**TKINTER**

**WHAT IS TKINTER?**

Tkinter is an acronym for "Tk interface". Tkinter was developed as a GUI extension for the Tcl scripting language by John Ousterhout.

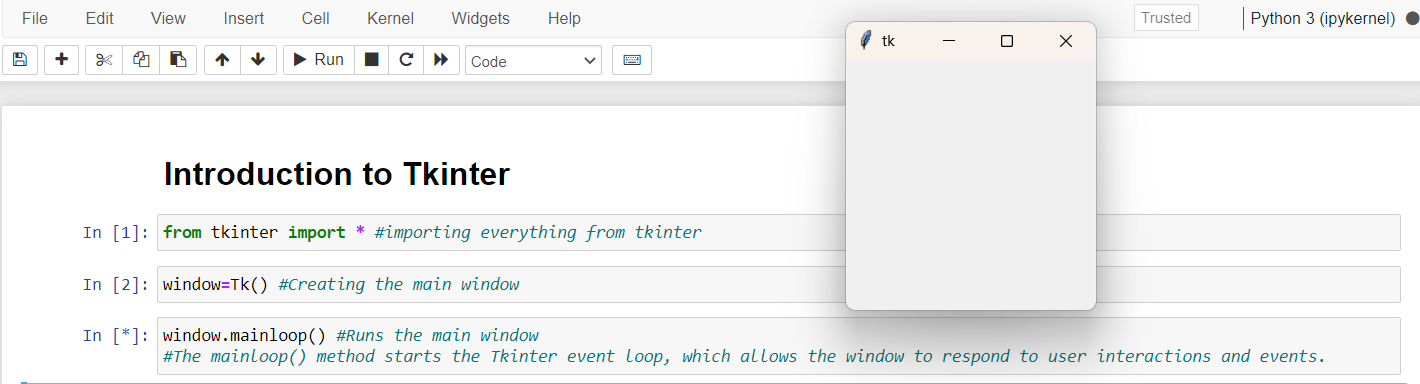
Tkinter is a Python library that provides a simple and efficient way to create graphical user interfaces (GUIs) for desktop based applications. It is a standard library included with Python, so there is no need to install any additional packages.

It provides a set of tools and widgets that allow you to create windows, buttons, menus, text boxes, and other GUI elements for your Python applications.

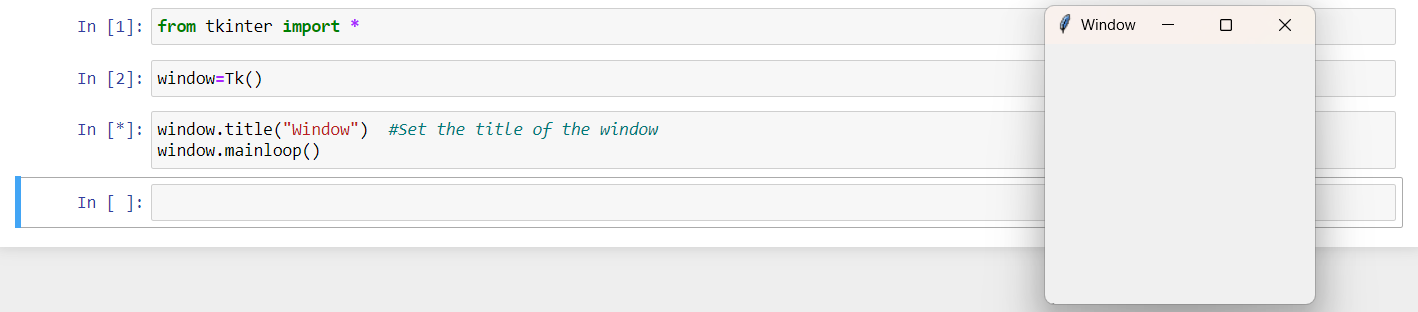
**HOW DO WE USE TKINTER?**

* import the Tkinter module.
* Create the main application window.
* Add the widgets like labels, buttons, frames, etc. to the window.
* Call the main event loop so that the actions can take place on the user's computer screen.

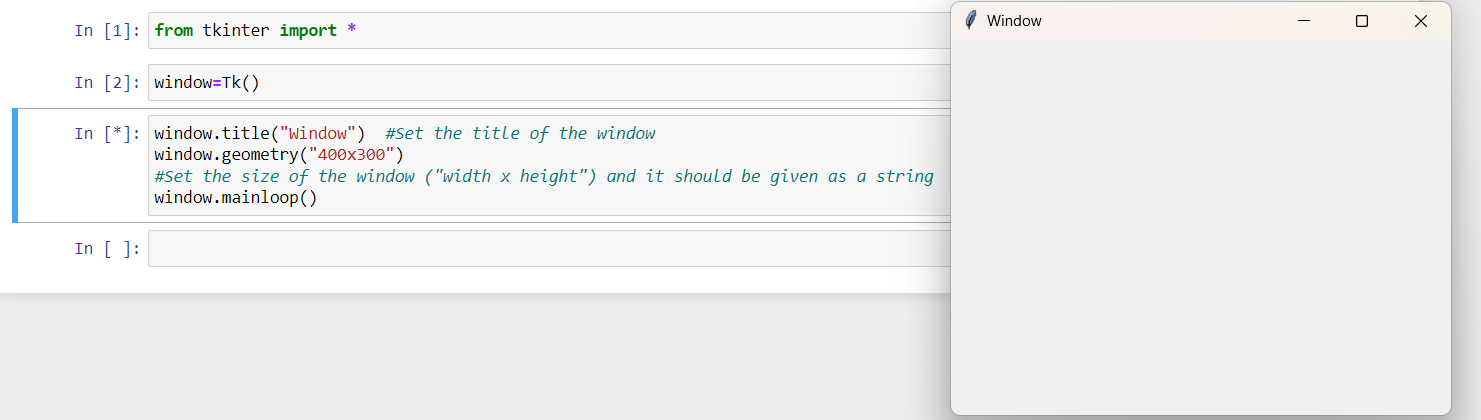
Example:

****

* Firstly, we’ve imported everything from tkinter module.
* Secondly, we created a main window by using Tk( ) initializer and stored it in a variable window.
* Finally, to display the window, we run the main window by using mainloop( ) which starts the tkinter event loop and responds to the user interactions.
* We can also set title to the window by using title( ) attribute:



* We can even set dimensions to the window using geometry( ) attribute :



* We can further customize the window by adding other widgets, such as buttons, labels, entry fields, etc.

**TKINTER WIDGETS:**

Tkinter provides a variety of widgets that you can use to create graphical user interfaces (GUIs).

|  |  |  |
| --- | --- | --- |
| SNo | WIDGET | DESCRIPTION |
|  | Button | The Button is used to add various kinds of buttons to the python application. |
|  | Canvas | The canvas widget is used to draw the canvas on the window. |
|  | Checkbutton | The Checkbutton is used to display the CheckButton on the window. |
|  | Entry | The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values. |
|  | Frame | It can be defined as a container to which, another widget can be added and organized. |
|  | Label | A label is a text used to display some message or information about the other widgets. |
|  | ListBox | The ListBox widget is used to display a list of options to the user. |
|  | Menubutton | The Menubutton is used to display the menu items to the user. |
|  | Menu | It is used to add menu items to the user. |
|  | Message | The Message widget is used to display the message-box to the user. |
|  | Radiobutton | The Radiobutton is different from a checkbutton. Here, the user is provided with various options and the user can select only one option among them. |
|  | Scale | It is used to provide the slider to the user. |
|  | Scrollbar | It provides the scrollbar to the user so that the user can scroll the window up and down. |
|  | Text | It is different from Entry because it provides a multi-line text field to the user so that the user can write the text and edit the text inside it. |
|  | Toplevel | It is used to create a separate window container. |
|  | Spinbox | It is an entry widget used to select from options of values. |
|  | PanedWindow | It is like a container widget that contains horizontal or vertical panes. |
|  | LabelFrame | A LabelFrame is a container widget that acts as the container |
|  | Messagebox | This module is used to display the message-box in the desktop based applications. |

**Button:**

* The button widget is used to add various types of buttons to the python application. Python allows us to configure the look of the button according to our requirements. Various options can be set or reset depending upon the requirements.
* We can also associate a method or function with a button which is called when the button is pressed.

