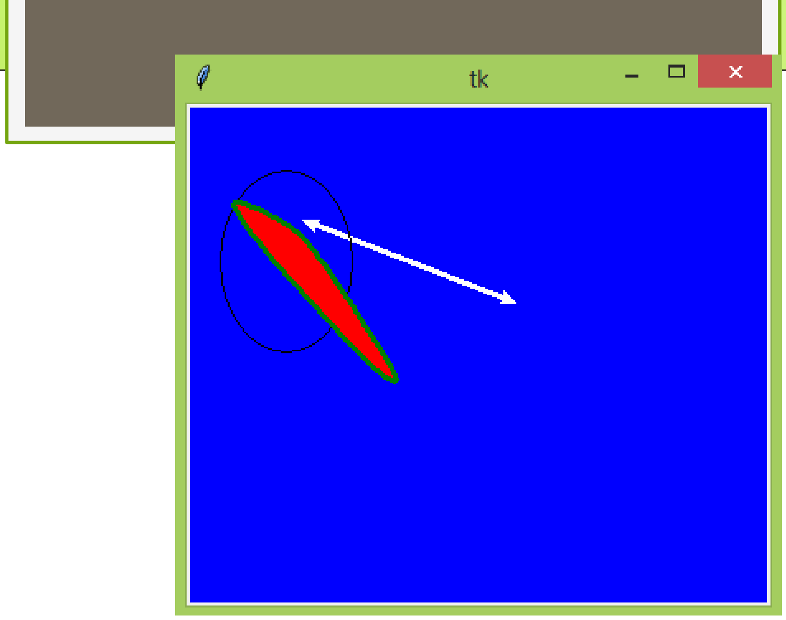
```
line = canvas.create_line(x0, y0, x1, y1, ..., xn, yn, options)
```

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)
```

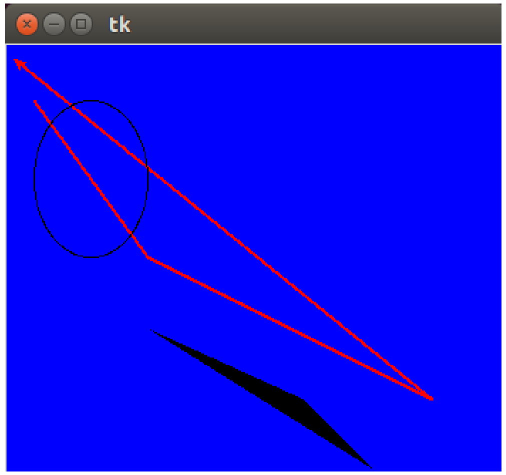
polygon: Creates a polygon item that must have at least three vertices.

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

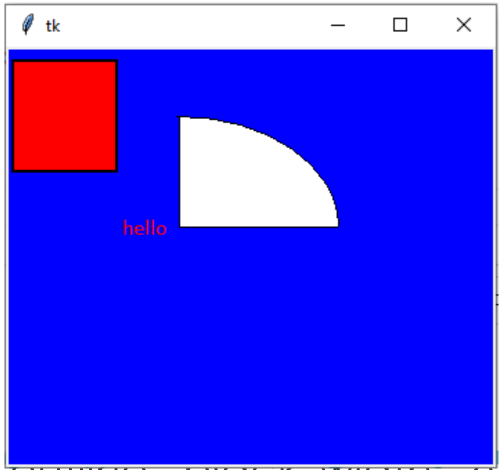


```
import tkinter

top = tkinter.Tk()

C = tkinter.Canvas(top, bg="blue",cursor='circle',height=300,width=350)
#line
line=C.create_line(70,70,200,120,arrow='both',fill='white',width=3)
#oval
oval=C.create_oval(20,40,100,150)
#polygon
polygon=C.create_polygon(10,50,60,70,80,90,150,200,outline='green',fill='red',sm
    ooth=1,width=3)
C.pack()
top.mainloop()
```

```
import Tkinter
top = tkinter.Tk()

C = tkinter.Canvas(top, bg="blue",cursor='circle',height=300,width=350)

#line
line=C.create_line(20,40,100,150,300,250,5,10,arrow='last',fill='red',width=2)

#oval
oval=C.create_oval(20,40,100,150)

#polygon
polygon=C.create_polygon(100,200,260,300,210,250)

C.pack()
top.mainloop()
```



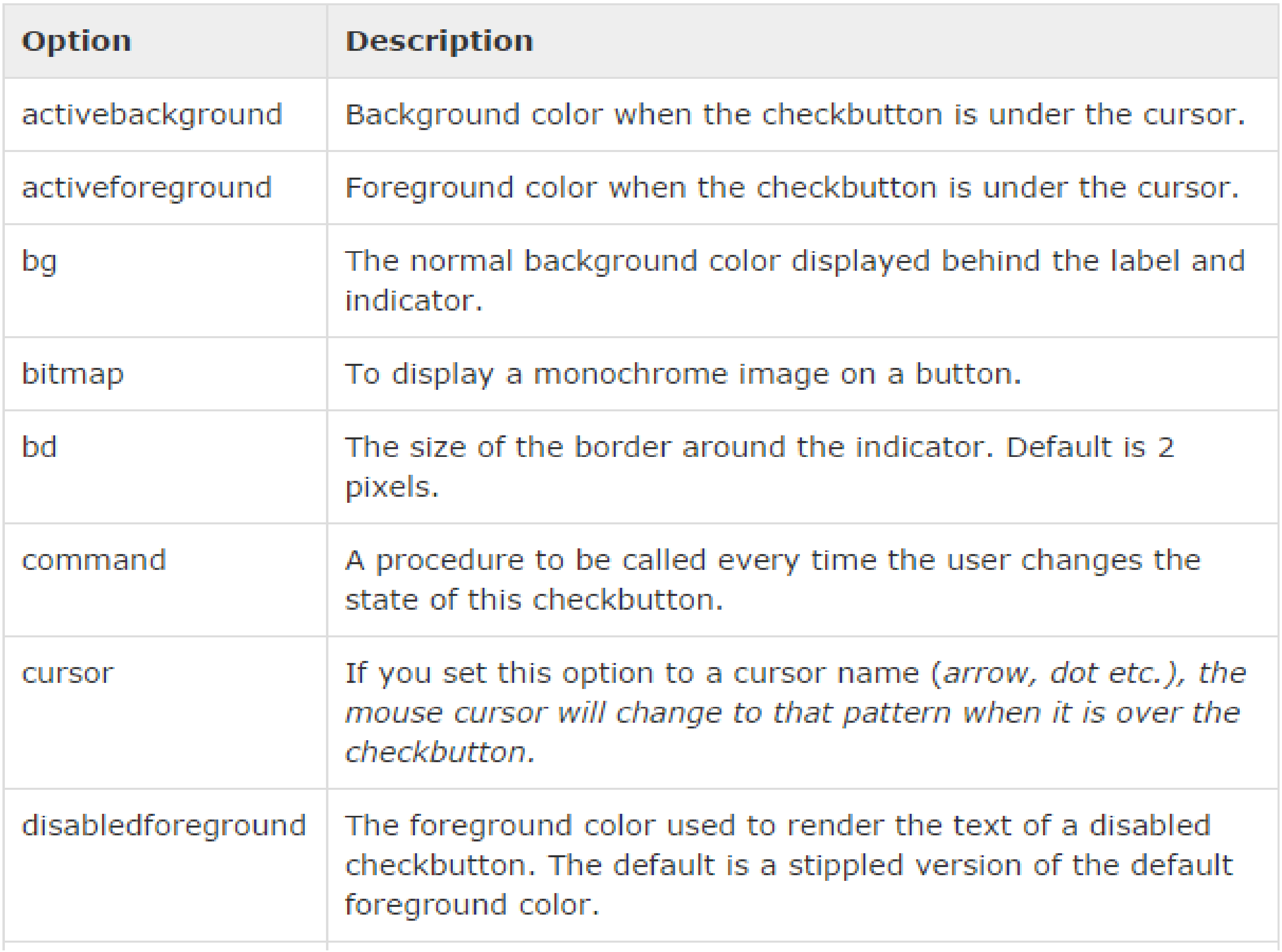
```
import tkinter
top = tkinter.Tk()

C = tkinter.Canvas(top, bg="blue",cursor='circle',
    height=300,width=350)

#rectangle
rectangle=C.create_rectangle(5,10,80,90,fill='red',outline='black',width=2)

coord = 10, 50, 240, 210
arc = C.create_arc(coord, start=0, extent=90, fill="white")

#Text canvas
text=C.create_text(100,130,text='hello',fill='red',font='Calibri')
C.pack()
top.mainloop()
```



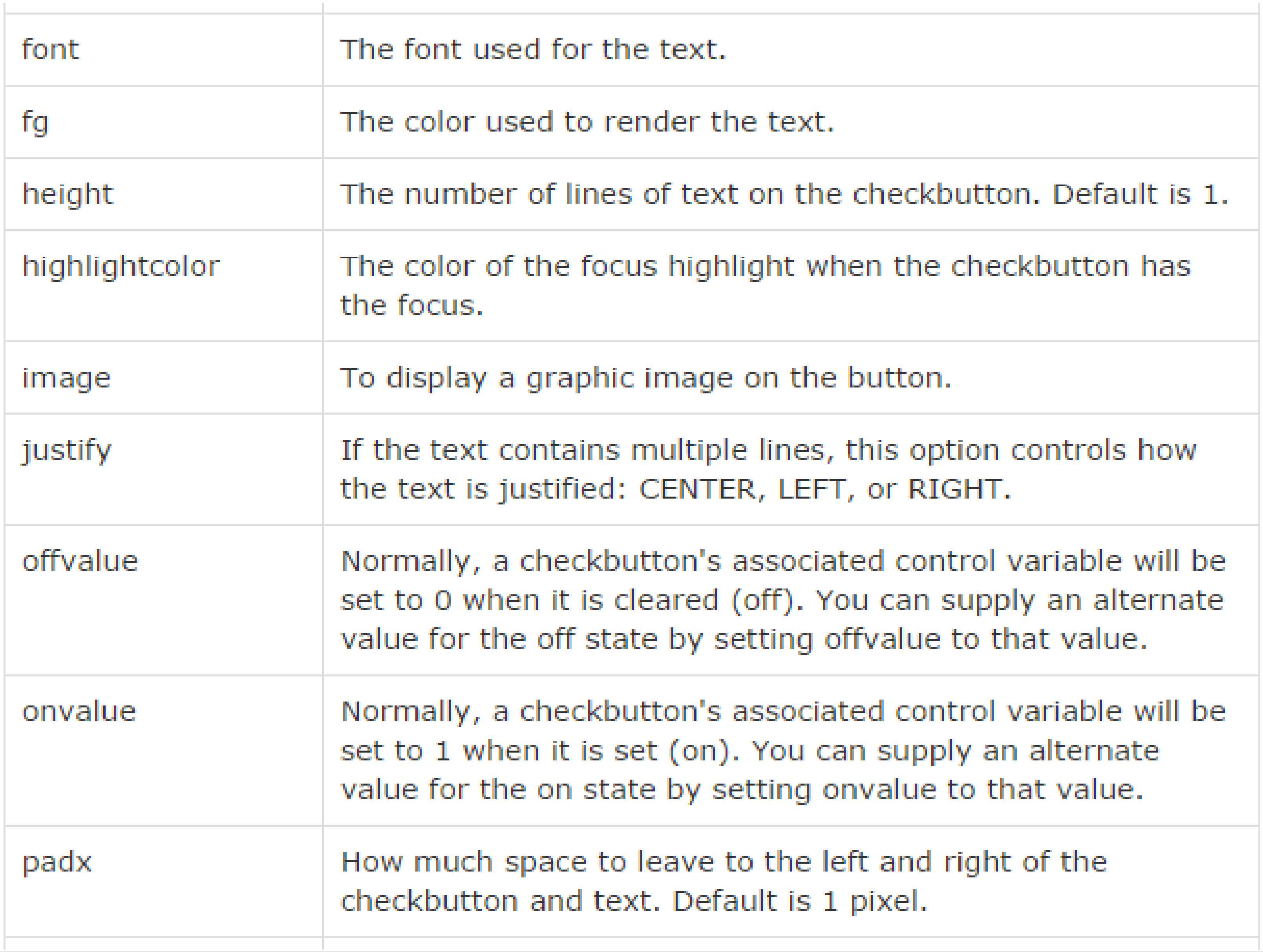
Checkbutton widget

- 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

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



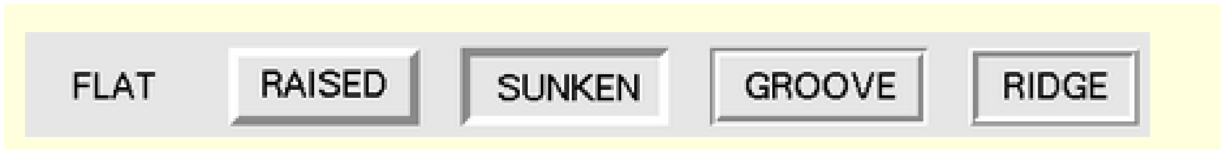
Option	Description
activebackground	Background color when the checkbutton is under the cursor.
activeforeground	Foreground color when the checkbutton is under the cursor.
bg	The normal background color displayed behind the label and indicator.
bitmap	To display a monochrome image on a button.
bd	The size of the border around the indicator. Default is 2 pixels.
command	A procedure to be called every time the user changes the state of this checkbutton.
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.*
disabledforeground	The foreground color used to render the text of a disabled checkbutton. The default is a stippled version of the default foreground color.



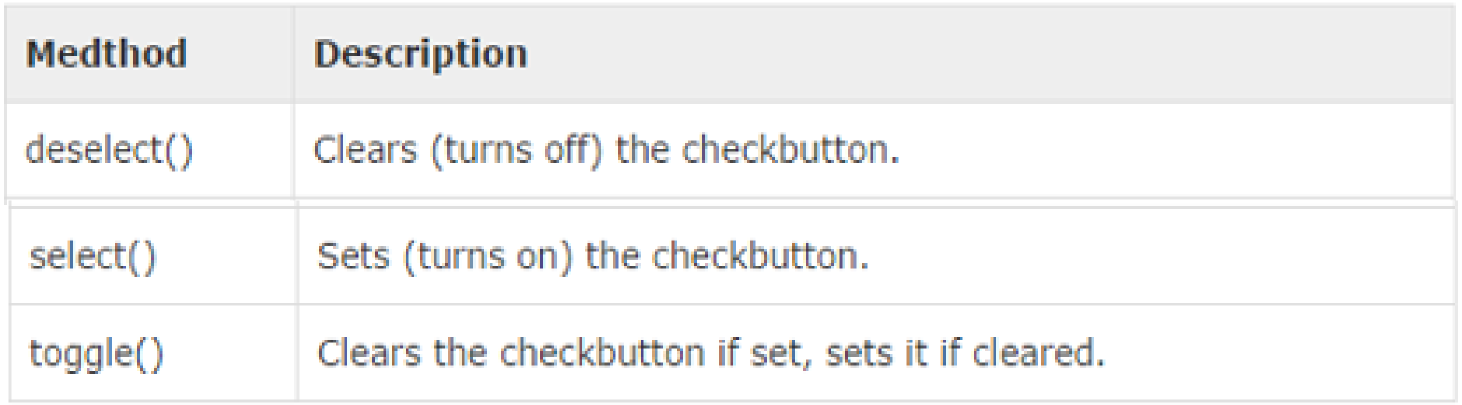
font	The font used for the text.
fg	The color used to render the text.
height	The number of lines of text on the checkbutton. Default is 1.
highlightcolor	The color of the focus highlight when the checkbutton has the focus.
image	To display a graphic image on the button.
justify	If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.
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.
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.
padx	How much space to leave to the left and right of the checkbutton and text. Default is 1 pixel.

pady	How much space to leave above and below the checkbutton and text. Default is 1 pixel.
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
selectcolor	The color of the checkbutton when it is set. Default is selectcolor="red".
selectimage	If you set this option to an image, that image will appear in the checkbutton when it is set.
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.
text	The label displayed next to the checkbutton. Use newlines ("\n") to display multiple lines of text.
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.

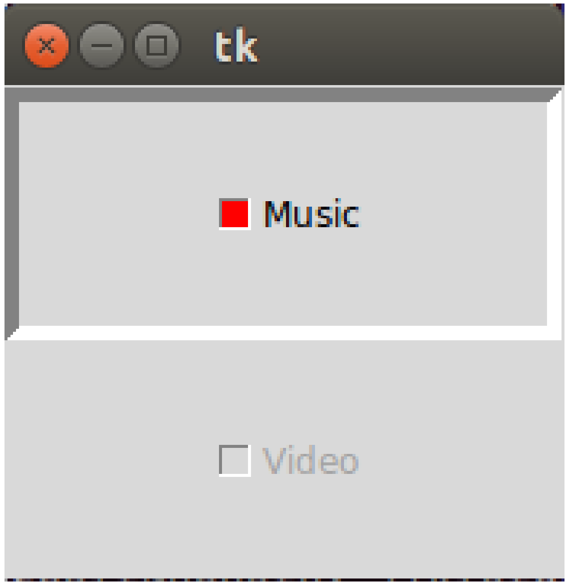
variable	The control variable that tracks the current state of the checkbutton. Normally this variable is an *IntVar*, and 0 means cleared and 1 means set, but see the offvalue and onvalue options above.
width	The default width of a checkbutton is determined by the size of the displayed image or text. You can set this option to a number of characters and the checkbutton will always have room for that many characters.
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:



Medthod	Description
deselect()	Clears (turns off) the checkbutton.
select()	Sets (turns on) the checkbutton.
toggle()	Clears the checkbutton if set, sets it if cleared.



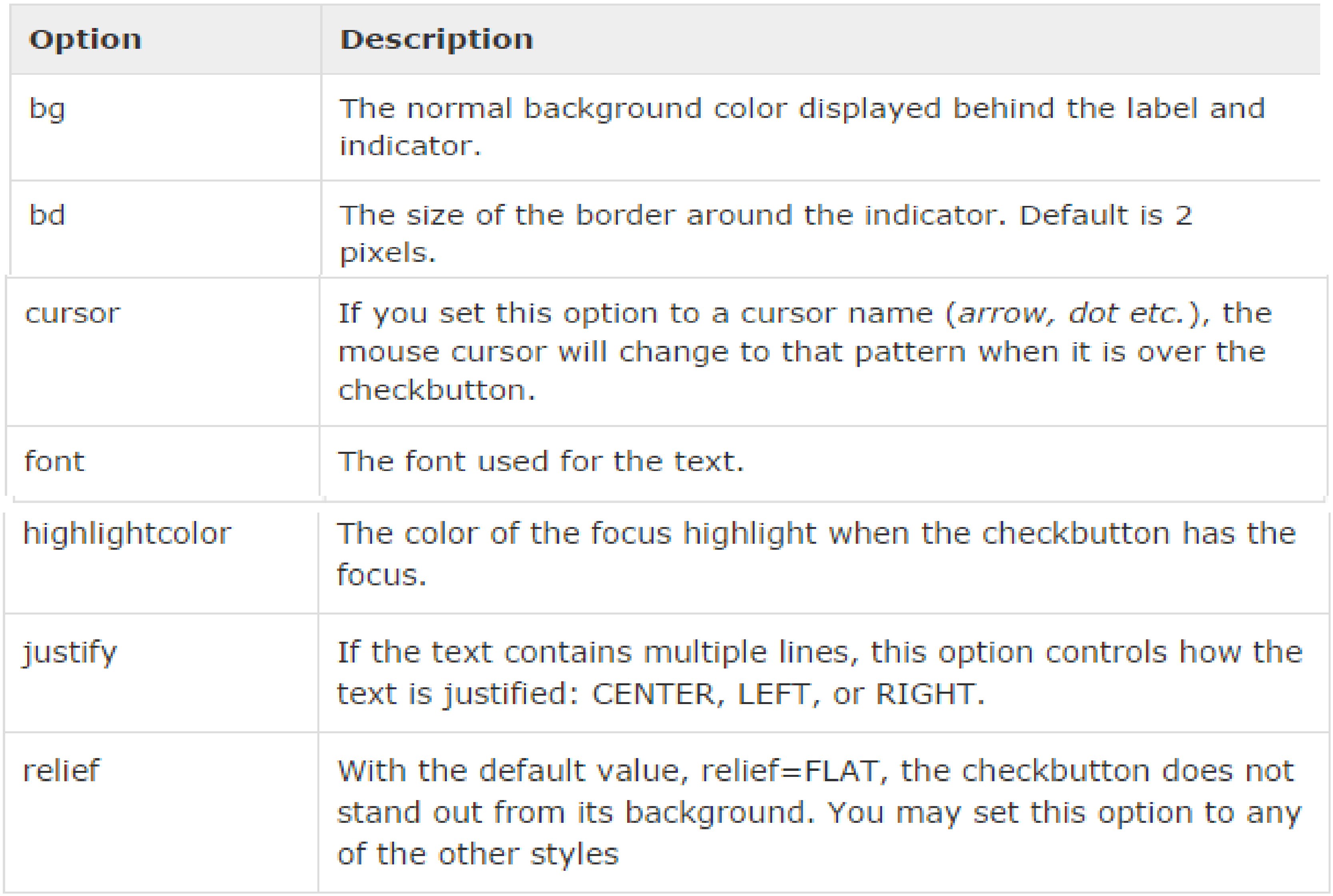
```
import tkinter
from tkinter import *

top = tkinter.Tk()

