Python 3 - 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. These options can be used as key-value pairs separated by commas.

Sr.No.	Option & 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.

Methods

These methods are available on Menu objects −

Sr.No.	Option & Description
1	**add_command (options)** Adds a menu item to the menu.
2	**add_radiobutton(options)** Creates a radio button menu item.
3	**add_checkbutton(options)** Creates a check button menu item.
4	**add_cascade(options)** Creates a new hierarchical menu by associating a given menu to a parent menu
5	**add_separator()** Adds a separator line to the menu.
6	**add(type, options)** Adds a specific type of menu item to the menu.
7	**delete(startindex [, endindex])** Deletes the menu items ranging from startindex to endindex.
8	**entryconfig(index, options)** Allows you to modify a menu item, which is identified by the index, and change its options.
9	**index(item)** Returns the index number of the given menu item label.
10	**insert_separator (index)** Insert a new separator at the position specified by index.
11	**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.
12	type (index) Returns the type of the choice specified by index: either "cascade", "checkbutton", "command", "radiobutton", "separator", or "tearoff".

Example

Try the following example yourself −

```
# !/usr/bin/python3
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()
```

Result

When the above code is executed, it produces the following result −