Syntax:

**W = Button(parent,options)**

Example:



List of possible options used in Button widget:

* **activebackground**: It represents the background of the button when the mouse hover the button.
* **activeforeground**: It represents the font color of the button when the mouse hover the button.
* **Bd**: It represents the border width in pixels.
* **Bg**: It represents the background color of the button.
* **Command**: It is set to the function call which is scheduled when the function is called.
* **Fg**: Foreground color of the button.
* **Font**: The font of the button text.
* **Height**: The height of the button. The height is represented in the number of text lines for the textual lines or the number of pixels for the images.
* **Highlightcolor**: The color of the highlight when the button has the focus.
* **Image**: It is set to the image displayed on the button.
* **justify**: It illustrates the way by which the multiple text lines are represented. It is set to LEFT for left justification, RIGHT for the right justification, and CENTER for the center.
* **Padx**: Additional padding to the button in the horizontal direction.
* **pady**: Additional padding to the button in the vertical direction.
* **Relief**: It represents the type of the border. It can be SUNKEN, RAISED, GROOVE, and RIDGE.
* **State**: This option is set to DISABLED to make the button unresponsive. The ACTIVE represents the active state of the button.
* **Underline**: Set this option to make the button text underlined.
* **Width**: The width of the button. It exists as a number of letters for textual buttons or pixels for image buttons.
* **Wraplength**: If the value is set to a positive number, the text lines will be wrapped to fit within this length.
* **cursor**: The cursor is used as the arrow, circle, dot, etc. on the canvas.

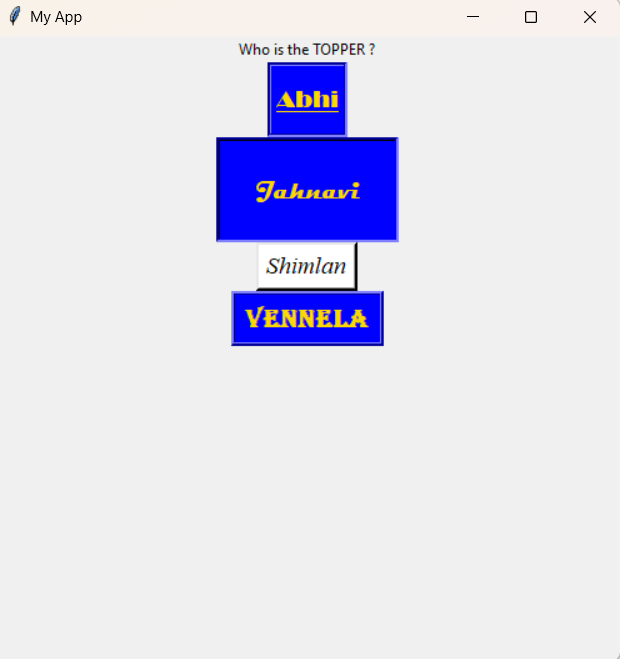
Python Tkinter supports quite a number of different mouse cursors available. The exact graphic may vary according to your operating system:

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

Example: Creating Buttons using the above options;



Output:



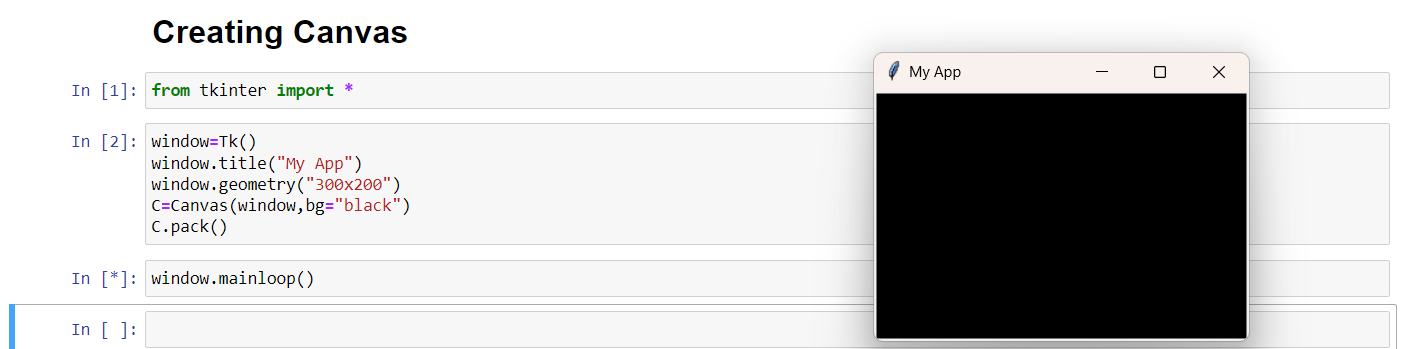
**Canvas:**

The canvas widget is used to add the structured graphics to the python application. It is used to draw the graph and plots to the python application.

Syntax:

**W = Canvas(parent,options)**

Example:



List of possible options used in Canvas widget:

* **bd**: The represents the border width. The default width is 2.
* **bg**: It represents the background color of the canvas.
* **confine**: It is set to make the canvas unscrollable outside the scroll region.
* **cursor**: The cursor is used as the arrow, circle, dot, etc. on the canvas.
* **height**: It represents the size of the canvas in the vertical direction.
* **highlightcolor**: It represents the highlight color when the widget is focused.
* **relief**: It represents the type of the border. The possible values are SUNKEN, RAISED, GROOVE, and RIDGE.
* **scrollregion**: It represents the coordinates specified as the tuple containing the area of the canvas.
* **width**: It represents the width of the canvas.
* **xscrollincrement**: If it is set to a positive value. The canvas is placed only to the multiple of this value.
* **xscrollcommand**: If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar.
* **yscrollincrement**: Works like xscrollincrement, but governs vertical movement.
* **yscrollcommand**: If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar.

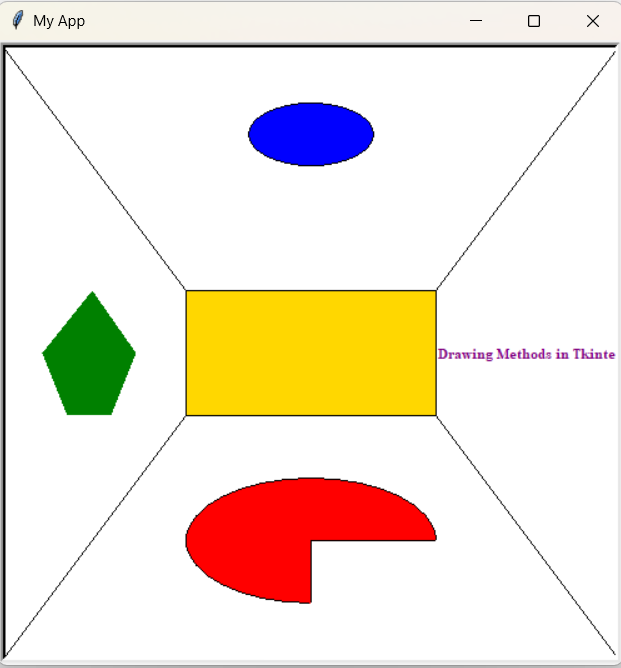
Drawing methods used in tkinter canvas:

* create\_line(x1, y1, x2, y2, options)
* create\_rectangle(x1, y1, x2, y2, options)
* create\_oval(x1, y1, x2, y2, options)
* create\_arc(x1, y1, x2, y2, options)
* create\_polygon(x1, y1, x2, y2, x3, y3, ..., options)
* create\_text(x, y, options)
* create\_image(x, y, options)