C1 = Checkbutton(top, text="Music", height=5, relief='sunken', width = 20, bd=5, selectcolor='red')
C2 = Checkbutton(top, text = "Video", state='disabled', height=5, width = 20)
C1.pack()
C2.pack()
top.mainloop()
```

```
import tkinter
from tkinter import *
top = tkinter.Tk()
def h():
   print("value: " ,var.get())
var=IntVar()
C1 = Checkbutton(top, text="Music", height=5, onvalue=5, offvalue=10,
variable=var, relief='sunken', width = 20, bd=5, selectcolor='red')
C2 = Checkbutton(top, text = "Video", disabledforeground='green',
state='disabled',height=5, width = 20)
B=Button(top, text='Show', command=h)
C1.pack()
C2.pack()
B.pack()
top.mainloop()
```

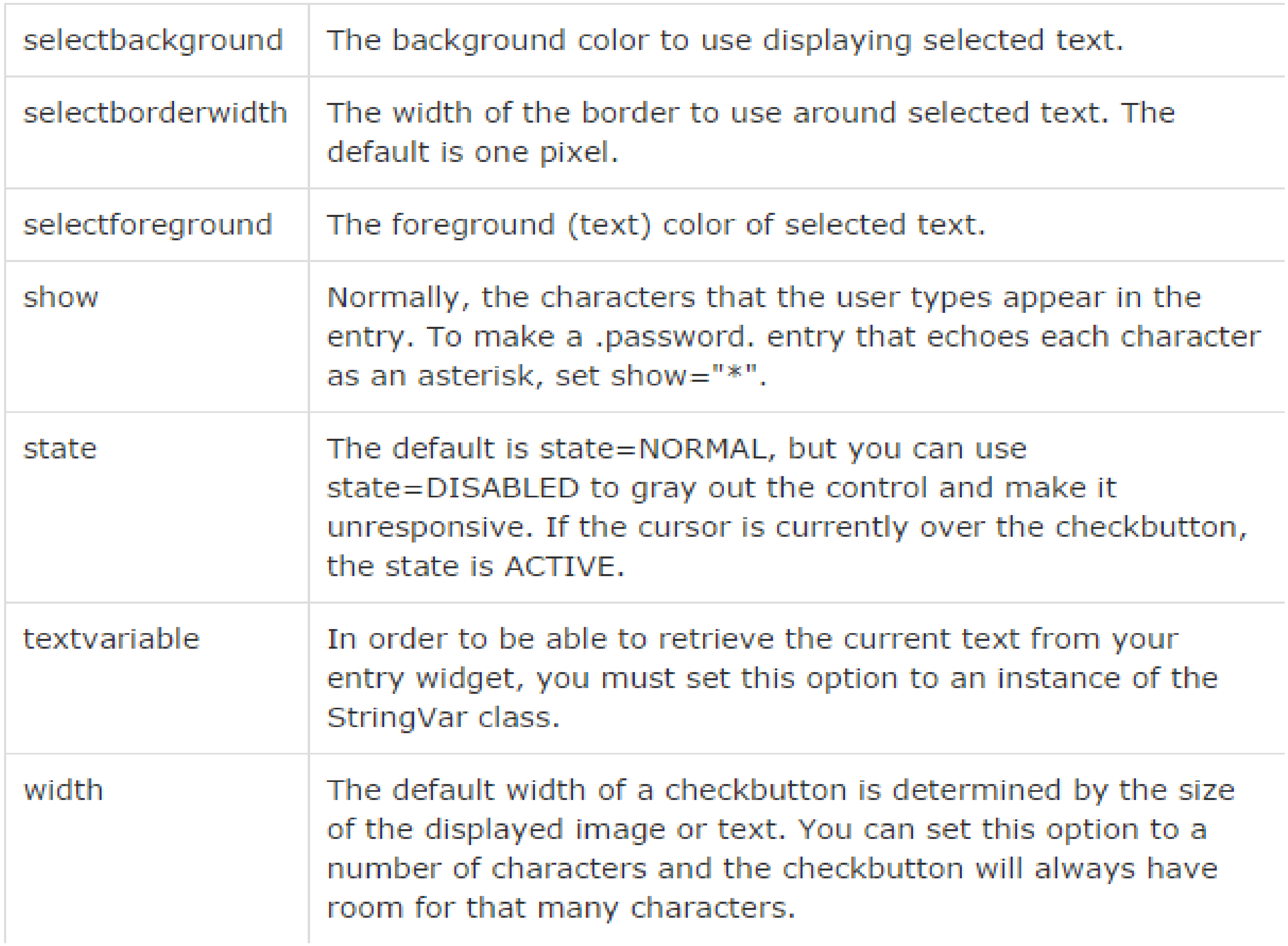
Entry widget

- 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, ... )
```

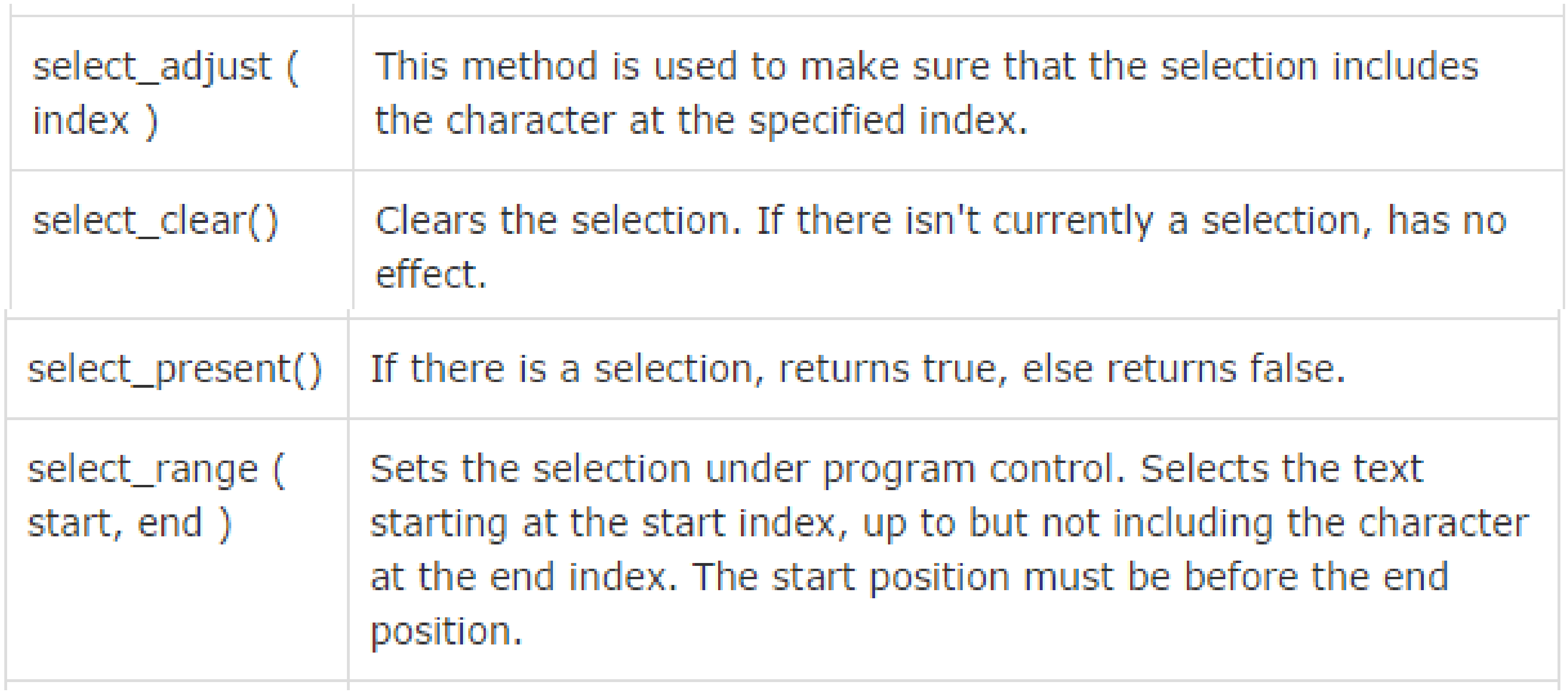


Option	Description
bg	The normal background color displayed behind the label and indicator.
bd	The size of the border around the indicator. Default is 2 pixels.
cursor	If you set this option to a cursor name (*arrow, dot etc.*), the mouse cursor will change to that pattern when it is over the checkbutton.
font	The font used for the text.
highlightcolor	The color of the focus highlight when the checkbutton has the focus.
justify	If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.
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



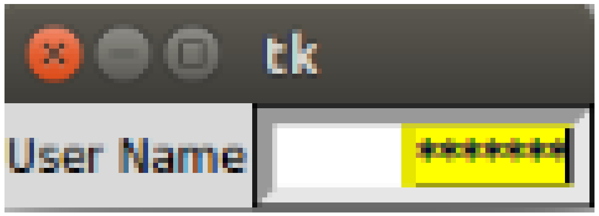
selectbackground	The background color to use displaying selected text.
selectborderwidth	The width of the border to use around selected text. The default is one pixel.
selectforeground	The foreground (text) color of selected text.
show	Normally, the characters that the user types appear in the entry.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

Medthod	Description
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.

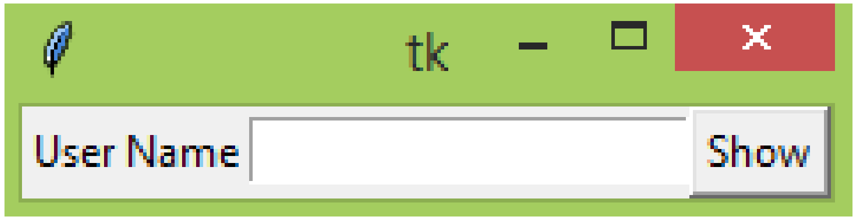


select_adjust (index)	This method is used to make sure that the selection includes the character at the specified index.
select_clear()	Clears the selection. If there isn't currently a selection, has no effect.
select_present()	If there is a selection, returns true, else returns false.
select_range (start, 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.

```
from tkinter import *

top = Tk()
L1 = Label(top, text="User Name")
L1.pack(side = LEFT)
E1 = Entry(top, bd =5,bg='white',fg='blue',cursor='dot',justify='right',\
selectbackground='yellow', selectborderwidth=4,show='*',width=10)
E1.pack(side = LEFT)

top.mainloop()
```



```
from tkinter import *

top = Tk()
def show_entry_fields():
   print("Text: %s" % E1.get())
L1 = Label(top, text="User Name")
L1.pack(side = LEFT)
E1 = Entry(top)
E1.pack(side = LEFT)
b=Button(top, text='Show', command=show_entry_fields)
b.pack()
top.mainloop()
```

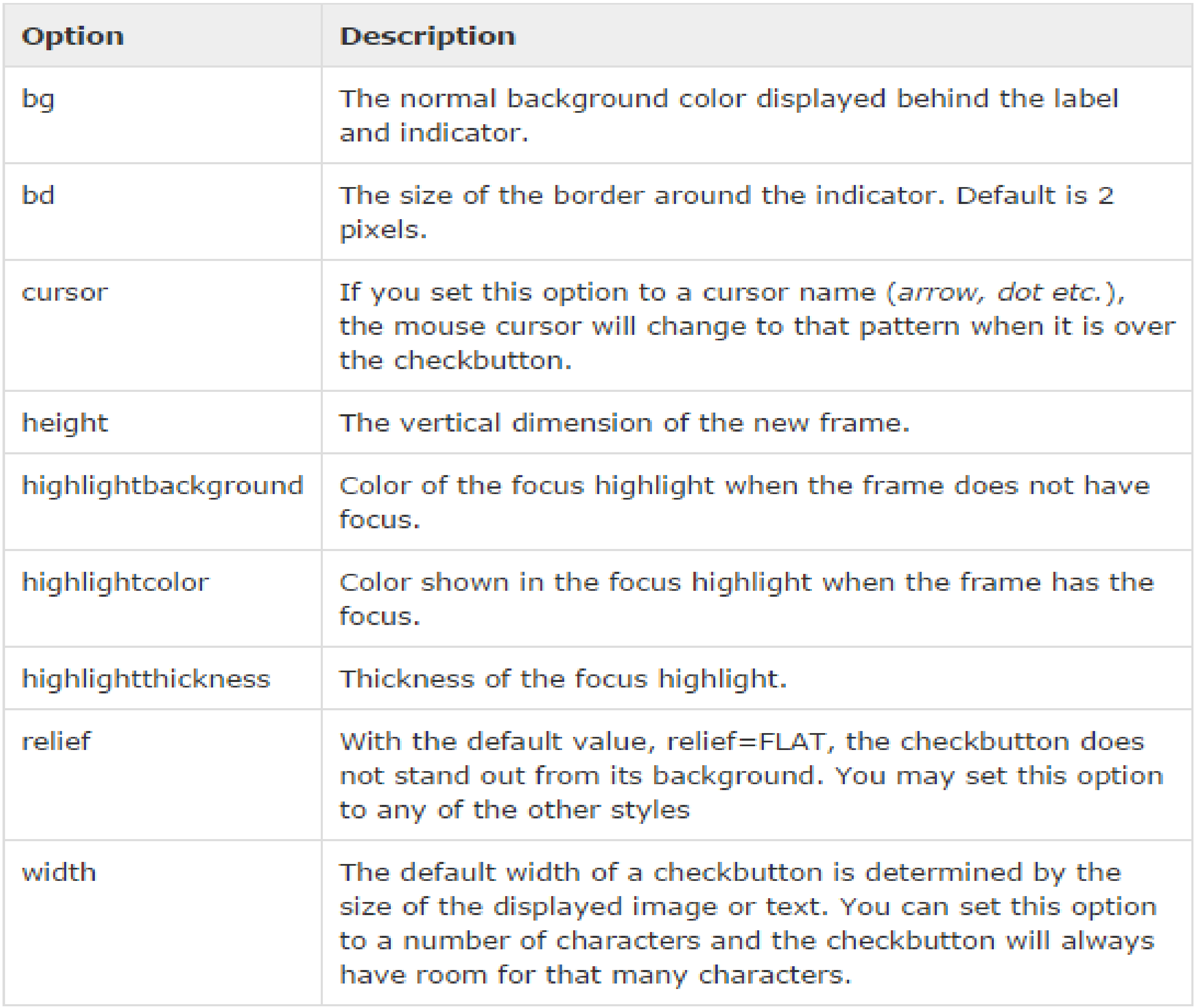


```
from tkinter import *

top = Tk()
def show_entry_fields():
    E1.delete(0)
    #E1.delete(0,2)
    #E1.icursor(2)
    #E1.insert(1,"hello")
    #E1.select_adjust(2)
    #E1.select_range(2,4)
L1 = Label(top, text="User Name")
L1.pack(side = LEFT)
E1 = Entry(top)
E1.pack(side = LEFT)
b=Button(top, text='Show', command=show_entry_fields)
b.pack()
top.mainloop()
```

Frame widget

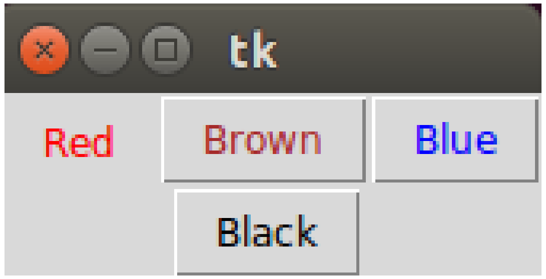
- The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.
- It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets.
- Syntax:

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



Option	Description
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.
highlightbackground	Color of the focus highlight when the frame does not have focus.
highlightcolor	Color shown in the focus highlight when the frame has the focus.
highlightthickness	Thickness of the focus highlight.
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
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.