Example:



Output:



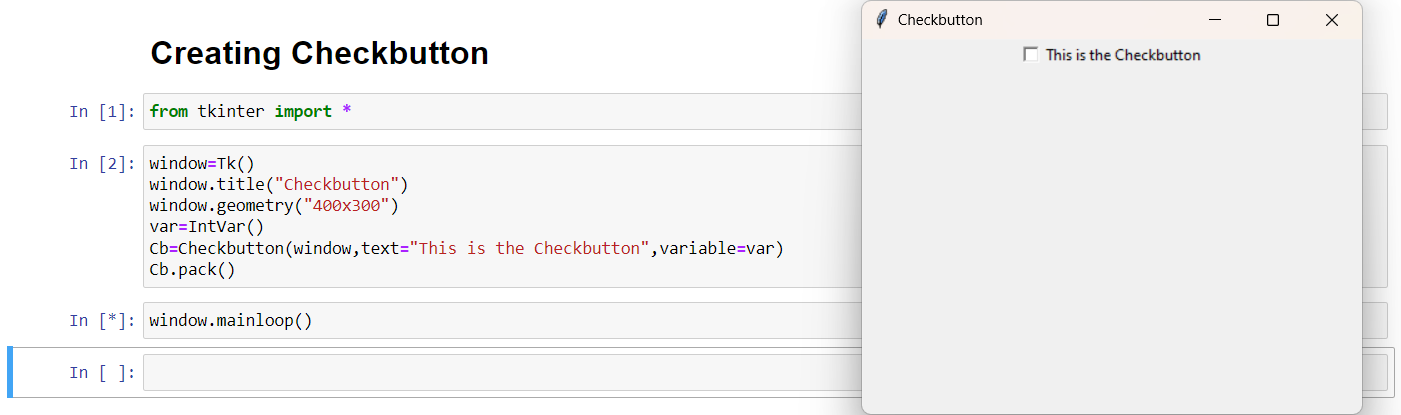
**Checkbutton:**

In tkinter, the Checkbutton widget is used to provide multiple options for selection. It allows the user to choose one or more options from a set of checkboxes.

Syntax:

**W = Checkbutton(parent,options)**

Example:



List of possible options used in Checkbutton widget:

* **activebackground**: It represents the background color when the checkbutton is under the cursor.
* **activeforeground**: It represents the foreground color of the checkbutton when the checkbutton is under the cursor.
* **bg:** The background color of the button.
* **bitmap**: It displays an image (monochrome) on the button.
* **bd**: The size of the border around the corner.
* **command**: It is associated with a function to be called when the state of the checkbutton is changed.
* **cursor**: The mouse pointer will be changed to the cursor name when it is over the checkbutton.
* **disableforeground**: It is the color which is used to represent the text of a disabled checkbutton.
* **font**: It represents the font of the checkbutton.
* **fg**: The foreground color (text color) of the checkbutton.
* **height**: It represents the height of the checkbutton (number of lines). The default height is 1.
* **highlightcolor**: The color of the focus highlight when the checkbutton is under focus.
* **image**: The image used to represent the checkbutton.
* **justify**: This specifies the justification of the text if the text contains multiple lines.
* **offvalue**: The associated control variable is set to 0 by default if the button is unchecked. We can change the state of an unchecked variable to some other one.
* **onvalue**: The associated control variable is set to 1 by default if the button is checked. We can change the state of the checked variable to some other one.
* **padx**: The horizontal padding of the checkbutton
* **pady**: The vertical padding of the checkbutton.
* **relief**: The type of the border of the checkbutton. By default, it is set to FLAT.
* **selectcolor**: The color of the checkbutton when it is set. By default, it is red.
* **selectimage**: The image is shown on the checkbutton when it is set.
* **state**: It represents the state of the checkbutton. By default, it is set to normal. We can change it to DISABLED to make the checkbutton unresponsive. The state of the checkbutton is ACTIVE when it is under focus.
* **underline**: It represents the index of the character in the text which is to be underlined. The indexing starts with zero in the text.
* **variable**: It represents the associated variable that tracks the state of the checkbutton.
* **width**: It represents the width of the checkbutton. It is represented in the number of characters that are represented in the form of texts.
* **wraplength**: If this option is set to an integer number, the text will be broken into the number of pieces.

Example:



Output:



**Entry:**

The Entry widget is used to provde the single line text-box to the user to accept a value from the user.

Syntax:

**W = Entry(parent,options)**

Example:



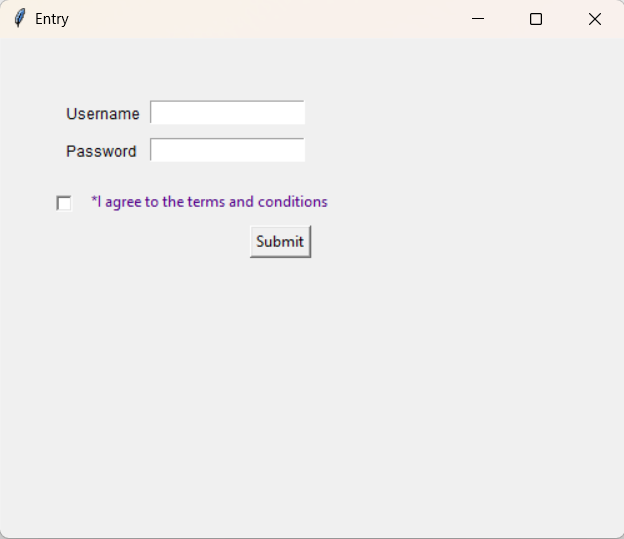
List of possible options used in Entry widget:

* **bg**: The background color of the widget.
* **bd**: The border width of the widget in pixels.
* **cursor**: The mouse pointer will be changed to the cursor type set to the arrow, dot, etc.
* **exportselection**: The text written inside the entry box will be automatically copied to the clipboard by default. We can set the exportselection to 0 to not copy this.
* **fg**: It represents the color of the text.
* **font**: It represents the font type of the text.
* **highlightbackground**: It represents the color to display in the traversal highlight region when the widget does not have the input focus.
* **highlightcolor**: It represents the color to use for the traversal highlight rectangle that is drawn around the widget when it has the input focus.
* **highlightthickness**: It represents a non-negative value indicating the width of the highlight rectangle to draw around the outside of the widget when it has the input focus.
* **insertbackground**: It represents the color to use as background in the area covered by the insertion cursor. This color will normally override either the normal background for the widget.
* **insertborderwidth**: It represents a non-negative value indicating the width of the 3-D border to draw around the insertion cursor. The value may have any of the forms acceptable to Tk\_GetPixels.
* **insertofftime**: It represents a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "off" in each blink cycle. If this option is zero, then the cursor doesn't blink: it is on all the time.
* **insertontime**: Specifies a non-negative integer value indicating the number of milliseconds the insertion cursor should remain "on" in each blink cycle.
* **insertwidth**: It represents the value indicating the total width of the insertion cursor. The value may have any of the forms acceptable to Tk\_GetPixels.
* **justify**: It specifies how the text is organized if the text contains multiple lines.
* **relief**: It specifies the type of the border. Its default value is FLAT.
* **selectbackground**: The background color of the selected text.
* **selectborderwidth**: The width of the border to display around the selected task.
* **selectforeground**: The font color of the selected task.
* **show**: It is used to show the entry text of some other type instead of the string. For example, the password is typed using stars (\*).
* **textvariable**: It is set to the instance of the StringVar to retrieve the text from the entry.
* **width**: The width of the displayed text or image.
* **xscrollcommand**: The entry widget can be linked to the horizontal scrollbar if we want the user to enter more text then the actual width of the widget.

Example:



Output:



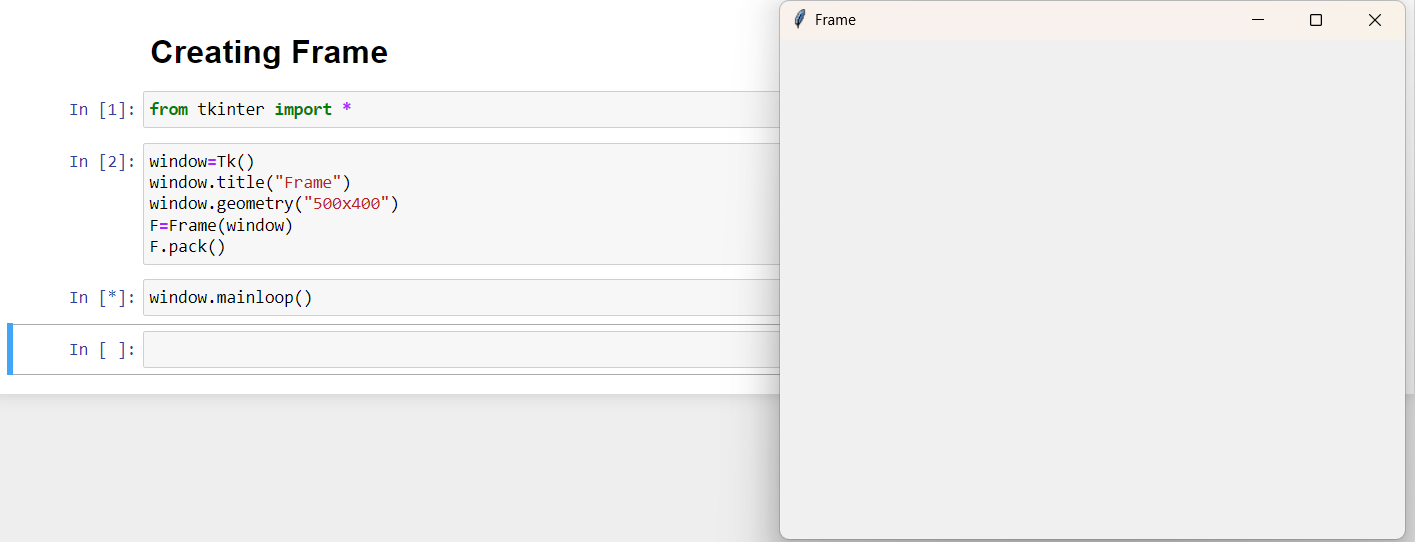
**Frame:**

Python Tkinter Frame widget is used to organize the group of widgets. It acts like a container which can be used to hold the other widgets.

Syntax:

**W = Frame(parent,options)**

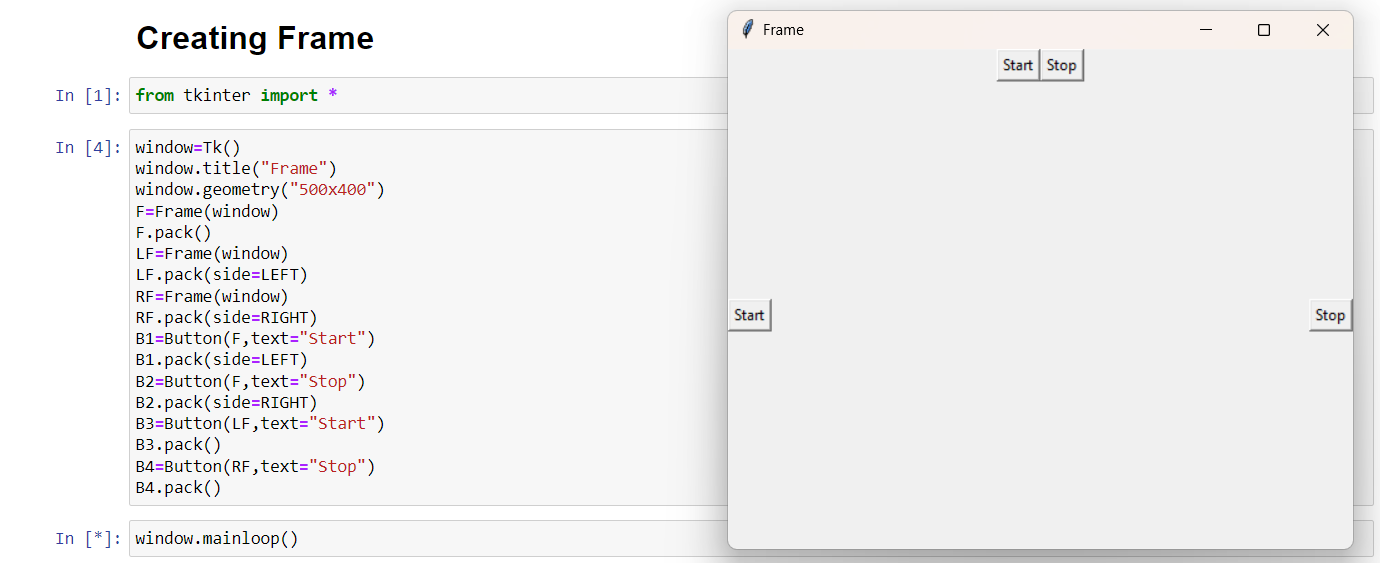
Example:



List of possible options used in Frame widget:

* **bd**: It represents the border width.
* **bg**: The background color of the widget.
* **cursor**: The mouse pointer is changed to the cursor type set to different values like an arrow, dot, etc.
* **height**: The height of the frame.
* **highlightbackground**: The color of the background color when it is under focus.
* **highlightcolor**: The text color when the widget is under focus.
* **highlightthickness**: It specifies the thickness around the border when the widget is under the focus.
* **relief**: It specifies the type of the border. The default value if FLAT.
* **width**: It represents the width of the widget.

Example:



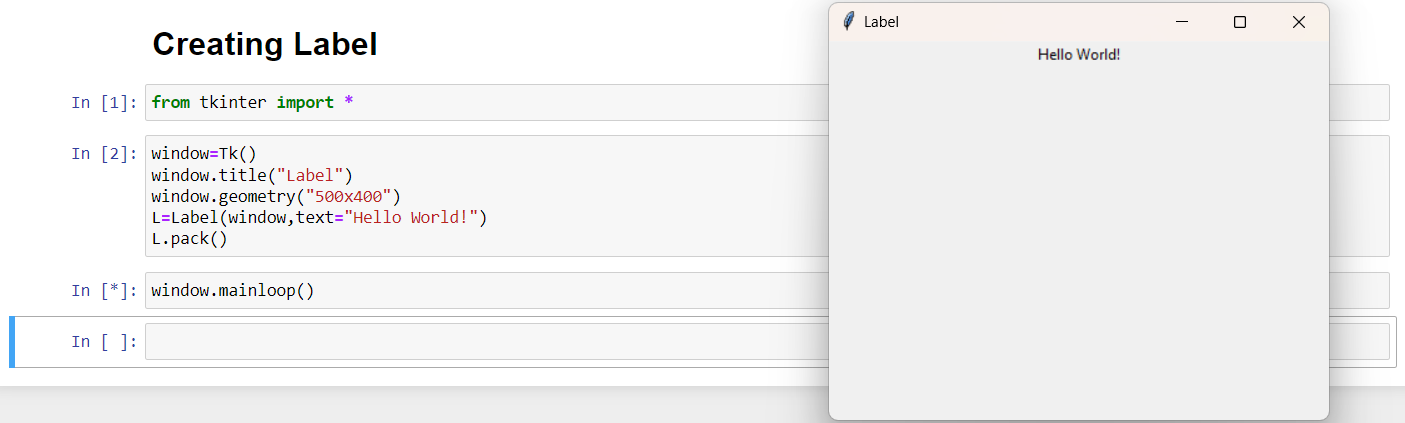
**Label:**

The Label is used to specify the container box where we can place the text or images. This widget is used to provide the message to the user about other widgets used in the python application.