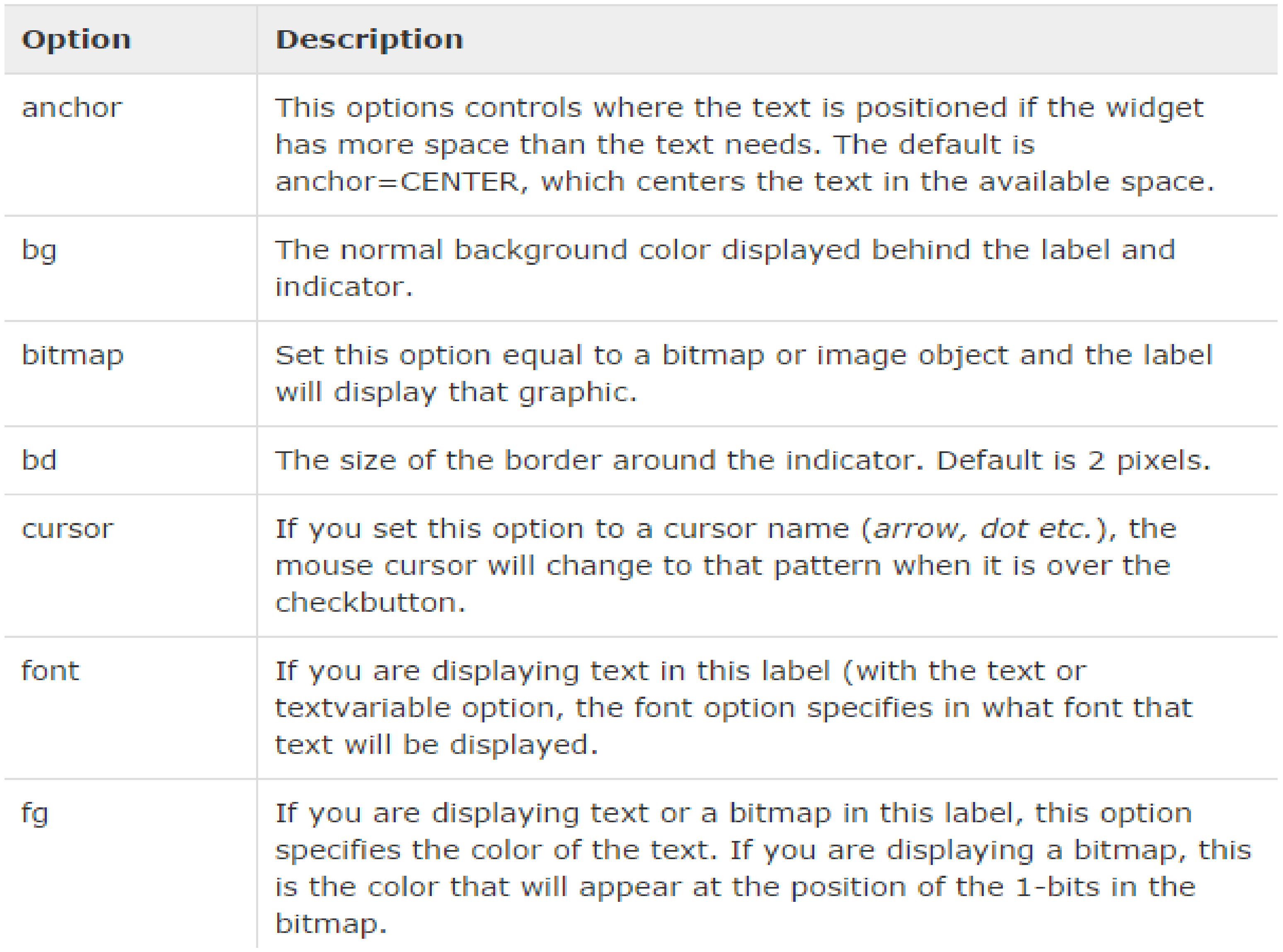
```
from tkinter import *

root = Tk()
frame = Frame(root)
frame.pack()

bottomframe = Frame(root)
bottomframe.pack(side = BOTTOM)

redbutton = Button(frame, text="Red", relief='flat',fg="red")
redbutton.pack(side=LEFT)

greenbutton = Button(frame, text="Brown", fg="brown")
greenbutton.pack(side=LEFT)

bluebutton = Button(frame, text="Blue", fg="blue")
bluebutton.pack(side=LEFT)

blackbutton = Button(bottomframe, text="Black", fg="black")
blackbutton.pack(side=LEFT)

root.mainloop()
```



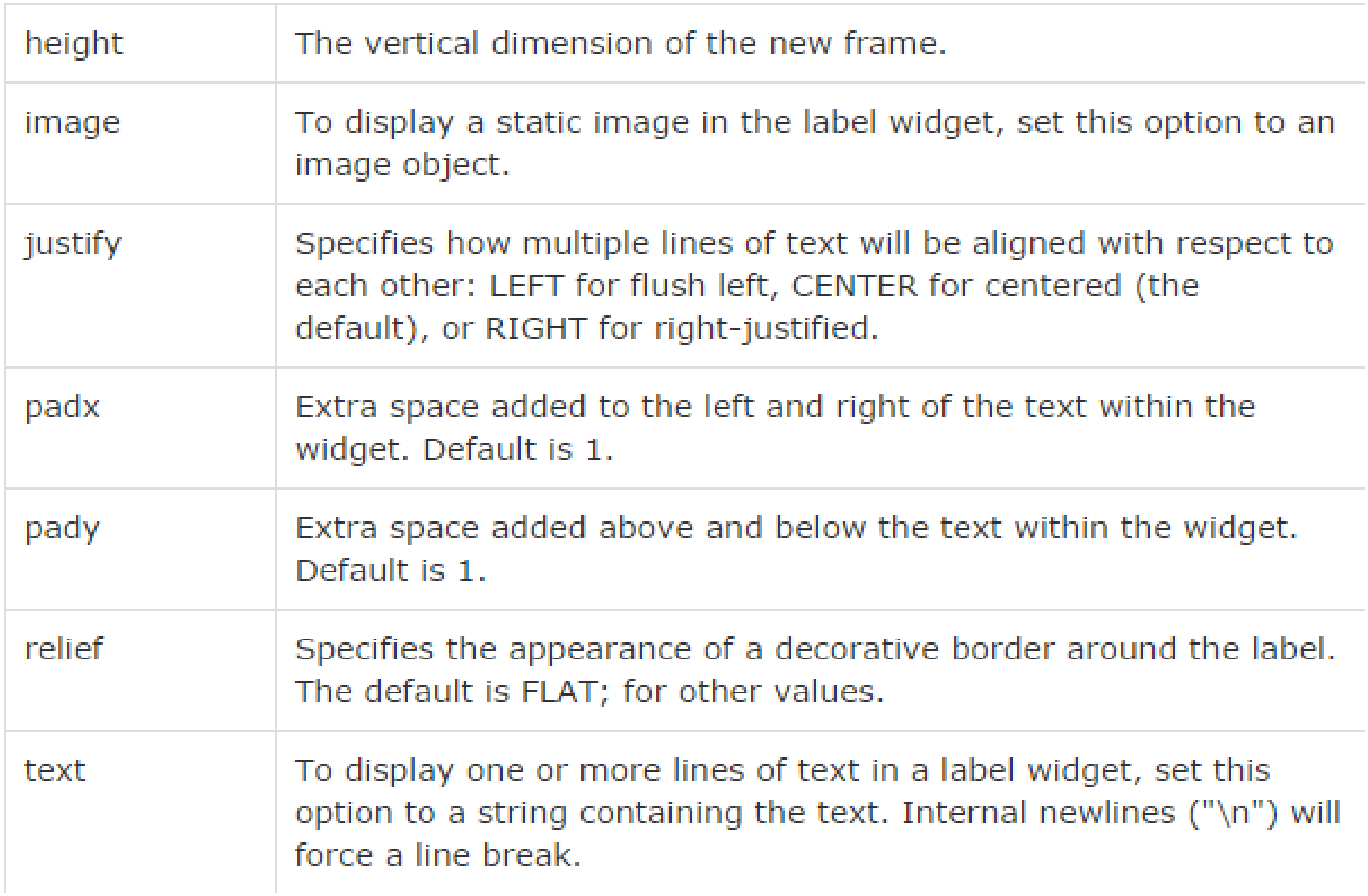
Label widget

- This widget implements a display box where you can place text or images.
- Syntax:

```
w = Label ( master, option, ... )
```



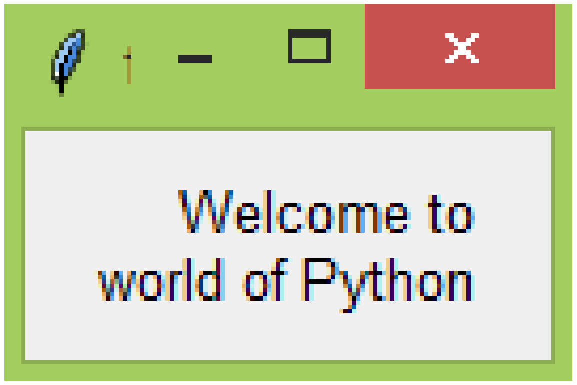
Option	Description
anchor	This options controls where the text is positioned if the widget has more space than the text needs. The default is anchor=CENTER, which centers the text in the available space.
bg	The normal background color displayed behind the label and indicator.
bitmap	Set this option equal to a bitmap or image object and the label will display that graphic.
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.
font	If you are displaying text in this label (with the text or textvariable option, the font option specifies in what font that text will be displayed.
fg	If you are displaying text or a bitmap in this label, this option specifies the color of the text. If you are displaying a bitmap, this is the color that will appear at the position of the 1-bits in the bitmap.



height	The vertical dimension of the new frame.
image	To display a static image in the label widget, set this option to an image object.
justify	Specifies how multiple lines of text will be aligned with respect to each other: LEFT for flush left, CENTER for centered (the default), or RIGHT for right-justified.
padx	Extra space added to the left and right of the text within the widget. Default is 1.
pady	Extra space added above and below the text within the widget. Default is 1.
relief	Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.
text	To display one or more lines of text in a label widget, set this option to a string containing the text. Internal newlines ("\n") will force a line break.

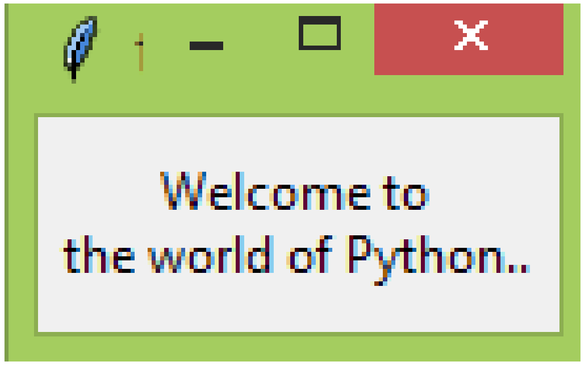
textvariable	To slave the text displayed in a label widget to a control variable of class *StringVar*, set this option to that variable.
underline	You can display an underline (_) below the nth letter of the text, counting from 0, by setting this option to n. The default is underline=-1, which means no underlining.
width	Width of the label in characters (not pixels!). If this option is not set, the label will be sized to fit its contents.
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.

```
from tkinter import *

root = Tk()
label = Label(root,text="Welcome to\nworld of Python",justify="right",height=3,font=("Helvetica", 10))

label.pack()
root.mainloop()
```



```
  Welcome to
world of Python
```

```
from tkinter import *

root = Tk()

var = StringVar()
label = Label(root,textvariable=var,height=3)

var.set("Welcome to\nthe world of Python..")
label.pack()
root.mainloop()
```



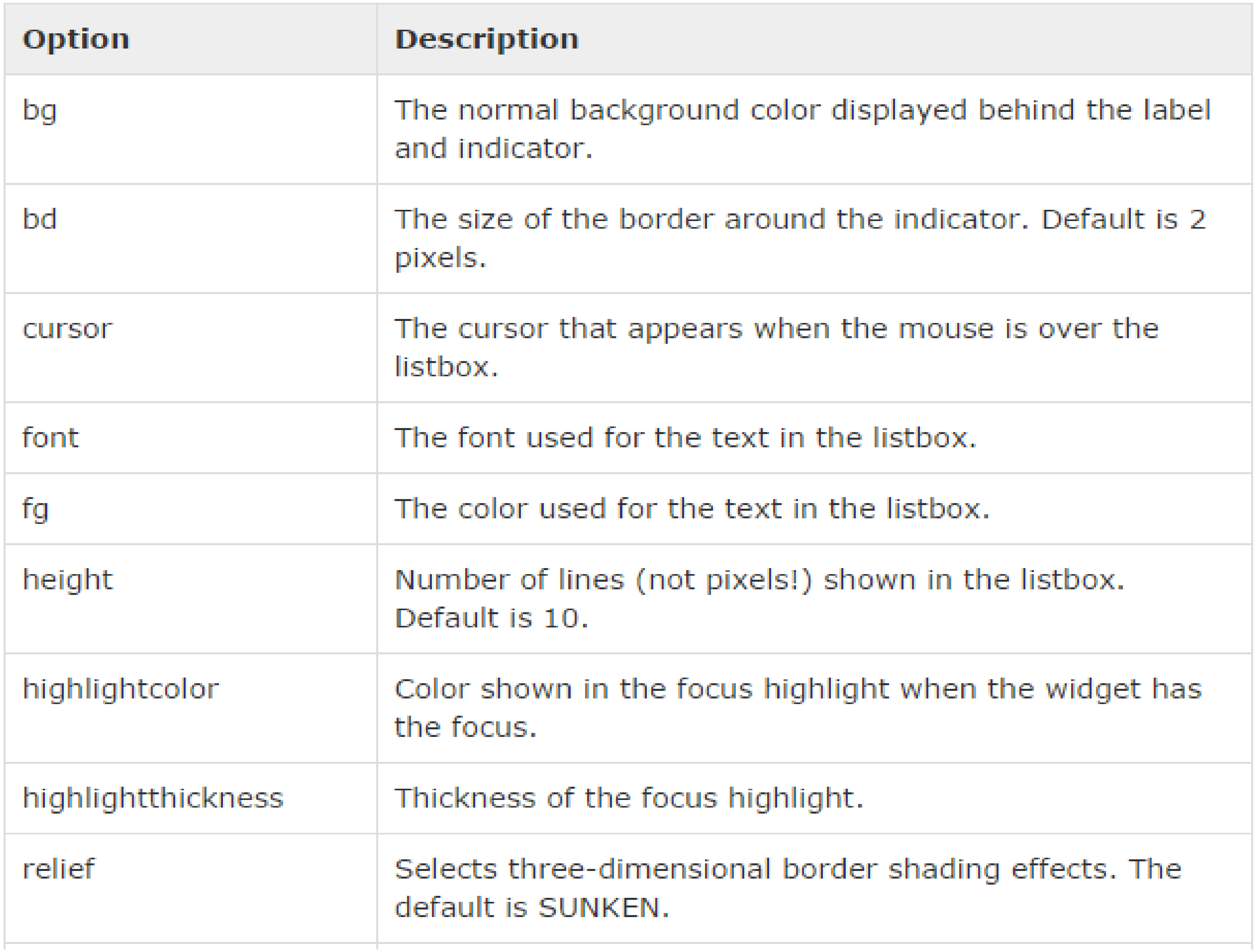
```
Welcome to
the world of Python..
```

Listbox widget

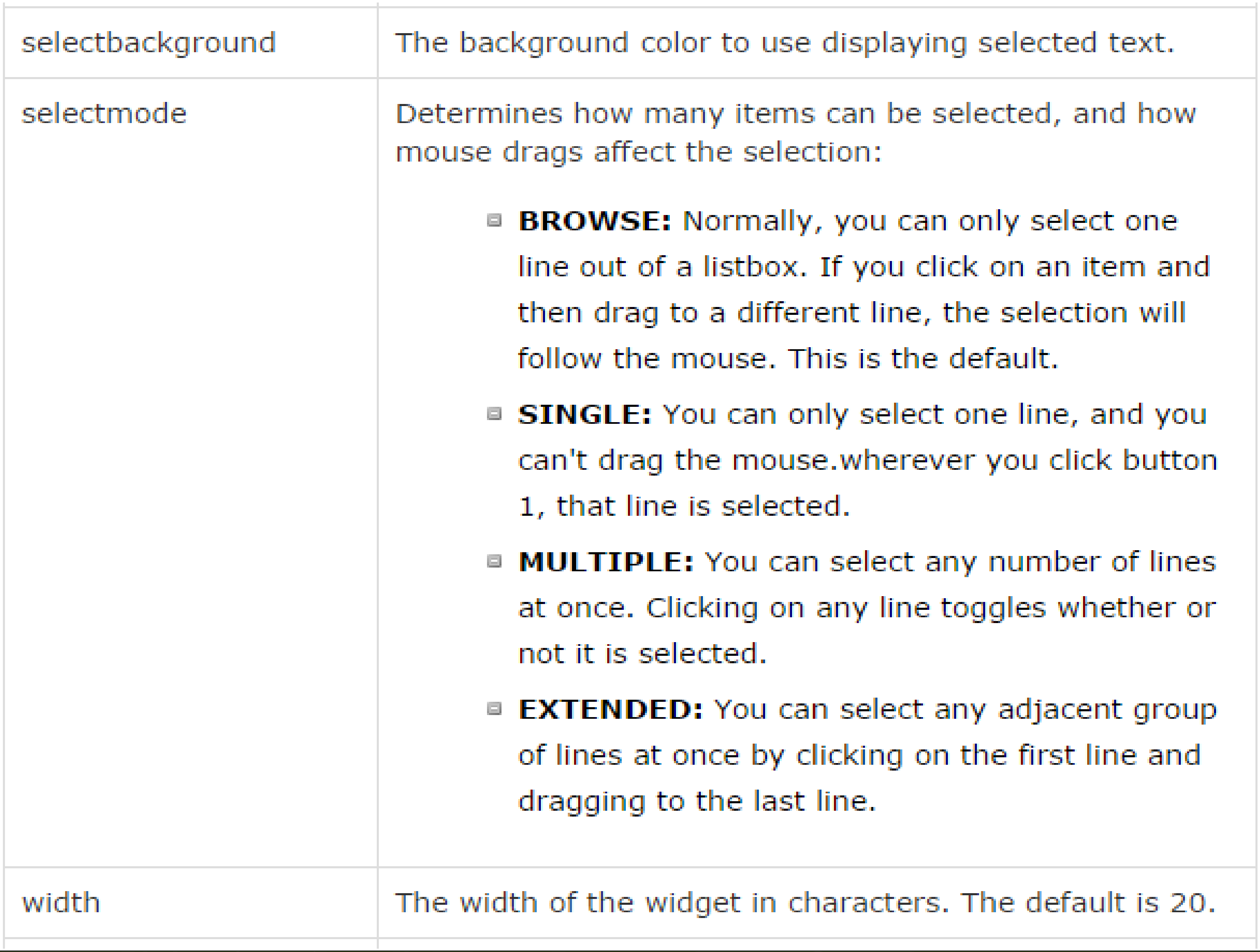
- The Listbox widget is used to display a list of items from which a user can select a number of items.