Syntax:

**W = Label(parent,options)**

Example:



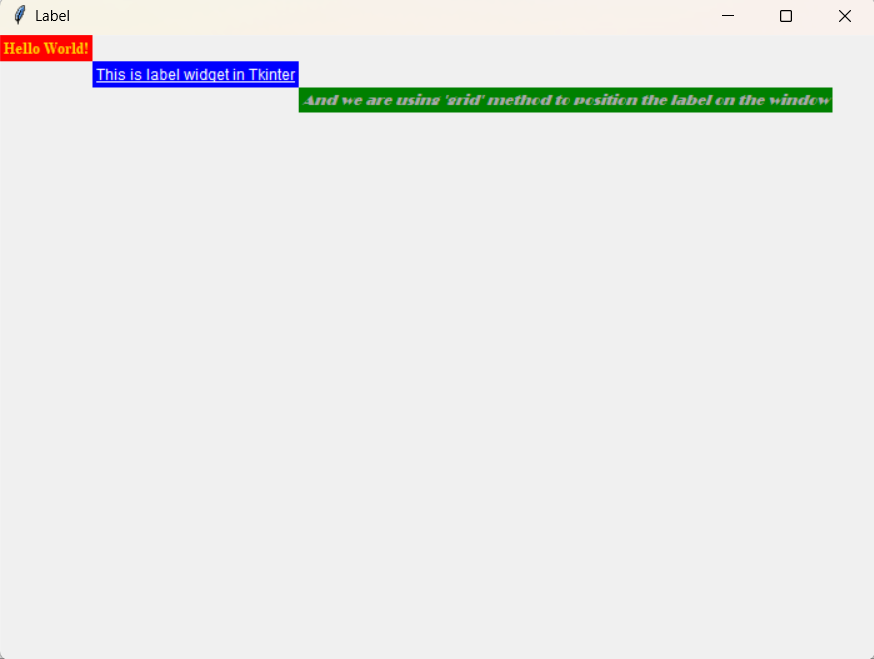
List of possible options used in Label widget:

* **anchor**: It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
* **bg**: The background color displayed behind the widget.
* **bitmap**: It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
* **bd**: It represents the width of the border. The default is 2 pixels.
* **cursor**: The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
* **font**: The font type of the text written inside the widget.
* **fg**: The foreground color of the text written inside the widget.
* **height**: The height of the widget.
* **image**: The image that is to be shown as the label.
* **justify**: It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for right justification, and CENTER for center justification.
* **padx**: The horizontal padding of the text. The default value is 1.
* **pady**: The vertical padding of the text. The default value is 1.
* **relief**: The type of the border. The default value is FLAT.
* **text**: This is set to the string variable which may contain one or more line of text.
* **textvariable**: The text written inside the widget is set to the control variable StringVar so that it can be accessed and changed accordingly.
* **underline**: We can display a line under the specified letter of the text. Set this option to the number of the letter under which the line will be displayed.
* **width**: The width of the widget. It is specified as the number of characters.
* **wraplength**: Instead of having only one line as the label text, we can break it to the number of lines where each line has the number of characters specified to this option.

Example:



Output:



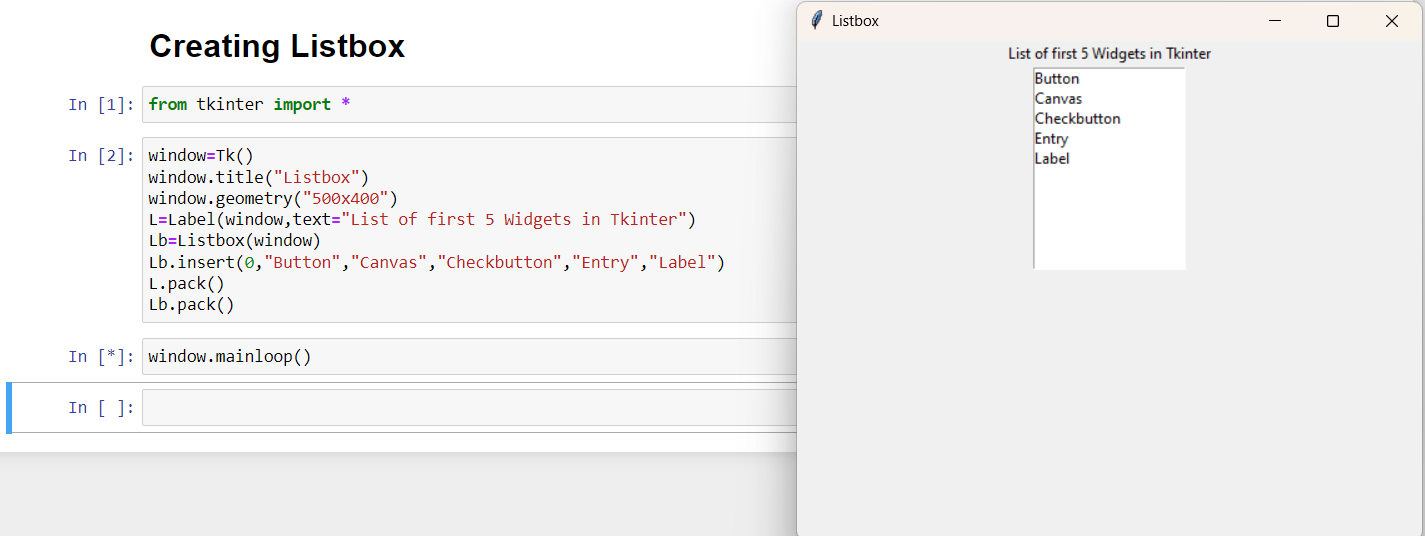
**Listbox:**

The Listbox widget is used to display the list items to the user. We can place only text items in the Listbox and all text items contain the same font and color.

Syntax:

**W = Listbox(parent,options)**

Example:



List of possible options used in Listbox widget:

* **bg**: The background color of the widget.
* **bd**: It represents the size of the border. Default value is 2 pixel.
* **cursor**: The mouse pointer will look like the cursor type like dot, arrow, etc.
* **font**: The font type of the Listbox items.
* **fg**: The color of the text.
* **height**: It represents the count of the lines shown in the Listbox. The default value is 10.
* **highlightcolor**: The color of the Listbox items when the widget is under focus.
* **highlightthickness**: The thickness of the highlight.
* **relief**: The type of the border. The default is SUNKEN.
* **selectbackground**: The background color that is used to display the selected text.
* **selectmode**: It is used to determine the number of items that can be selected from the list. It can set to BROWSE, SINGLE, MULTIPLE, EXTENDED.
* **width**: It represents the width of the widget in characters.
* **xscrollcommand**: It is used to let the user scroll the Listbox horizontally.
* **yscrollcommand**: It is used to let the user scroll the Listbox vertically.

Example:



Output:



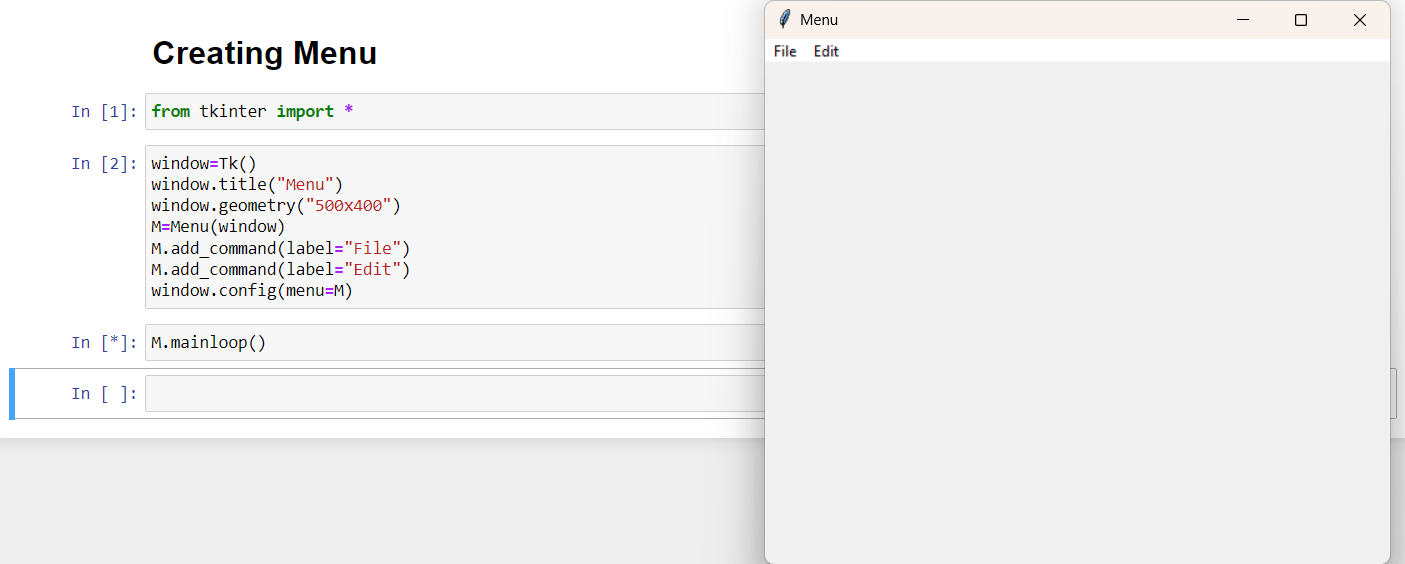
**Menu:**

The Menu widget is used to create various types of menus (top level, pull down, and pop up) in the python application. The top-level menus are the one which is displayed just under the title bar of the parent window. We need to create a new instance of the Menu widget and add various commands to it by using the add() method.

Syntax:

**W = Menu(parent,options)**

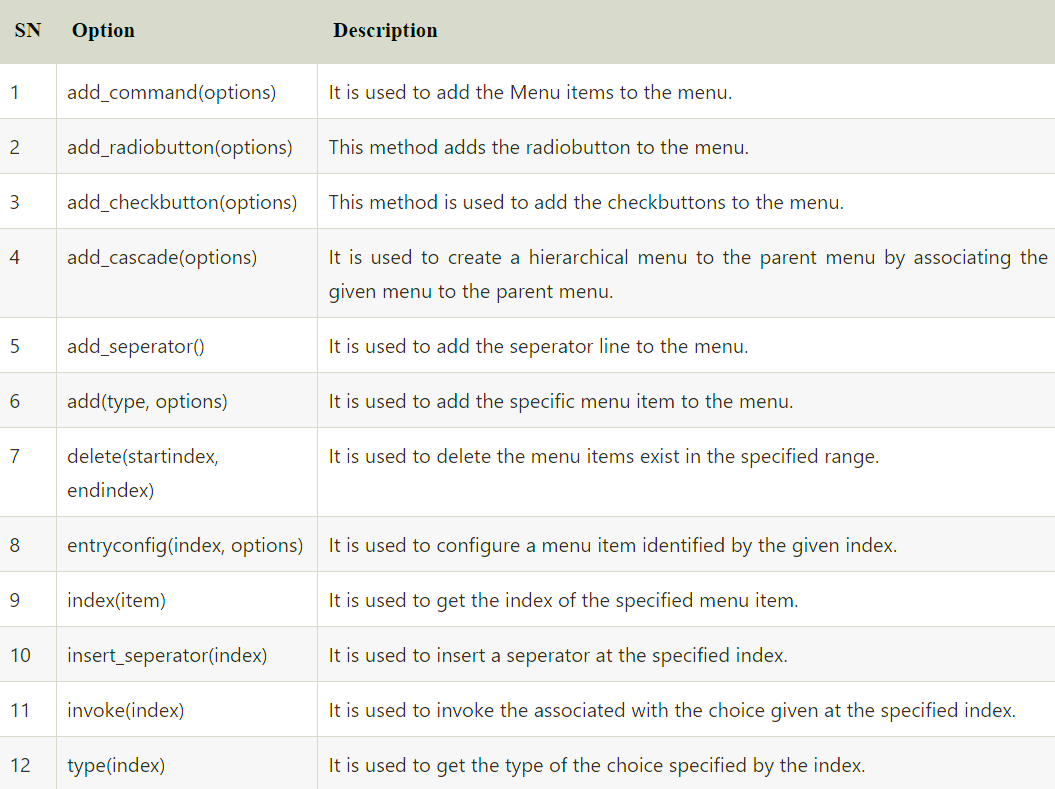
Example:



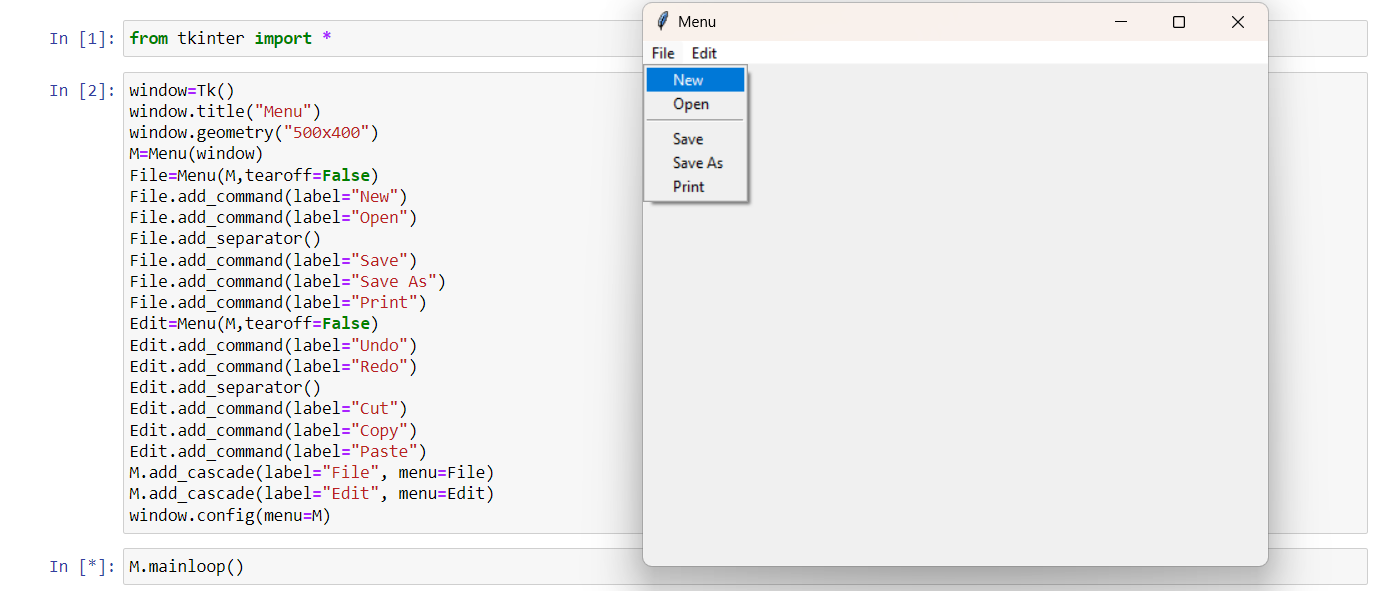
List of possible options used in Menu widget:

* **activebackground**: The background color of the widget when the widget is under the focus.
* **activeborderwidth**: The width of the border of the widget when it is under the mouse. The default is 1 pixel.
* **activeforeground**: The font color of the widget when the widget has the focus.
* **bg**: The background color of the widget.
* **bd**: The border width of the widget.
* **cursor**: The mouse pointer is changed to the cursor type when it hovers the widget. The cursor type can be set to arrow or dot.
* **disabledforeground**: The font color of the widget when it is disabled.
* **font**: The font type of the text of the widget.
* **fg**: The foreground color of the widget.
* **postcommand**: The postcommand can be set to any of the function which is called when the mourse hovers the menu.
* **relief**: The type of the border of the widget. The default type is RAISED.
* **image**: It is used to display an image on the menu.
* **selectcolor**: The color used to display the checkbutton or radiobutton when they are selected.
* **tearoff**: The tearoff option is used to enable or disable the "tear off" feature of a menu.
* **title**: Set this option to the title of the window if you want to change the title of the window.