- Syntax:

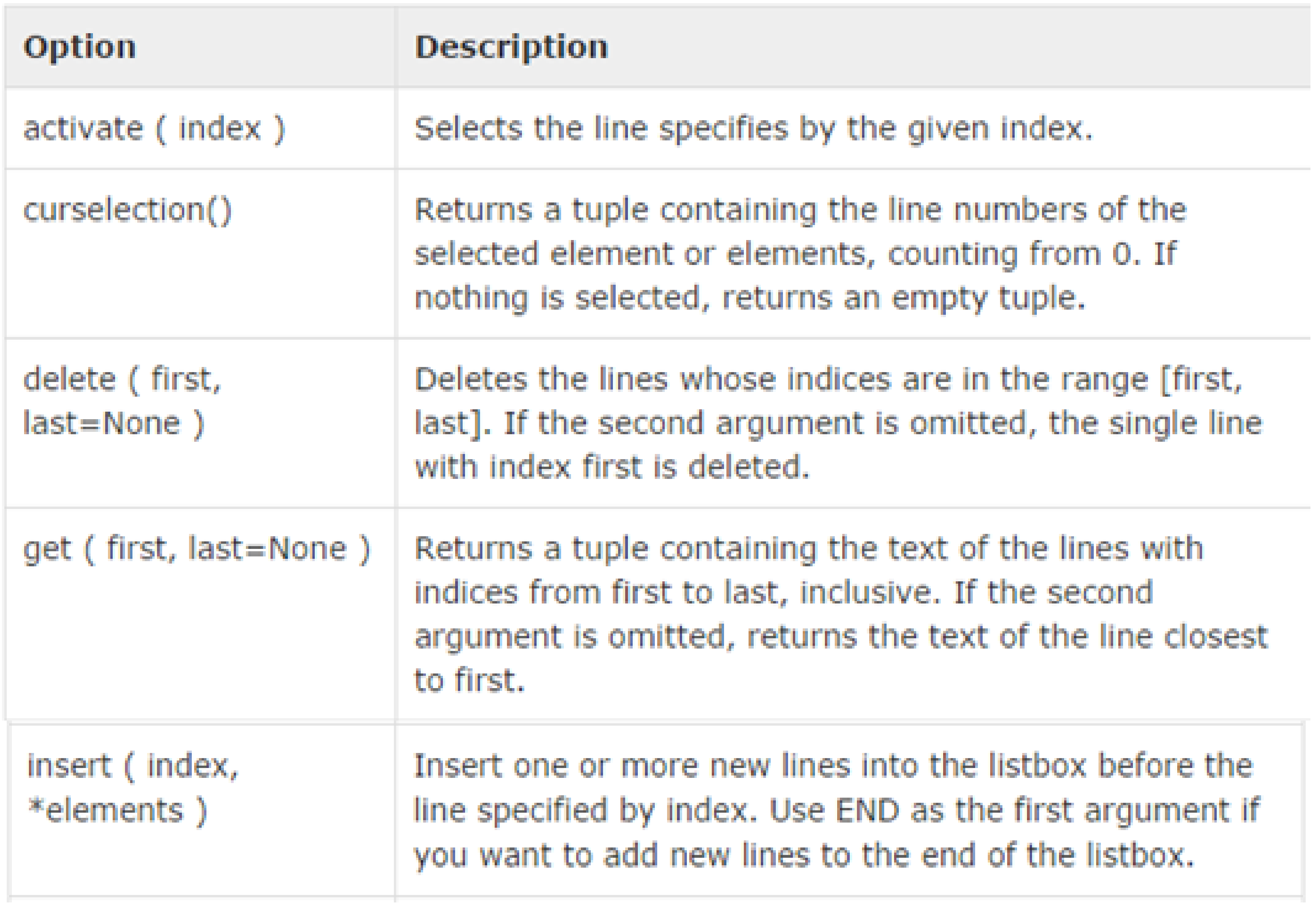
```
w = Listbox ( master, option, ... )
```



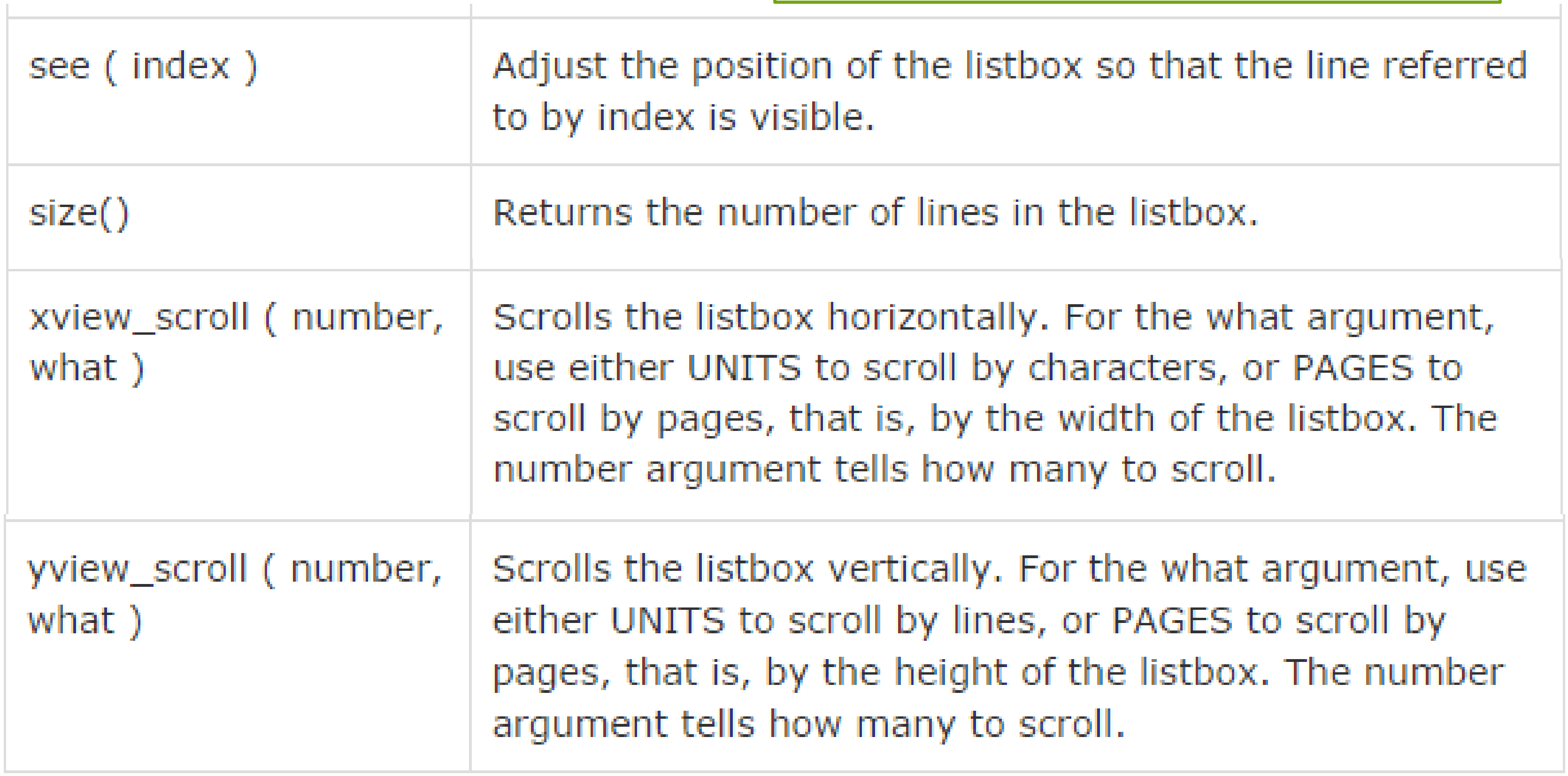
Option	Description
bg	The normal background color displayed behind the label and indicator.
bd	The size of the border around the indicator. Default is 2 pixels.
cursor	The cursor that appears when the mouse is over the listbox.
font	The font used for the text in the listbox.
fg	The color used for the text in the listbox.
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.
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.
width	The width of the widget in characters. The default is 20.

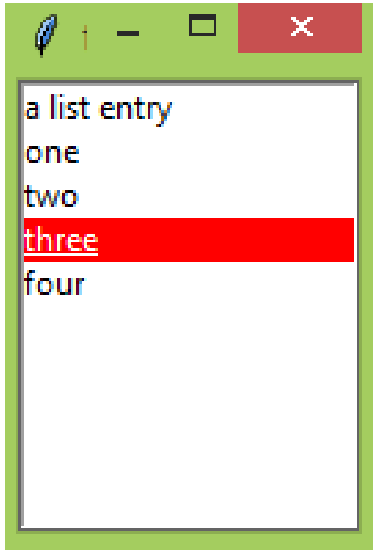


Option	Description
activate (index)	Selects the line specifies by the given index.
curselection()	Returns a tuple containing the line numbers of the selected element or elements, counting from 0. If nothing is selected, returns an empty tuple.
delete (first, last=None)	Deletes the lines whose indices 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.
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.



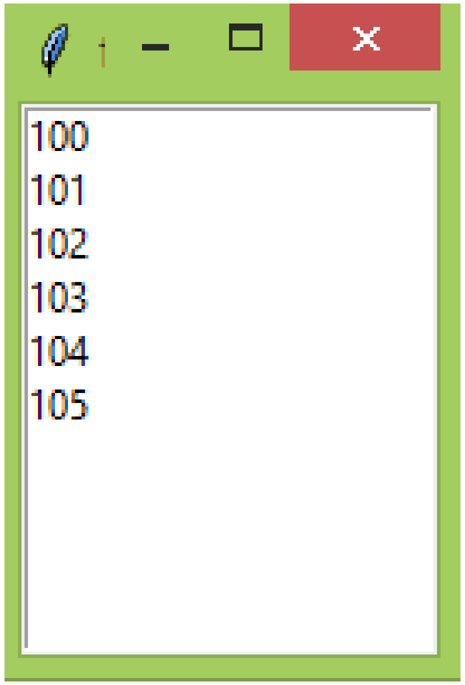
see (index)	Adjust the position of the listbox so that the line referred to by index is visible.
size()	Returns the number of lines in the listbox.
xview_scroll (number, what)	Scrolls the listbox horizontally. For the what argument, use either UNITS to scroll by characters, or PAGES to scroll by pages, that is, by the width of the listbox. The number argument tells how many to scroll.
yview_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.

```
from tkinter import *

master = Tk()

listbox = Listbox(master,selectbackground="red")
#try selectmode option with different values

listbox.pack()

listbox.insert(END, "a list entry")

for item in ["one", "two", "three", "four"]:
    listbox.insert(END, item)

mainloop()

#try delete, size, see options
```



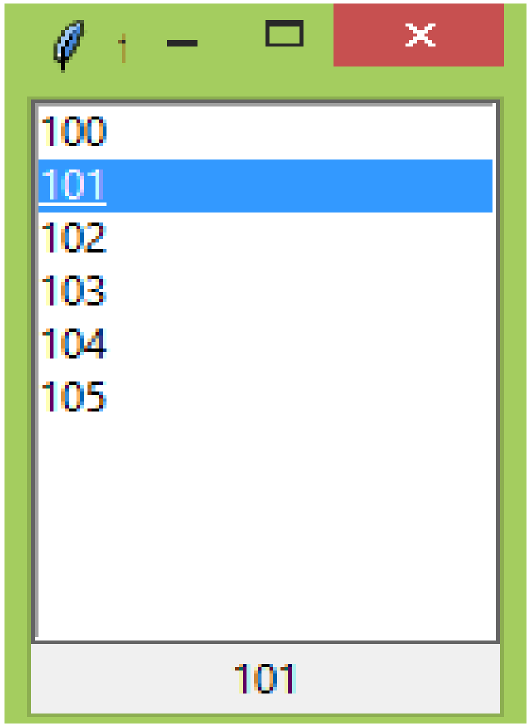
```
a list entry
one
two
three
four
```

```
import tkinter

top=tkinter.Tk()
L = tkinter.Listbox(top)

L.pack()

for i in range(6):
  L.insert(i, i+100)

def callback(event):
   print(L.get(L.curselection()))

L.bind("<<ListboxSelect>>", callback)

top.mainloop()
```



```
100
101
102
103
104
105
```

```
import tkinter
from tkinter import *

top=tkinter.Tk()
L = tkinter.Listbox(top)
Lb=Label(top)
Lb.pack(side=BOTTOM)
L.pack()

for i in range(6):
   L.insert(i, i+100)

def callback(event):
    Lb.config(text=L.get(L.curselection()))

L.bind("<<ListboxSelect>>", callback)

top.mainloop()
```

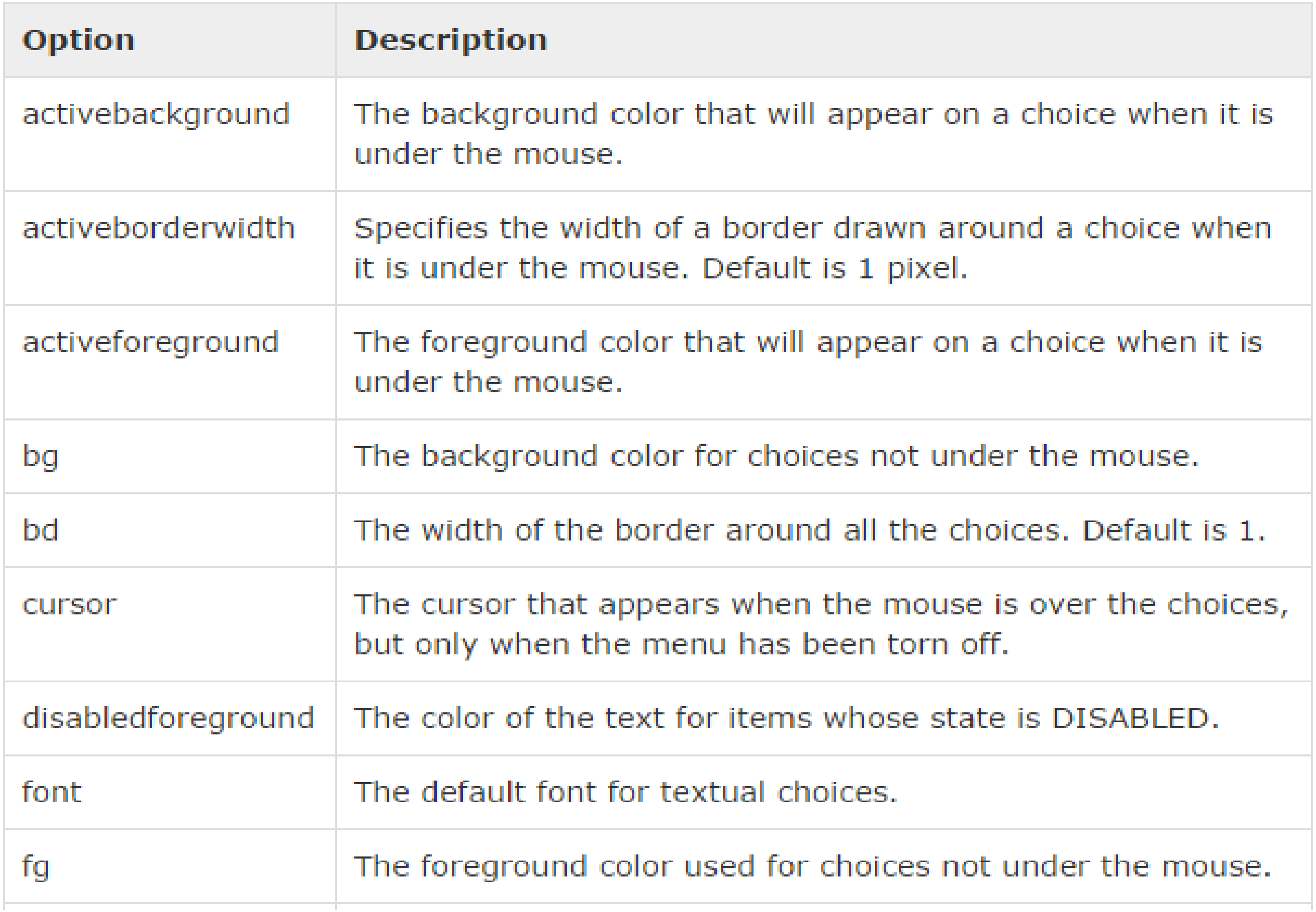


```
100
101
102
103
104
105

101
```

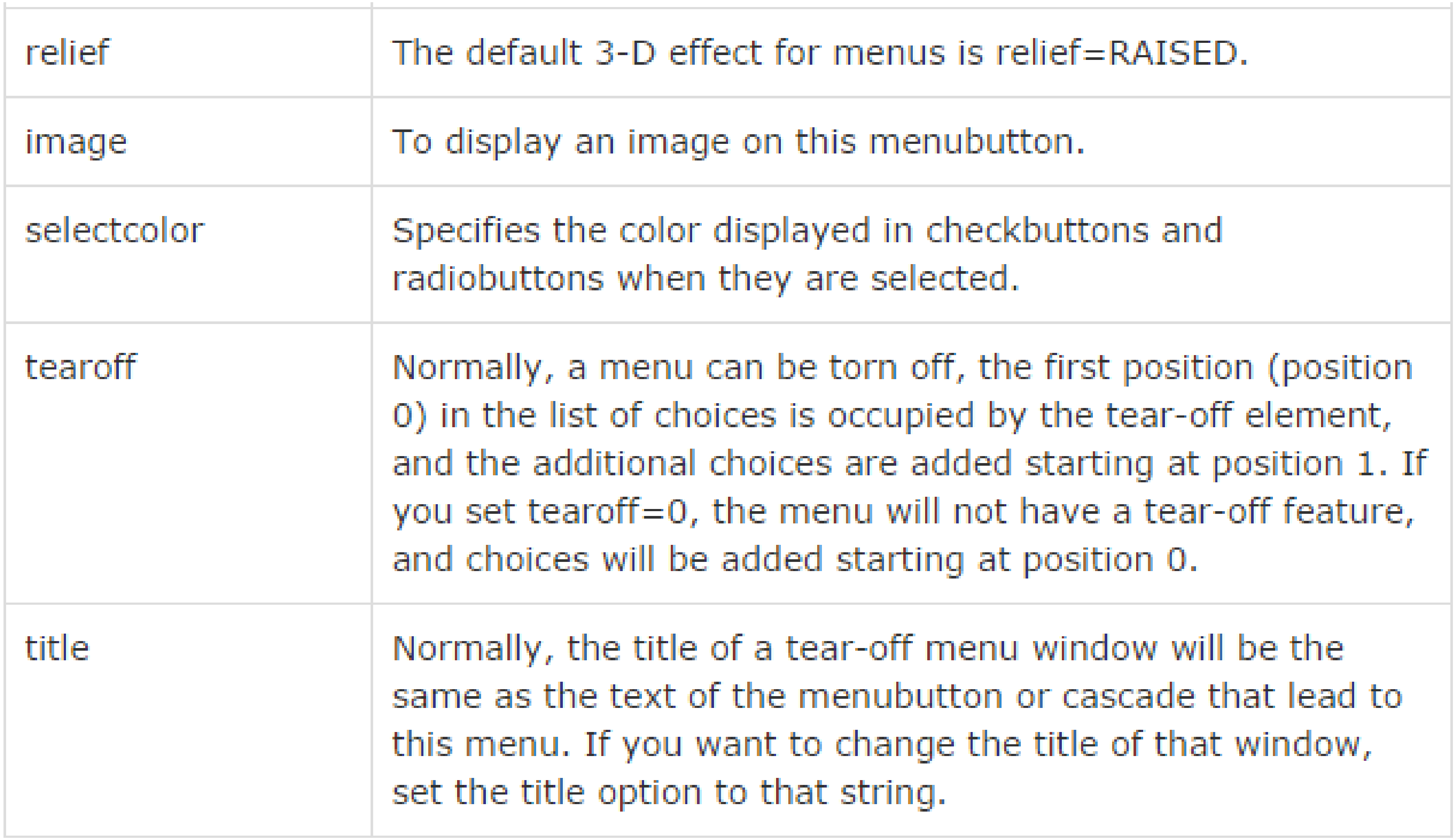
Menu widget

- The goal of this widget is to allow us to create all kinds of menus that can be used by our applications.
- Syntax:

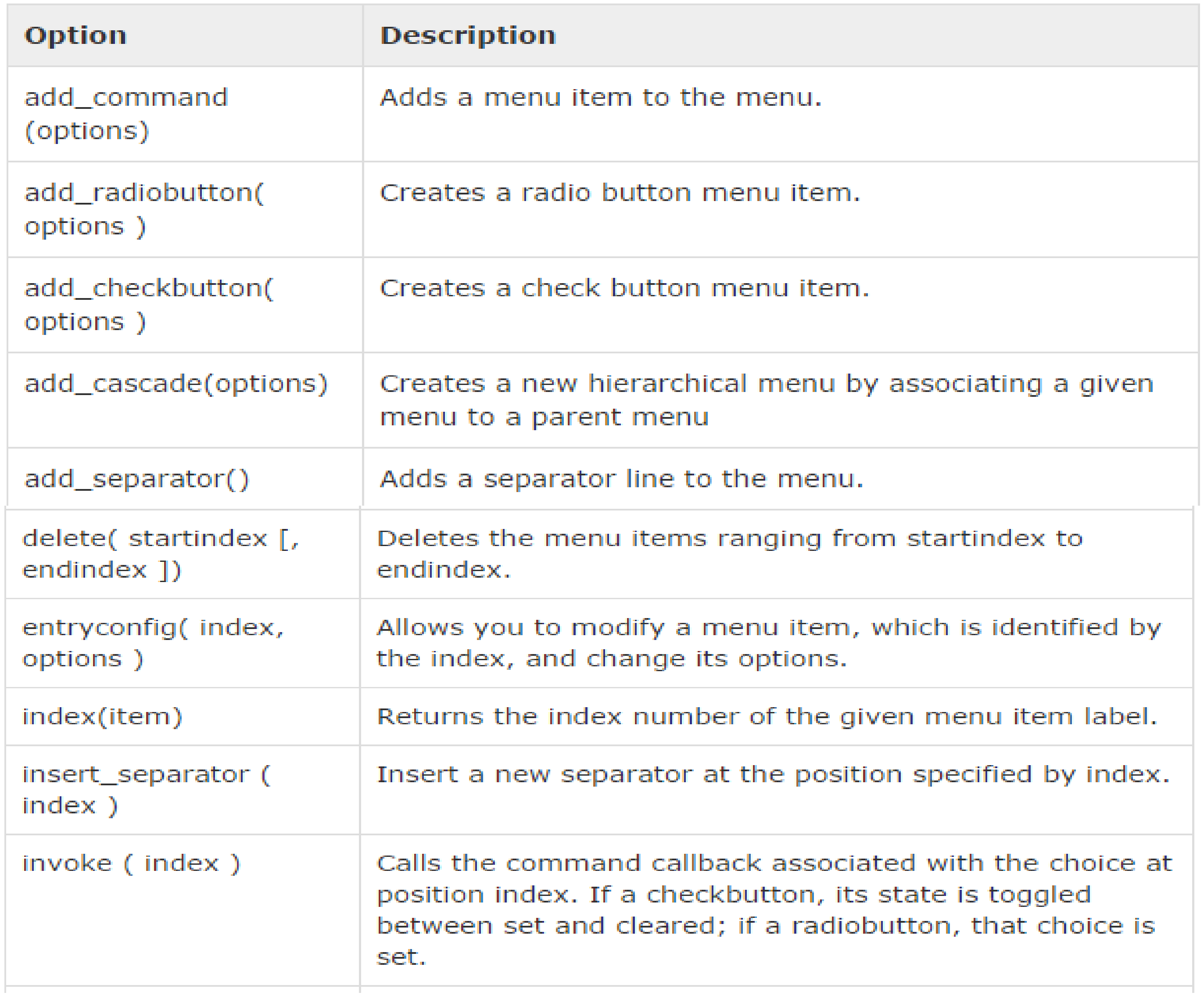
```
w = Menu ( master, option, ... )
```



Option	Description
activebackground	The background color that will appear on a choice when it is under the mouse.
activeborderwidth	Specifies the width of a border drawn around a choice when it is under the mouse. Default is 1 pixel.
activeforeground	The foreground color that will appear on a choice when it is under the mouse.
bg	The background color for choices not under the mouse.
bd	The width of the border around all the choices. Default is 1.
cursor	The cursor that appears when the mouse is over the choices, but only when the menu has been torn off.
disabledforeground	The color of the text for items whose state is DISABLED.
font	The default font for textual choices.
fg	The foreground color used for choices not under the mouse.



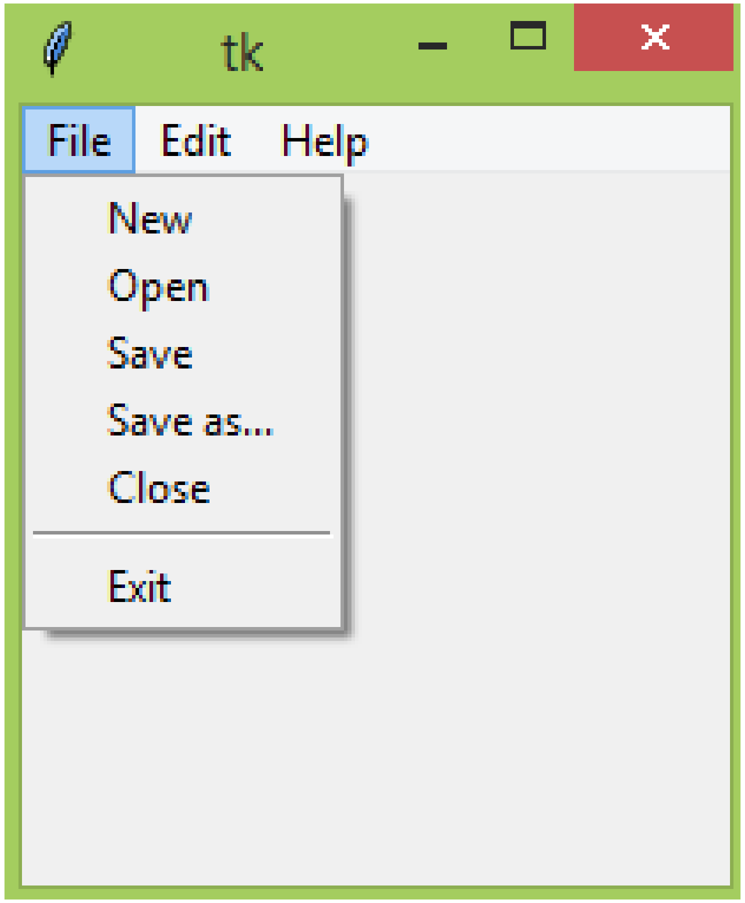
relief	The default 3-D effect for menus is relief=RAISED.
image	To display an image on this menubutton.
selectcolor	Specifies the color displayed in checkbuttons and radiobuttons when they are selected.
tearoff	Normally, a menu can be torn off, the first position (position 0) in the list of choices is 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.
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.



Option	Description
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.
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.

```
from tkinter import *

def donothing():
	filewin = Toplevel(root)
	button = Button(filewin, text="Do nothing button")
	button.pack()

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="Save as...", 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="Delete", 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()
```

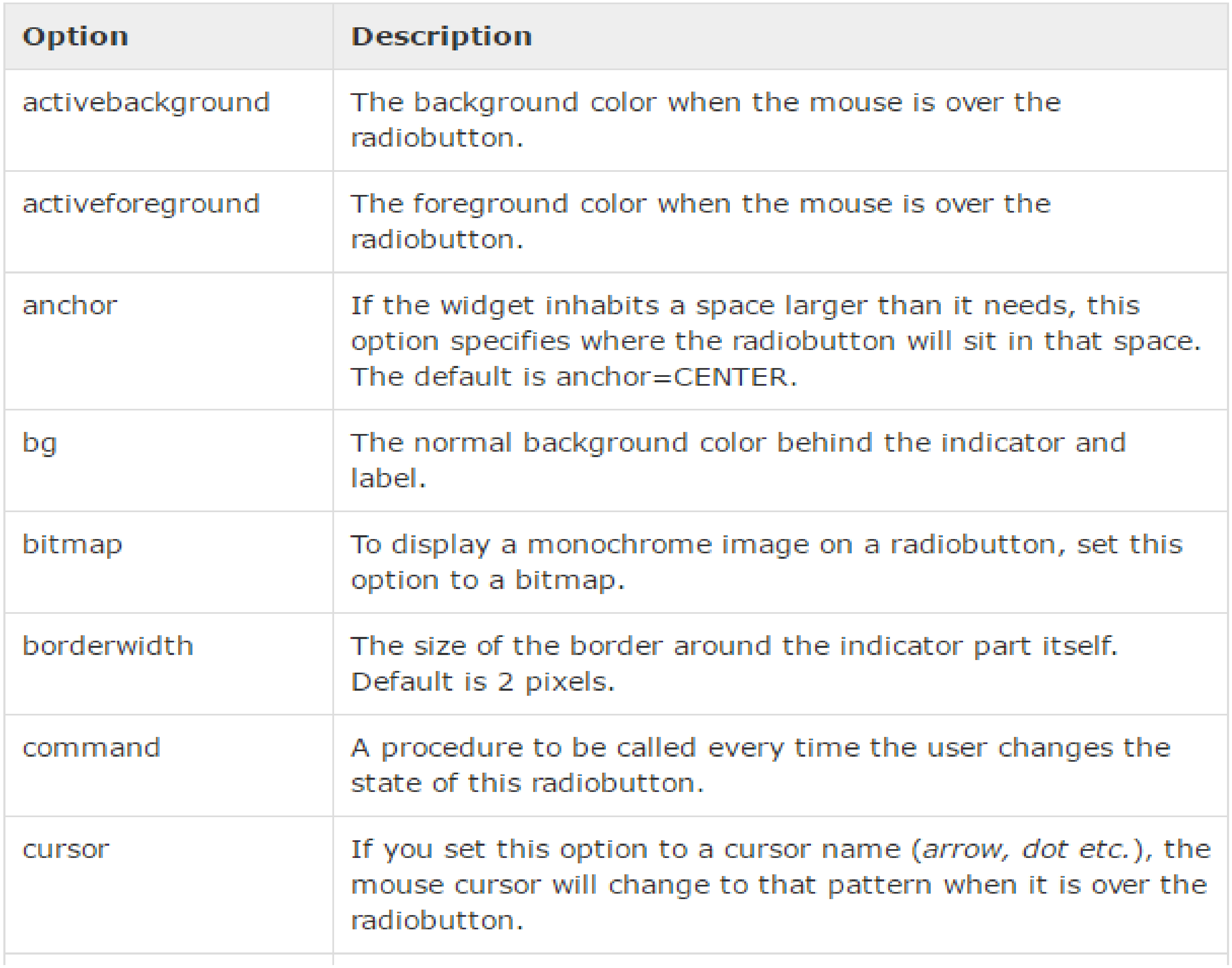

tk
File
Edit
Help
New
Open
Save
Save as...
Close
Exit

Radiobutton widget

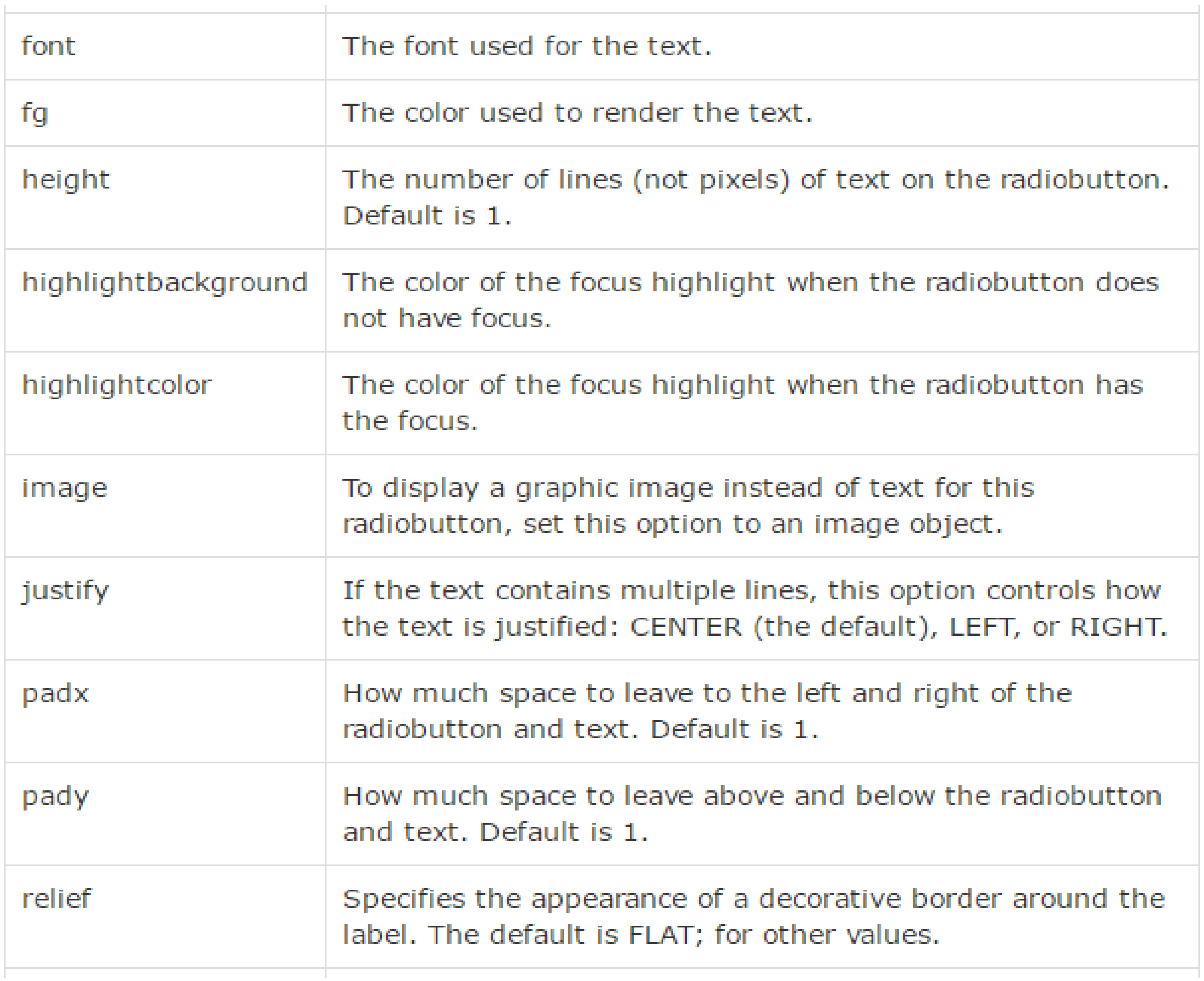
- This widget implements a multiple-choice button, which is a way to offer many possible selections to the user and lets user choose only one of them.

- **Syntax:**

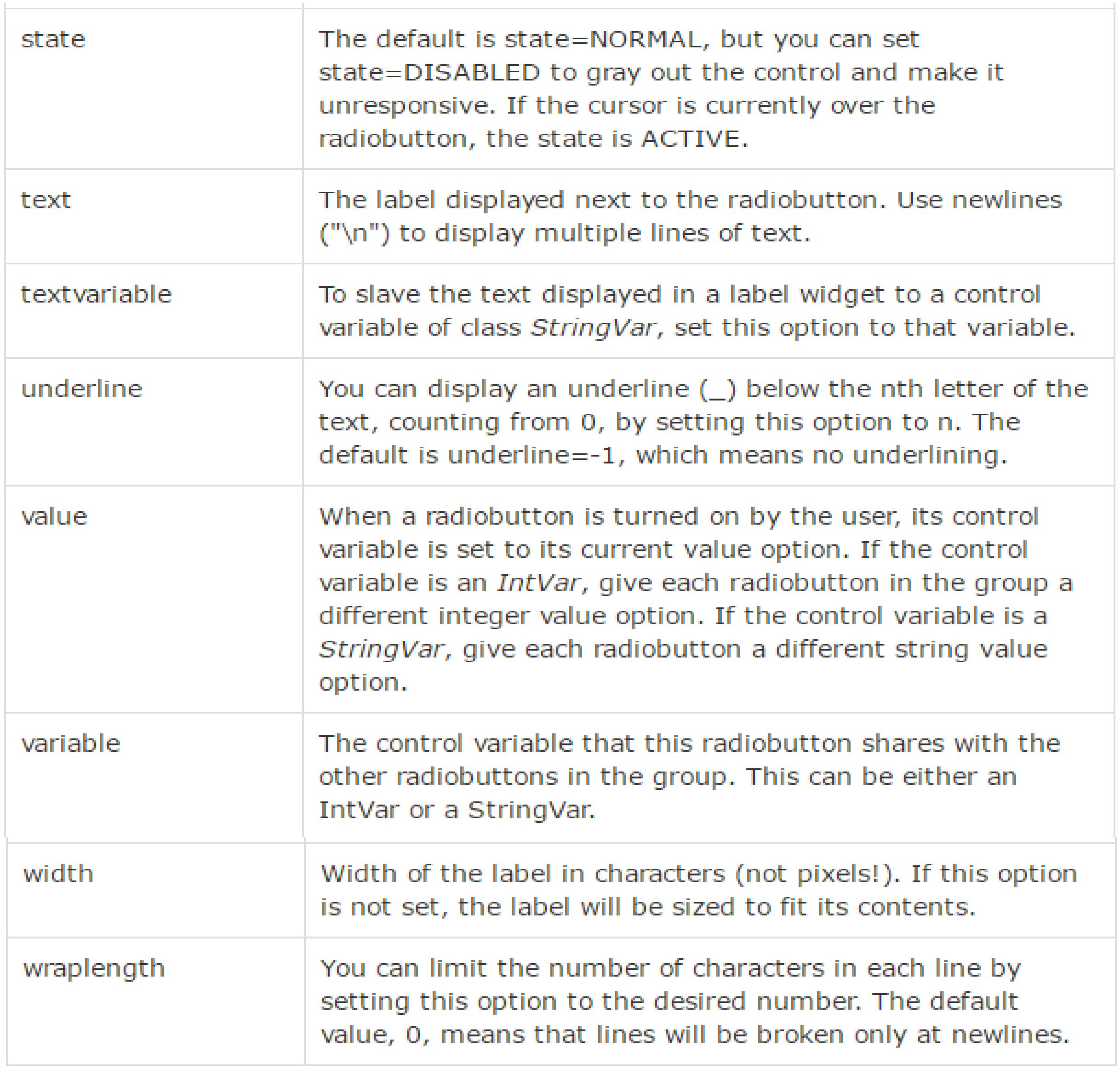
  w = Radiobutton (master, option, ...)



Option	Description
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.
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.

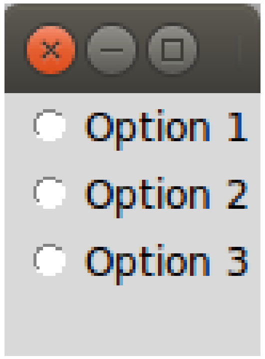


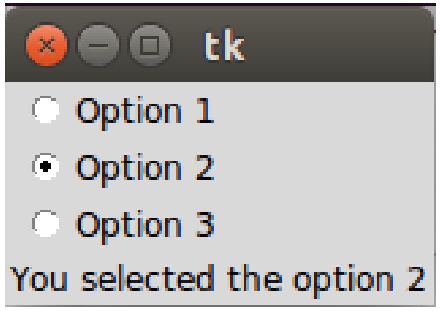
font	The font used for the text.
fg	The color used to render the text.
height	The number of lines (not pixels) of text on the radiobutton. Default is 1.
highlightbackground	The color of the focus highlight when the radiobutton does not have focus.
highlightcolor	The color of the focus highlight when the radiobutton has the focus.
image	To display a graphic image instead of text for this radiobutton, set this option to an image object.
justify	If the text contains multiple lines, this option controls how the text is justified: CENTER (the default), LEFT, or RIGHT.
padx	How much space to leave to the left and right of the radiobutton and text. Default is 1.
pady	How much space to leave above and below the radiobutton and text. Default is 1.
relief	Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.