The Menu widget contains the following methods:



Example:



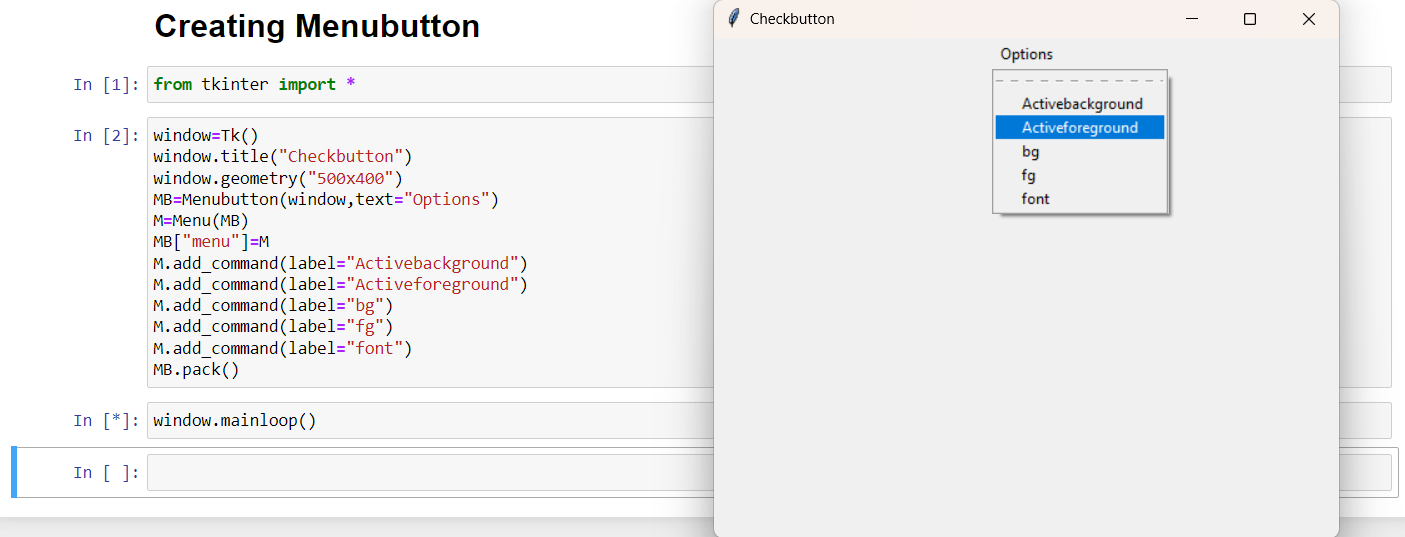
**Menubutton:**

In tkinter, the Menubutton is used to create a button that can display a dropdown menu when clicked. The dropdown menu typically contains a list of options or commands for the user to choose from.

Syntax:

**W = Menubutton(parent,options)**

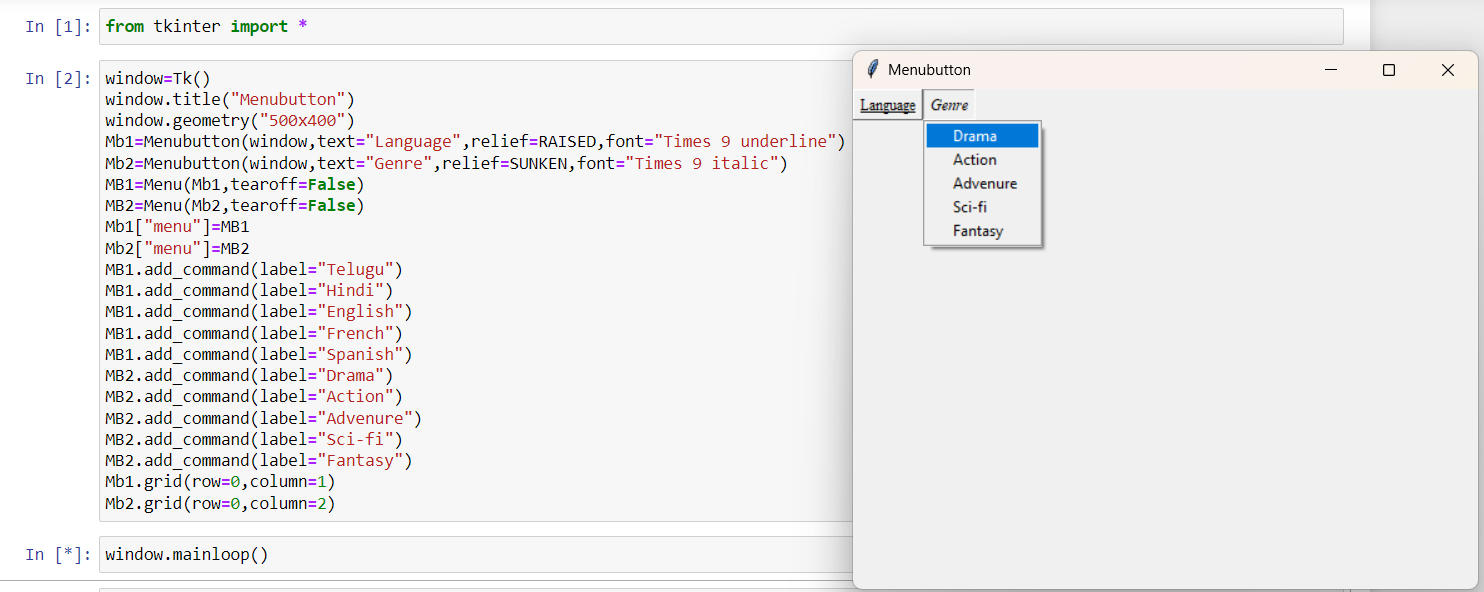
Example:



List of possible options used in Menubutton widget:

* **activebackground**: The background color of the widget when the widget is under focus.
* **activeforeground**: The font color of the widget text when the widget is under focus.
* **anchor**: It specifies the exact position of the widget content when the widget is assigned more space than needed.
* **bg**: It specifies the background color of the widget.
* **bitmap**: It is set to the graphical content which is to be displayed to the widget.
* **bd**: It represents the size of the border. The default value is 2 pixels.
* **cursor**: The mouse pointer will be changed to the cursor type specified when the widget is under the focus. The possible value of the cursor type is arrow, or dot etc.
* **direction**: It direction can be specified so that menu can be displayed to the specified direction of the button. Use LEFT, RIGHT, or ABOVE to place the widget accordingly.
* **disabledforeground**: The text color of the widget when the widget is disabled.
* **fg**: The normal foreground color of the widget.
* **height**: The vertical dimension of the Menubutton. It is specified as the number of lines.
* **highlightcolor**: The highlight color shown to the widget under focus.
* **image**: The image displayed on the widget.
* **justify**: This specified the exact position of the text under the widget when the text is unable to fill the width of the widget. We can use the LEFT for the left justification, RIGHT for the right justification, CENTER for the centre justification.
* **menu**: It represents the menu specified with the Menubutton.
* **padx**: The horizontal padding of the widget.
* **pady**: The vertical padding of the widget.
* **relief**: This option specifies the type of the border. The default value is RAISED.
* **state**: The normal state of the Mousebutton is enabled. We can set it to DISABLED to make it unresponsive.
* **text**: The text shown with the widget.
* **textvariable**: We can set the control variable of string type to the text variable so that we can control the text of the widget at runtime.
* **underline**: The text of the widget is not underlined by default but we can set this option to make the text of the widget underlined.
* **width**: It represents the width of the widget in characters. The default value is 20.
* **wraplength**: We can break the text of the widget in the number of lines so that the text contains the number of lines not greater than the specified value.