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

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

root = Tk()
label = Label(root)
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.pack()
root.mainloop()
```





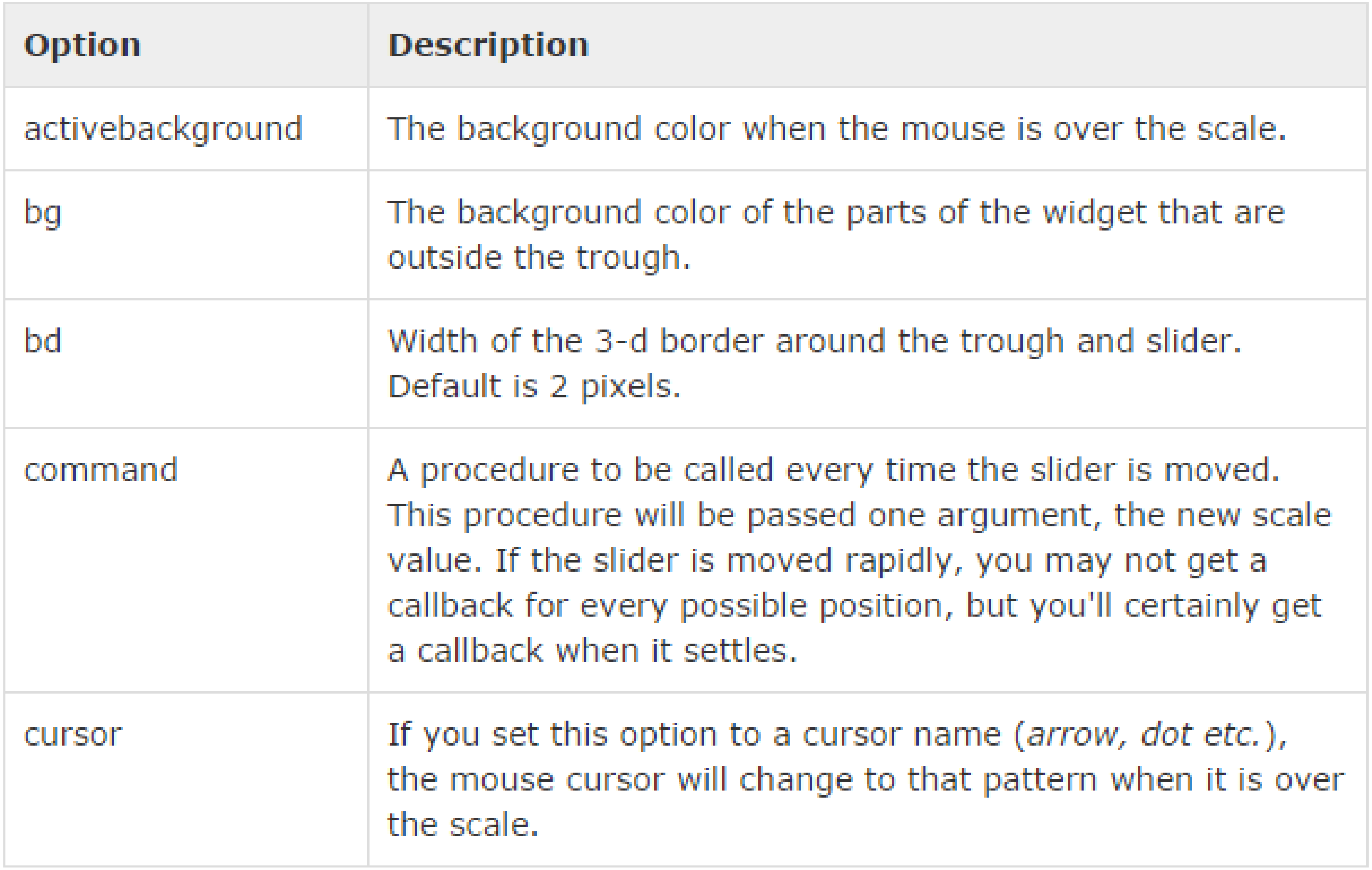
You selected the option 2

Scale widget

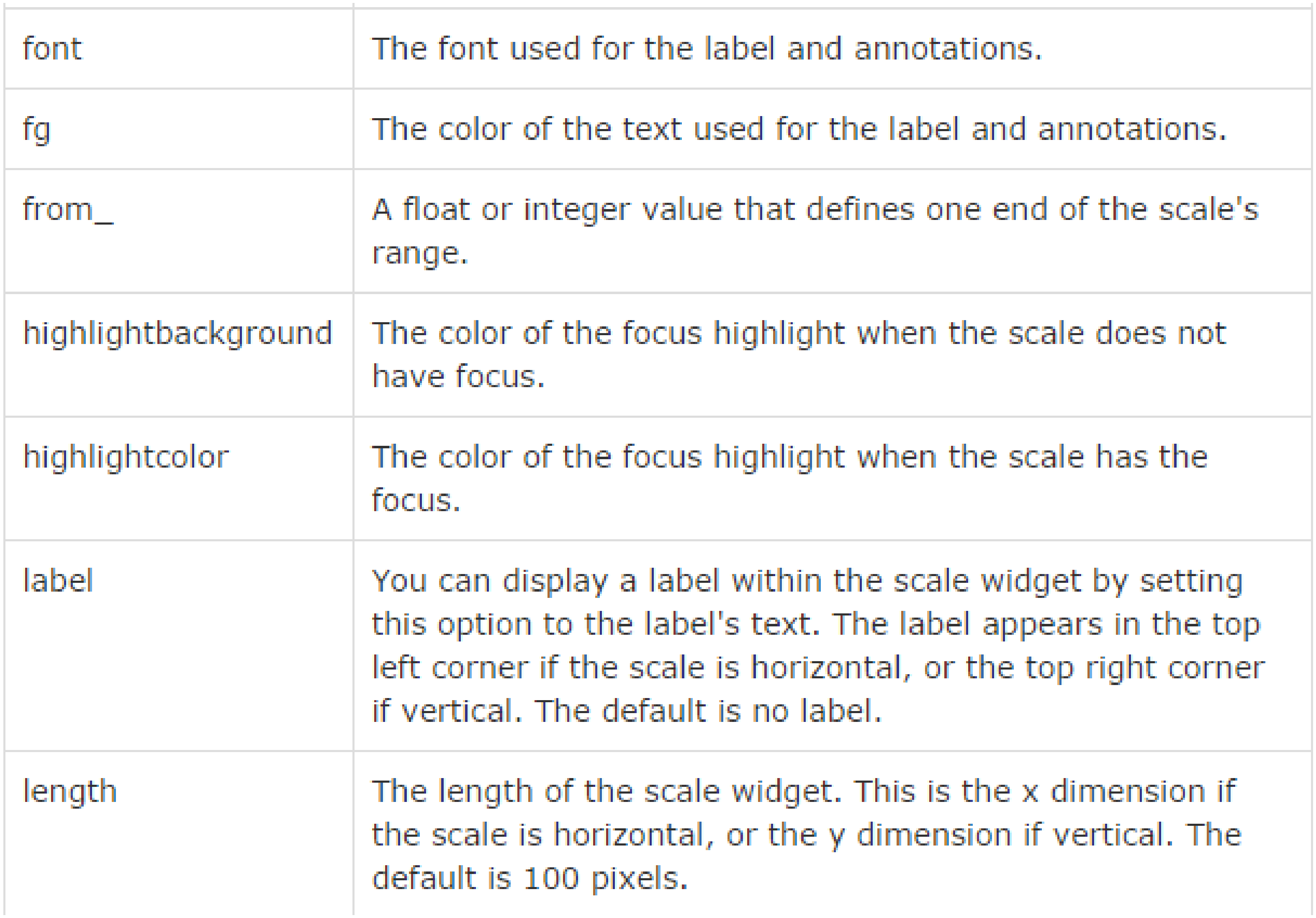
- The Scale widget provides a graphical slider object that allows you to select values from a specific scale.

- **Syntax:**

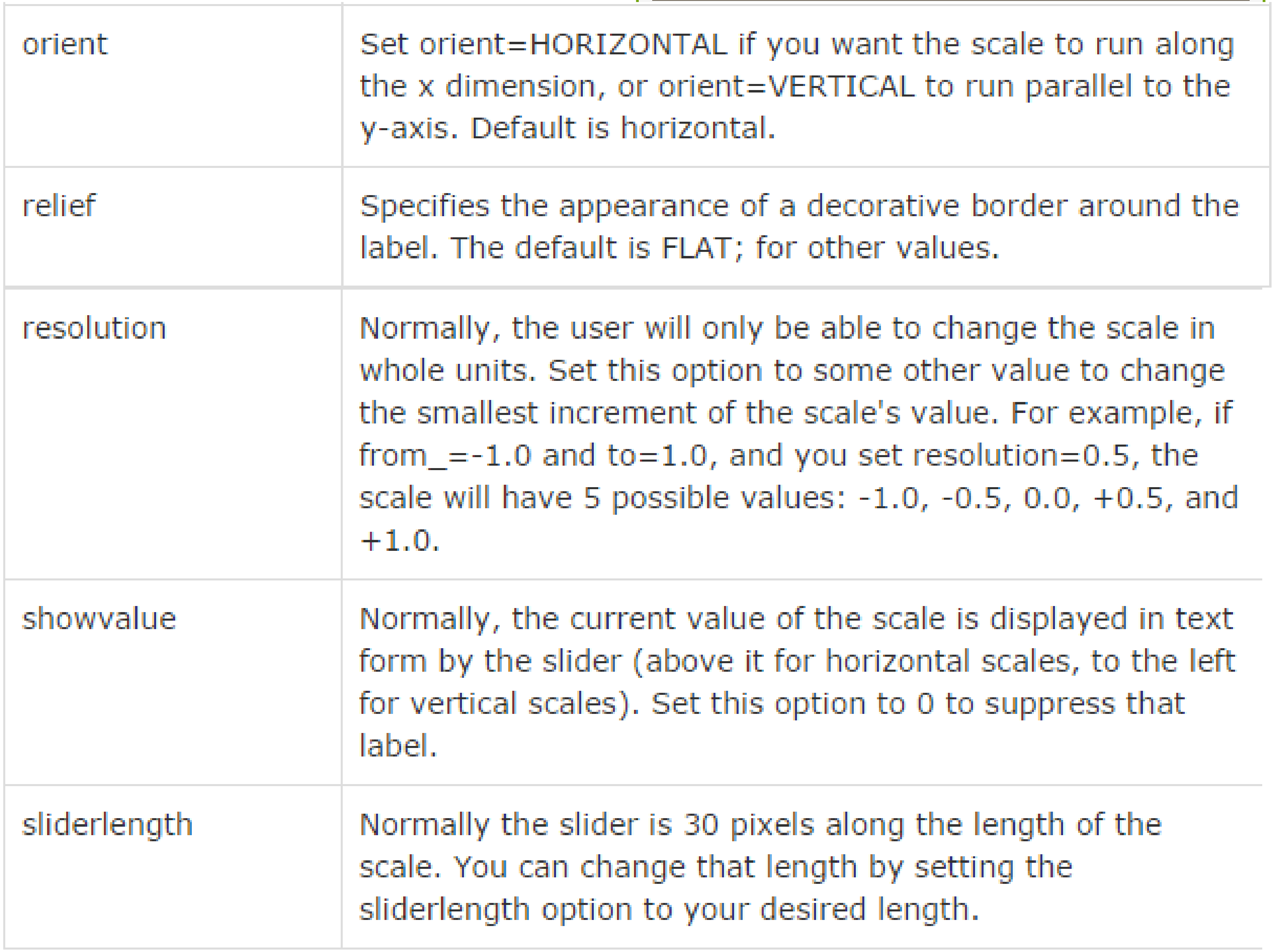
```
w = Scale ( master, option, ... )
```



Option	Description
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.
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.



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.
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.

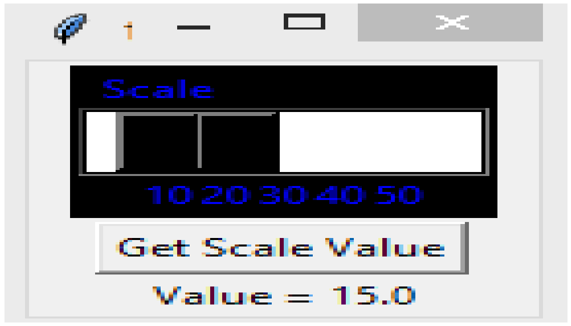


orient	Set orient=HORIZONTAL if you want the scale to run along the x dimension, or orient=VERTICAL to run parallel to the y-axis. Default is horizontal.
relief	Specifies the appearance of a decorative border around the label. The default is FLAT; for other values.
resolution	Normally, the user will 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_=-1.0 and to=1.0, and you set resolution=0.5, the scale will have 5 possible values: -1.0, -0.5, 0.0, +0.5, and +1.0.
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.
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.
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.

troughcolor	The color of the trough.
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.
width	The width of the trough part of the widget. This is the x dimension for vertical scales and the y dimension if the scale has orient=HORIZONTAL. Default is 15 pixels.

```
from tkinter import *

root = Tk()
var=IntVar()
def sel():
   selection = "Value = " + str(var.get())
   label.config(text = selection)

scale = Scale(root,variable=var,activebackground='red',bd=2,bg='black',\
fg='blue',label='Scale',orient='horizontal',from_=10,to_=50,\
resolution=5,tickinterval=10,showvalue=0,sliderlength=40,troughcolor='white',
    width=30)
scale.pack()

button = Button(root, text="Get Scale Value", command=sel)
button.pack()

label = Label(root)
label.pack()

root.mainloop()
```



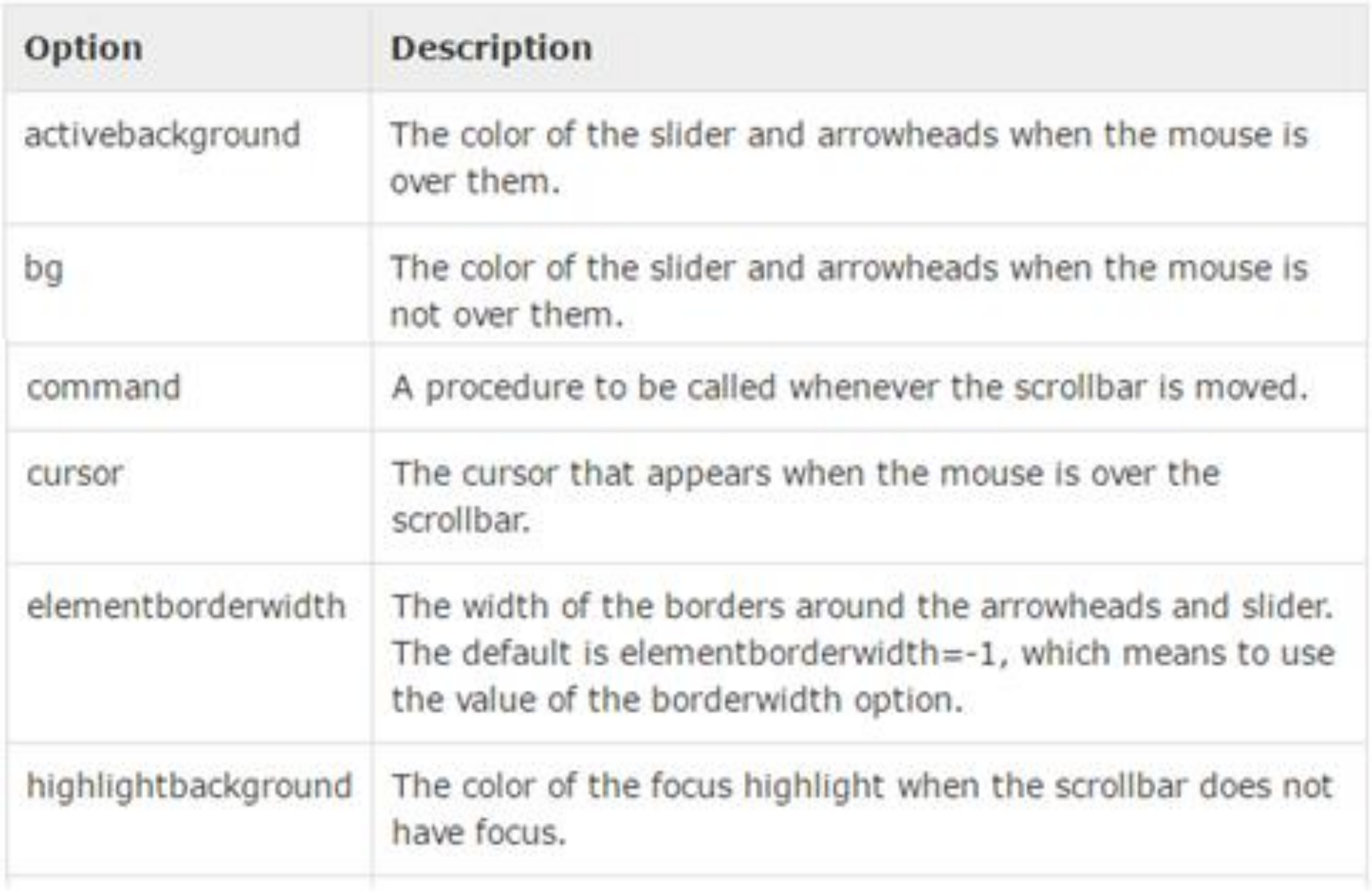
Value = 15.0

Scrollbar widget

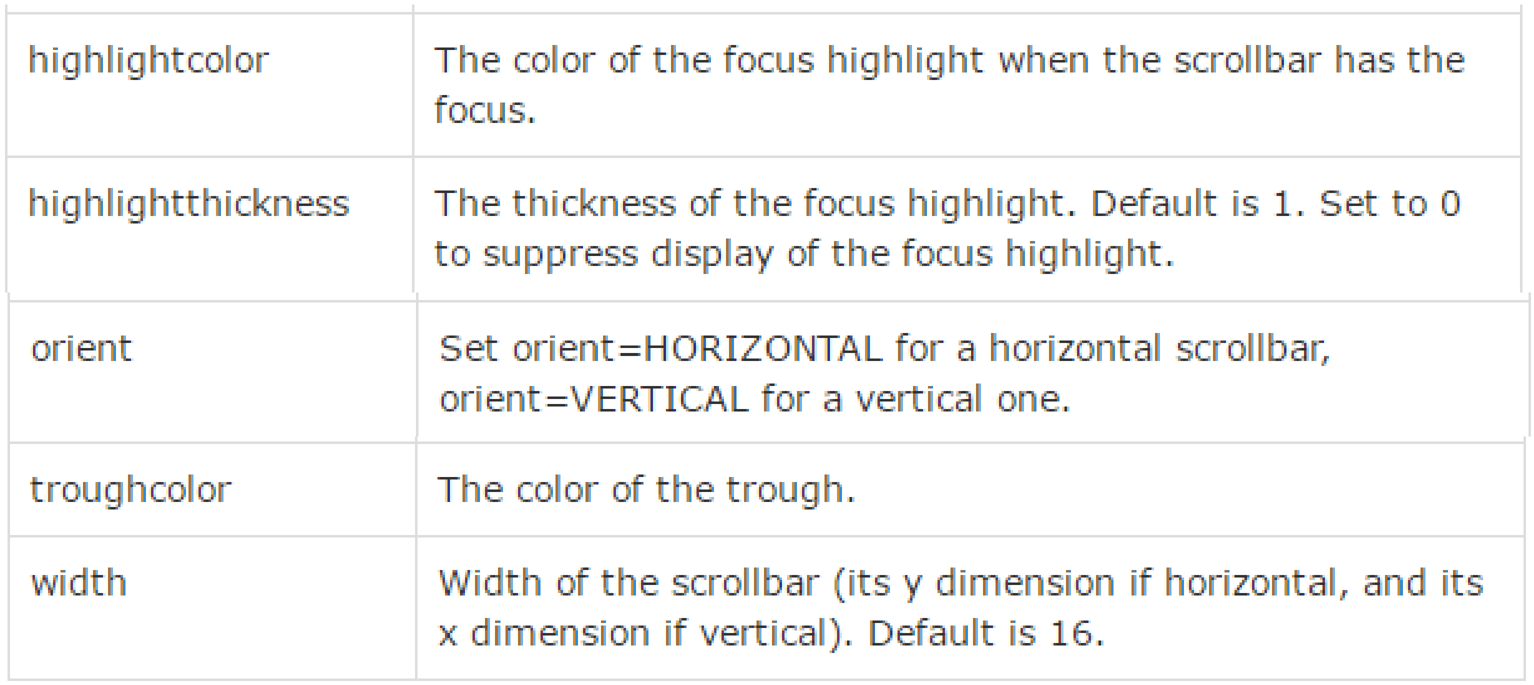
- This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text and Canvas. Note that you can also create horizontal scrollbars on Entry widgets.

- **Syntax:**

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

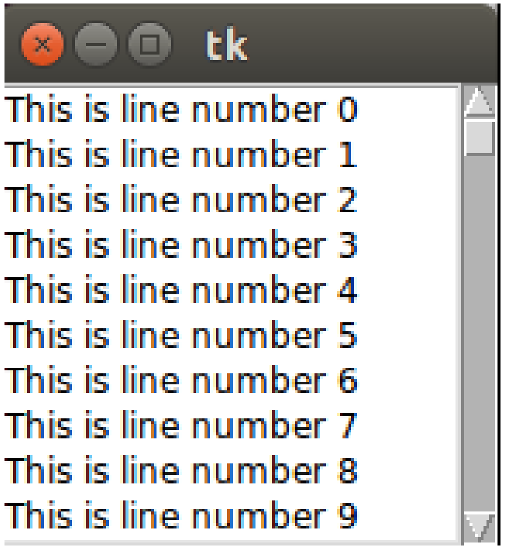


Option	Description
activebackground	The color of the slider and arrowheads when the mouse is over them.
bg	The color of the slider and arrowheads when the mouse is not over them.
command	A procedure to be called whenever the scrollbar is moved.
cursor	The cursor that appears when the mouse is over the scrollbar.
elementborderwidth	The width of the borders around the arrowheads and slider. The default is elementborderwidth=-1, which means to use the value of the borderwidth option.
highlightbackground	The color of the focus highlight when the scrollbar does not have focus.



highlightcolor	The color of the focus highlight when the scrollbar has the focus.
highlightthickness	The thickness of the focus highlight. Default is 1. Set to 0 to suppress display of the focus highlight.
orient	Set orient=HORIZONTAL for a horizontal scrollbar, orient=VERTICAL for a vertical one.
troughcolor	The color of the trough.
width	Width of the scrollbar (its y dimension if horizontal, and its x dimension if vertical). Default is 16.

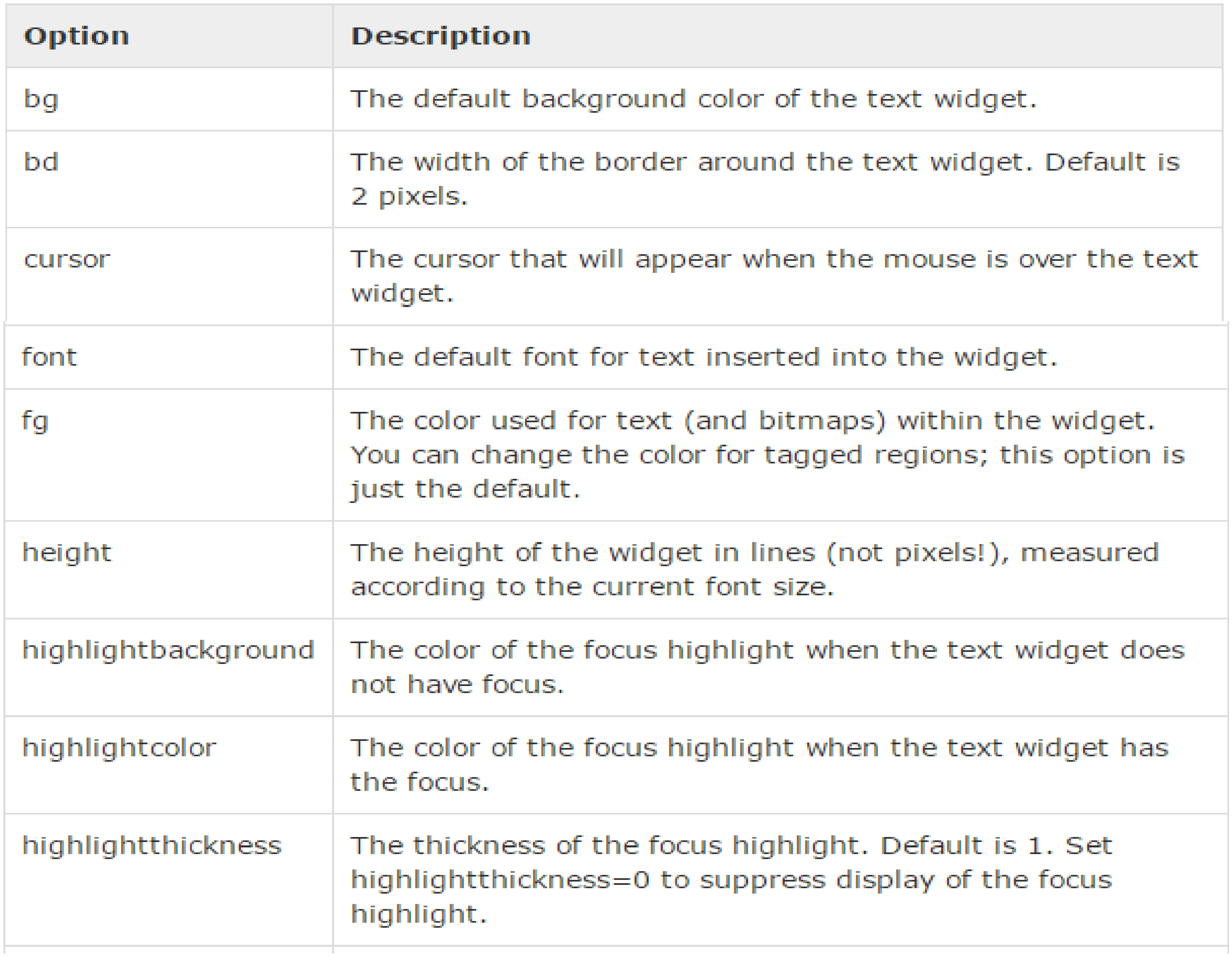
```
from tkinter import *

root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT,fill=Y)

mylist = Listbox(root, yscrollcommand = scrollbar.set )
for line in range(0,101):
    mylist.insert(END,"This is line number " + str(line))

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



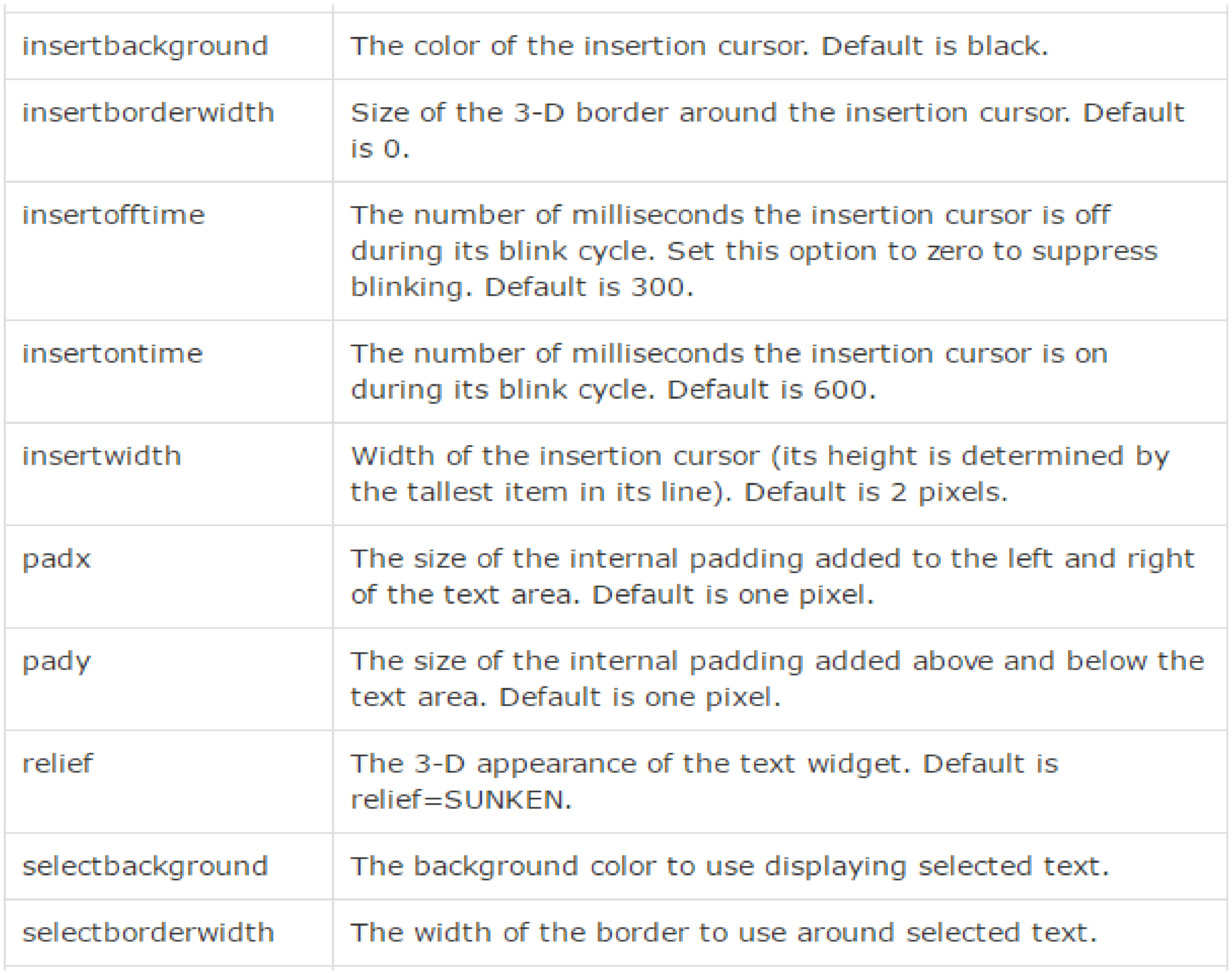
```
This is line number 0
This is line number 1
This is line number 2
This is line number 3
This is line number 4
This is line number 5
This is line number 6
This is line number 7
This is line number 8
This is line number 9
```

Text Widget

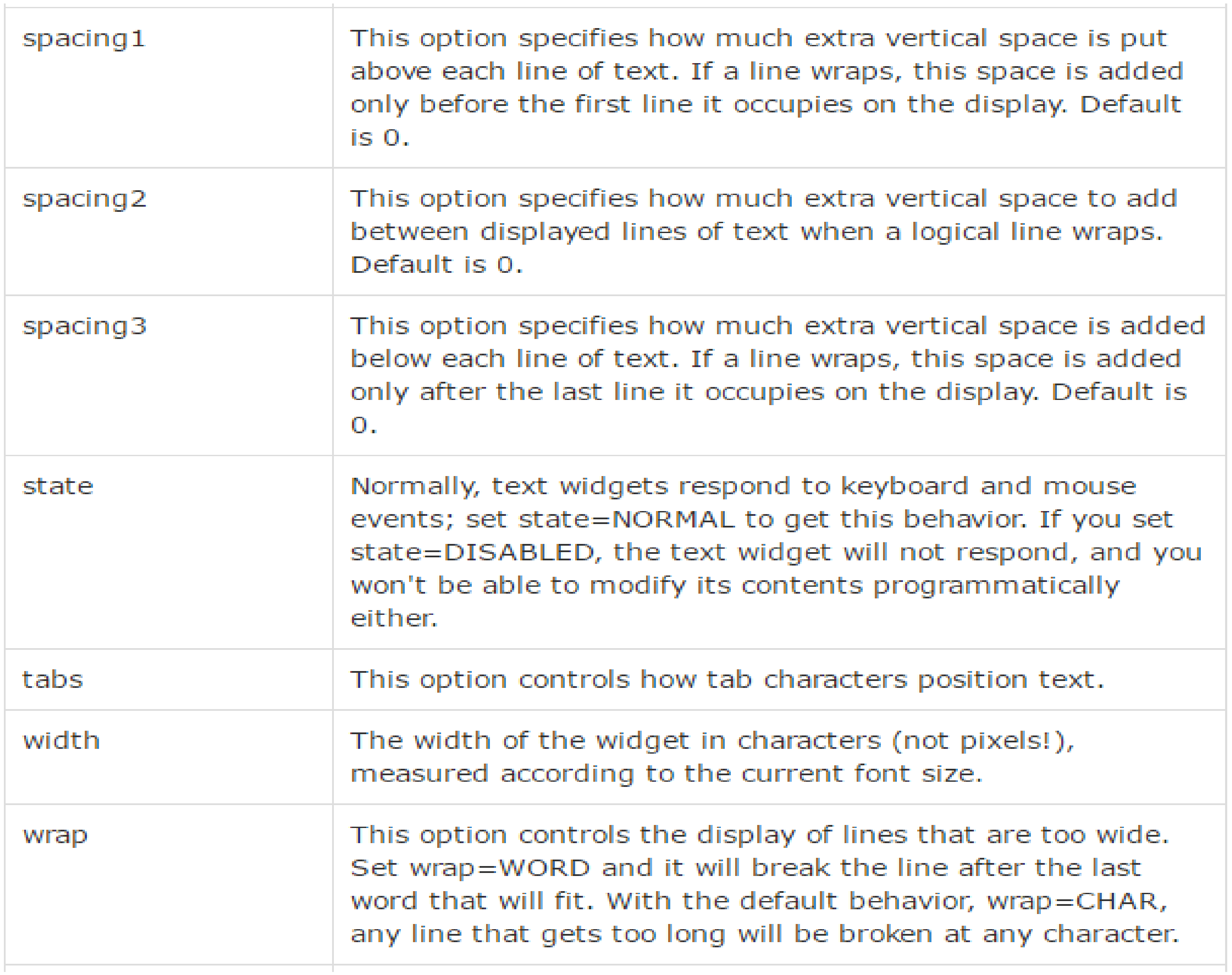
- A text widget is used for multi-line text area. The Tkinter text widget is very powerful and flexible and can be used for a wide range of tasks.
- Though one of the main purposes is to provide simple multi-line areas, as they are often used in forms, text widgets can also be used as simple text editors.
- Text widgets can be used to display links, images, and HTML, even using CSS styles.



Option	Description
bg	The default background color of the text widget.
bd	The width of the border around the text widget. Default is 2 pixels.
cursor	The cursor that will appear when the mouse is over the text widget.
font	The default font for text inserted into the widget.
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.
height	The height of the widget in lines (not pixels!), measured according to the current font size.
highlightbackground	The color of the focus highlight when the text widget does not have focus.
highlightcolor	The color of the focus highlight when the text widget has the focus.
highlightthickness	The thickness of the focus highlight. Default is 1. Set highlightthickness=0 to suppress display of the focus highlight.

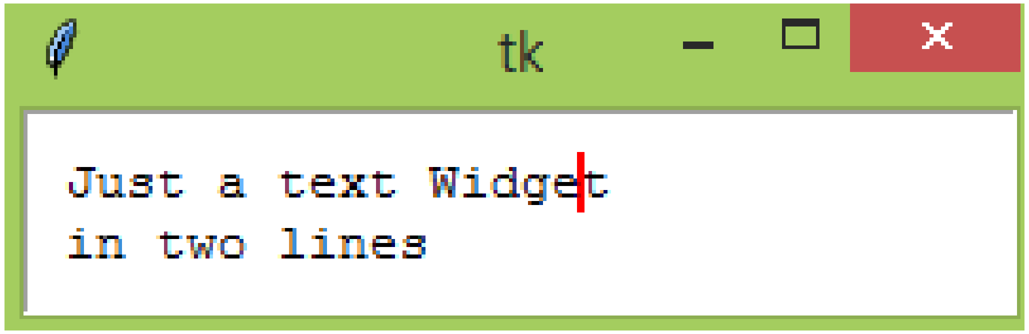


insertbackground	The color of the insertion cursor. Default is black.
insertborderwidth	Size of the 3-D border around the insertion cursor. Default is 0.
insertofftime	The number of milliseconds the insertion cursor is off during its blink cycle. Set this option to zero to suppress blinking. Default is 300.
insertontime	The number of milliseconds the insertion cursor is on during its blink cycle. Default is 600.
insertwidth	Width of the insertion cursor (its height is determined by the tallest item in its line). Default is 2 pixels.
padx	The size of the internal padding added to the left and right of the text area. Default is one pixel.
pady	The size of the internal padding added above and below the text area. Default is one pixel.
relief	The 3-D appearance of the text widget. Default is relief=SUNKEN.
selectbackground	The background color to use displaying selected text.
selectborderwidth	The width of the border to use around selected text.



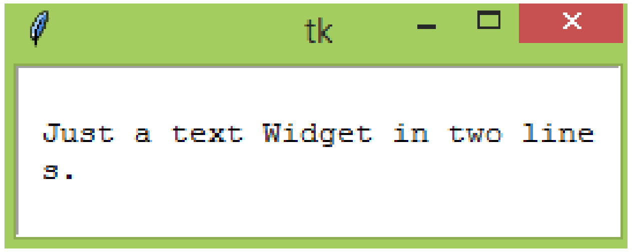
spacing1	This option specifies how much extra vertical space is put above each line of text. If a line wraps, this space is added only before the first line it occupies on the display. Default is 0.
spacing2	This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. Default is 0.
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.
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.
tabs	This option controls how tab characters position text.
width	The width of the widget in characters (not pixels!), measured according to the current font size.
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.

```
from tkinter import *

root = Tk()
T = Text(root, height=2, width=30, insertbackground="RED",
insertofftime=1600)
T.pack()
T.insert(END, "Just a text Widget\nin two lines\n")
mainloop()
```



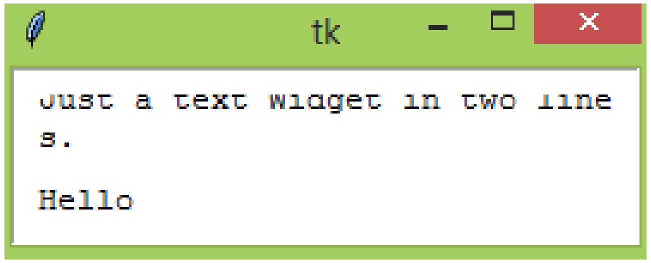
```
Just a text Widget
in two lines
```

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30,padx=10,pady=10,spacing1=10)
T.pack()
T.insert(END, "Just a text Widget in two lines.")
mainloop()
```



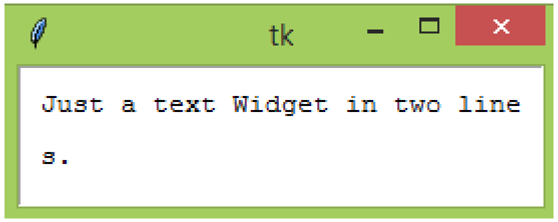
```
Just a text Widget in two line
s.
```

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30,padx=10,pady=10,spacing1=10)
T.pack()
T.insert(END, "Just a text Widget in two lines.\nHello")
mainloop()
```



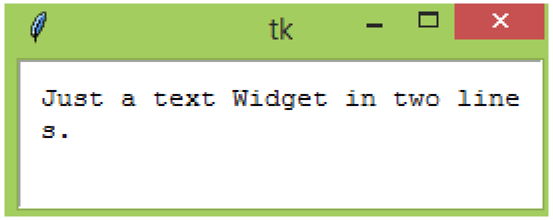
```
Just a text Widget in two line
s.

Hello
```

```
from tkinter import *

root = Tk()
T = Text(root, height=3,
width=30,padx=10,pady=10,spacing2=10)
T.pack()
T.insert(END, "Just a text Widget in two lines.")
mainloop()
```



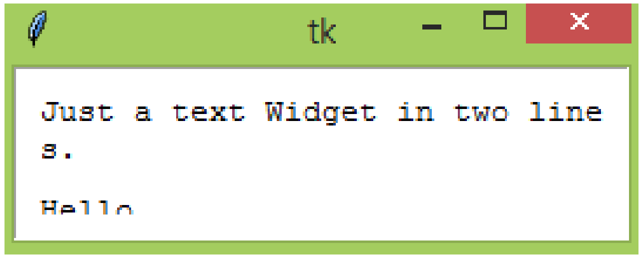
```
Just a text Widget in two line
s.
```

```
from tkinter import *

root = Tk()
T = Text(root, height=2, width=30,padx=10,pady=10,spacing3=10)
T.pack()
T.insert(END, "Just a text Widget in two lines.")
mainloop()
```



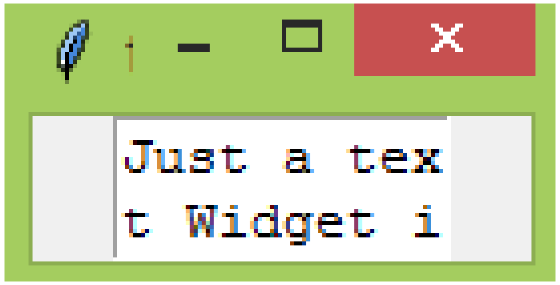
```
Just a text Widget in two line
s.
```

```
from tkinter import *
root = Tk()
T = Text(root, height=2, width=30,padx=10,pady=10,spacing3=10)
T.pack()
T.insert(END, "Just a text Widget in two lines.\nHello")
mainloop()
```



```
Just a text Widget in two line
s.
Hello
```

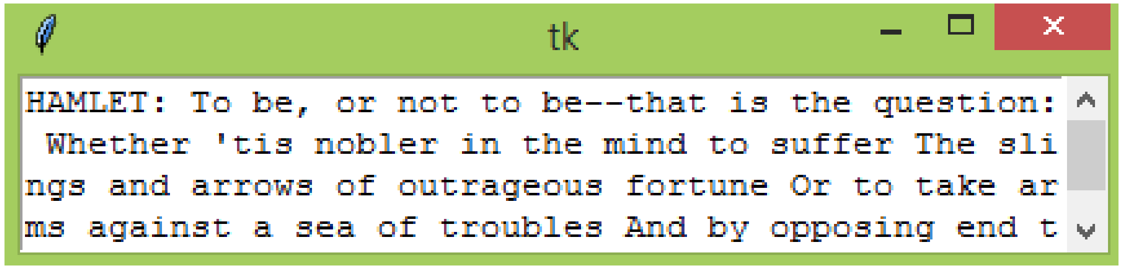
```
from tkinter import *

root = Tk()
T = Text(root, height=2, width=10)
T.pack()
T.insert(END, "Just a text Widget in line")
mainloop()
```



```
Just a tex
t Widget i
```

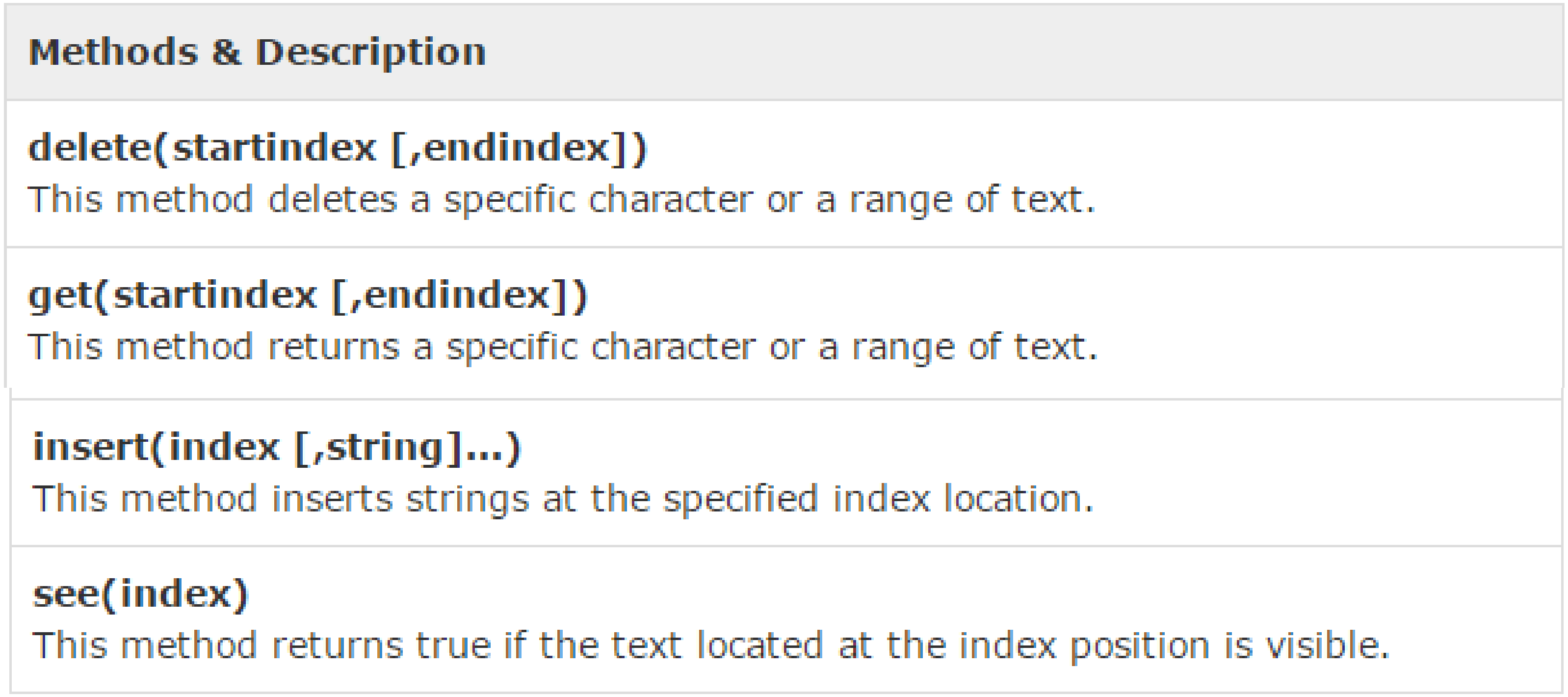
Using scrollbar

```
from tkinter import *

root = Tk()
S = Scrollbar(root)
T = Text(root, height=4, width=50)
S.pack(side=RIGHT, fill=Y)
T.pack()
S.config(command=T.yview)
T.config(yscrollcommand=S.set)
quote = """HAMLET: To be, or not to be--that is the question:
Whether 'tis nobler in the mind to suffer The slings and
arrows of outrageous fortune Or to take arms against a sea
of troubles And by opposing end them. To die, to sleep”””
T.insert(END, quote)
mainloop()
```



```
HAMLET: To be, or not to be--that is the question:
 Whether 'tis nobler in the mind to suffer The sli
ngs and arrows of outrageous fortune Or to take ar
ms against a sea of troubles And by opposing end t
```

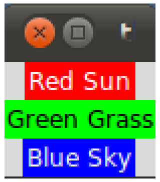
Text Methods



Methods & Description
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.
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.

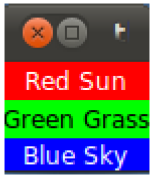
pack() method

- pack()

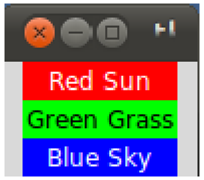


- pack(fill=X)

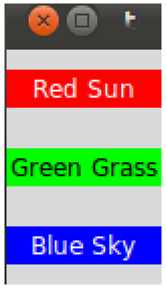
#fill=Y, fill=both)



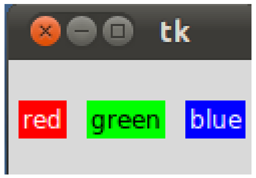
- pack(fill=X,padx=10)



- pack(fill=X,pady=10)



- pack(padx=5, pady=10, side=LEFT)



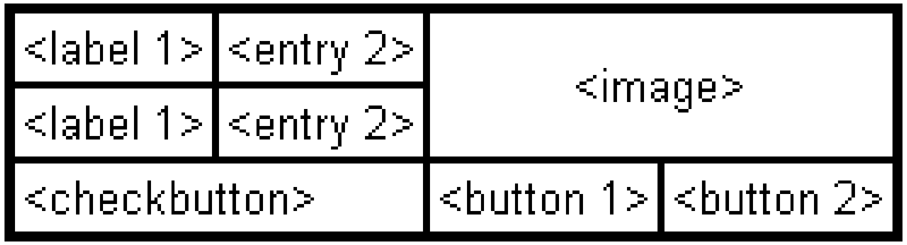
- The **expand** option tells the manager to assign additional space to the widget box. If the parent widget is made larger than necessary to hold all packed widgets, any exceeding space will be distributed among all widgets that have the **expand** option set to a non-zero value.

- Try: pack(expand=1)

Grid geometry manager

- This geometry manager organizes widgets in a table-like structure in the parent widget.
- **When to use the Grid Manager?**

  Consider the following example:



Creating this layout using the pack manager is possible, but it takes a number of extra frame widgets, and a lot of work to make things look good. If you use the grid manager instead, you only need one call per widget to get everything laid out properly.

- Syntax:

widget.grid(grid_options)

column : The column to put widget in; default 0 (leftmost column).

columnspan: How many columns widget occupies; default 1.

padx, pady : How many pixels to pad widget, horizontally and vertically, outside v's borders.

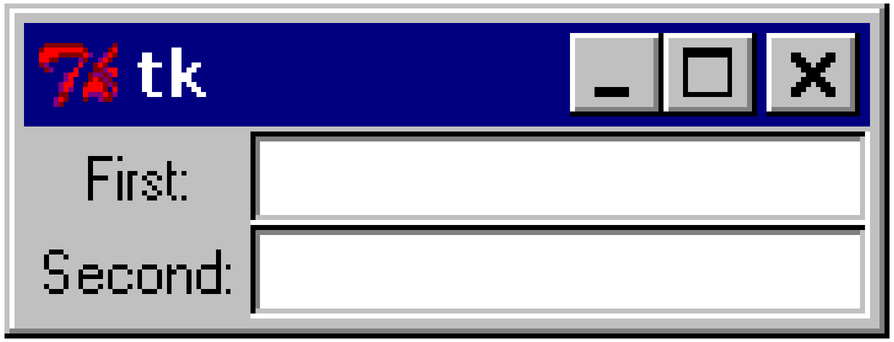
row: The row to put widget in; default the first row that is still empty.

rowspan : How many rows widget occupies; default 1.

- **Patterns**

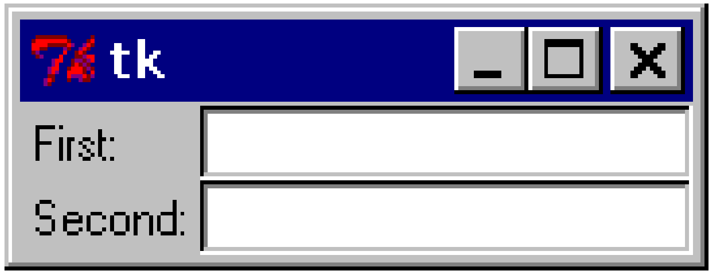
```
Label(master, text="First").grid(row=0) Label(master, text="Second").grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1) e2.grid(row=1, column=1)
```

Note that the column number defaults to 0 if not given.



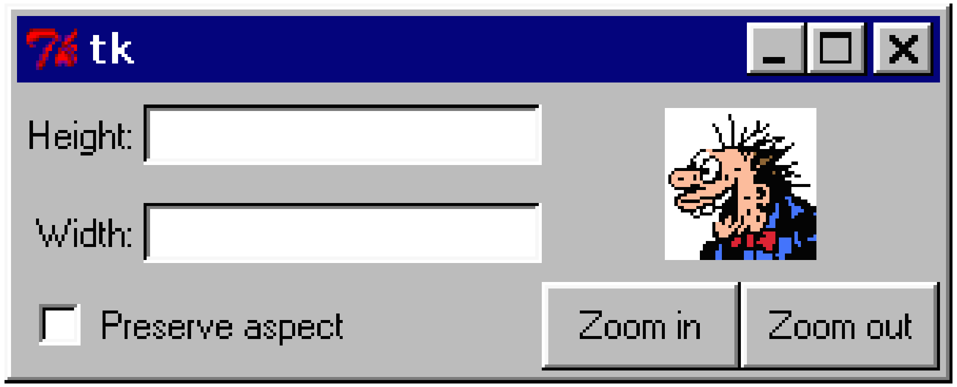
- Note that the widgets are centered in their cells. You can use the **sticky** option to change this; this option takes one or more values from the set **N**, **S**, **E**, **W**. To align the labels to the left border, you could use **W** (west):

```
Label(master,text="First").grid(row=0,sticky=W)
Label(master, text="Second").grid(row=1, sticky=W)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
```

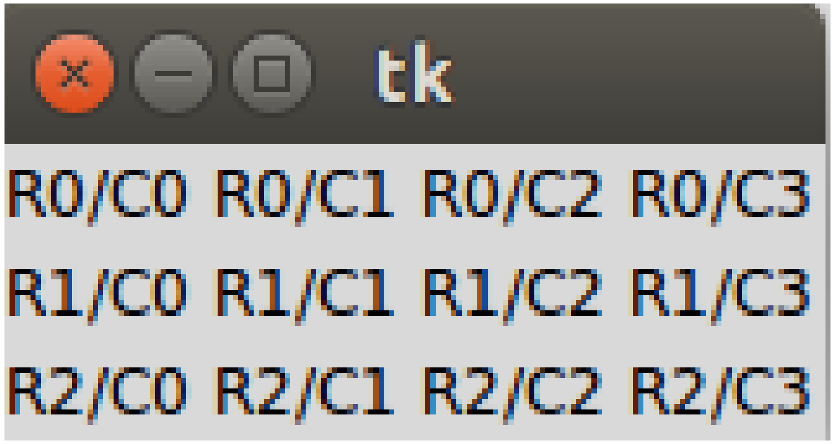


- **Use of columnspan and rowspan:**

```
label1.grid(sticky=E)
label2.grid(sticky=E)
entry1.grid(row=0, column=1)
entry2.grid(row=1, column=1)
checkbutton.grid(columnspan=2, sticky=W)
image.grid(row=0, column=2, columnspan=2, rowspan=2,
sticky=W+E+N+S, padx=5, pady=5)
button1.grid(row=2, column=2)
button2.grid(row=2, column=3)
```



```
import tkinter
root = tkinter.Tk(  )
for r in range(3):
    for c in range(4):
            tkinter.Label(root, text='R%s/C%s'%(r,c)).grid(row=r,column=c)
root.mainloop()
```

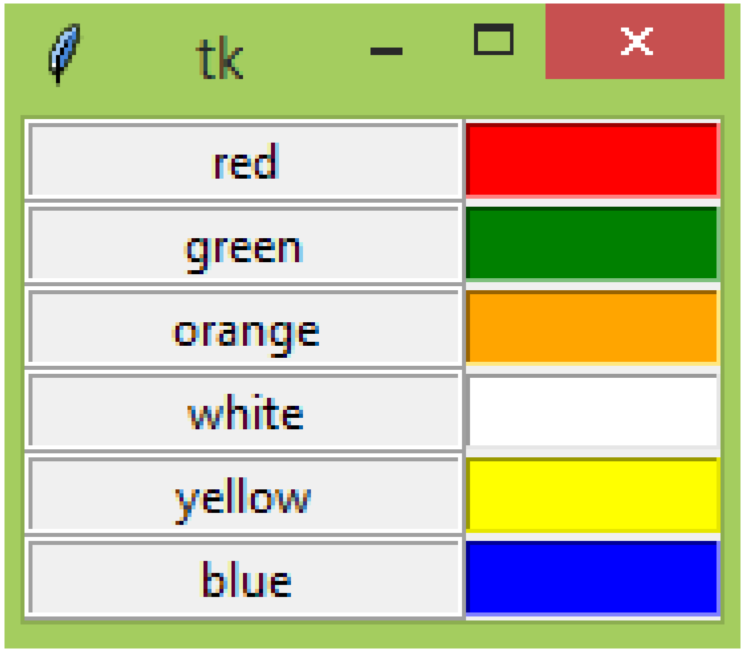


```
R0/C0 R0/C1 R0/C2 R0/C3
R1/C0 R1/C1 R1/C2 R1/C3
R2/C0 R2/C1 R2/C2 R2/C3
```

```
import tkinter
from tkinter import *

colours = ['red','green','orange','white','yellow','blue']
r = 0
for c in colours:
    Label(text=c, relief=RIDGE,width=15).grid(row=r,column=0)
    Entry(bg=c, relief=SUNKEN,width=10).grid(row=r,column=1)
    r = r + 1

mainloop()
```

Program

- Output



```
tk
red      🟥
green    🟩
orange   🟧
white    ⬜
yellow   🟨
blue     🟦
```

Place geometry manager

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

```
widget.place( place_options )
```

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

top = tkinter.Tk()

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

B = tkinter.Button(top, text ="Hello", command = helloCallBack)

B.pack()
B.place(x=10,y=10,height=100, width=100)
top.mainloop()
```