Example:



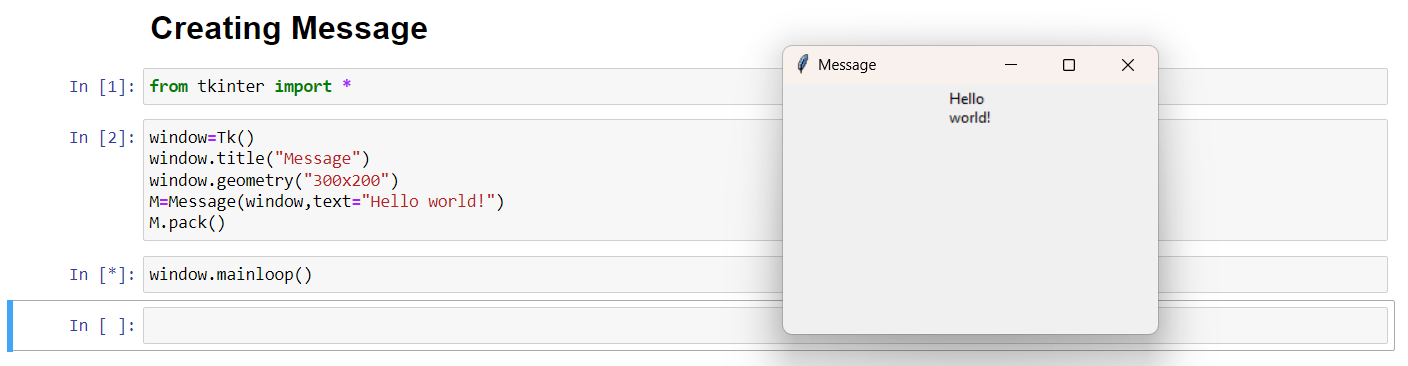
**Message:**

The Message widget is used to show the message to the user regarding the behaviour of the python application. The message widget shows the text messages to the user which can not be edited.

Syntax:

**W = Message(parent,options)**

Example:



List of possible options used in Message widget:

* **anchor**: It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
* **bg**: The background color displayed behind the widget.
* **bitmap**: It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
* **bd**: It represents the width of the border. The default is 2 pixels.
* **cursor**: The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
* **font**: The font type of the text written inside the widget.
* **fg**: The foreground color of the text written inside the widget.
* **height**: The height of the widget.
* **image**: The image that is to be shown as the label.
* **justify**: It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for right justification, and CENTER for center justification.
* **padx**: The horizontal padding of the text. The default value is 1.
* **pady**: The vertical padding of the text. The default value is 1.
* **relief**: The type of the border. The default value is FLAT.
* **text**: This is set to the string variable which may contain one or more line of text.
* **textvariable**: The text written inside the widget is set to the control variable StringVar so that it can be accessed and changed accordingly.
* **underline**: We can display a line under the specified letter of the text. Set this option to the number of the letter under which the line will be displayed.
* **width**: The width of the widget. It is specified as the number of characters.
* **wraplength**: Instead of having only one line as the label text, we can break it to the number of lines where each line has the number of characters specified to this option.

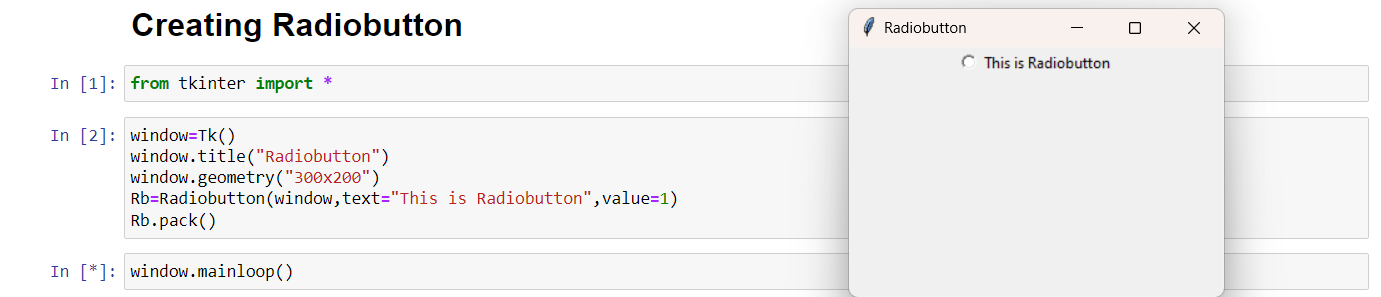
**Radiobutton:**

The Radiobutton widget is used to implement one-of-many selection in the python application. It shows multiple choices to the user out of which, the user can select only one out of them. We can associate different methods with each of the radiobutton.

Syntax:

**W = Radiobutton(parent,options)**

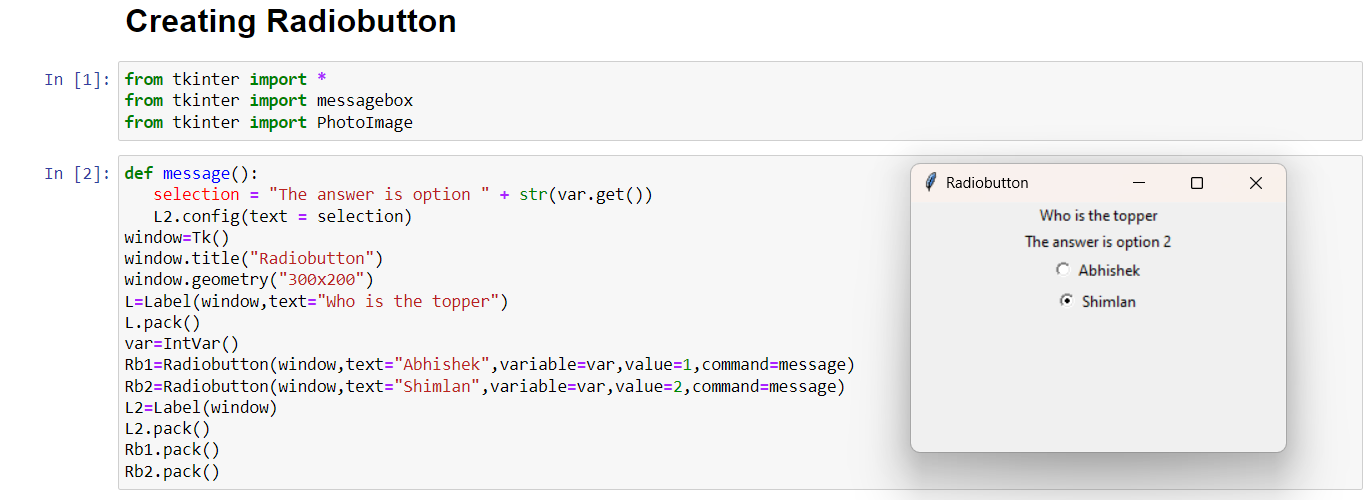
Example:



List of possible options used in Radiobutton widget:

* **anchor**: It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
* **bg**: The background color displayed behind the widget.
* **bitmap**: It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
* **bd**: It represents the width of the border. The default is 2 pixels.
* **cursor**: The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
* **font**: The font type of the text written inside the widget.
* **fg**: The foreground color of the text written inside the widget.
* **height**: The height of the widget.
* **image**: The image that is to be shown as the label.
* **justify**: It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for right justification, and CENTER for center justification.
* **padx**: The horizontal padding of the text. The default value is 1.
* **pady**: The vertical padding of the text. The default value is 1.
* **relief**: The type of the border. The default value is FLAT.
* **text**: This is set to the string variable which may contain one or more line of text.
* **textvariable**: The text written inside the widget is set to the control variable StringVar so that it can be accessed and changed accordingly.
* **underline**: We can display a line under the specified letter of the text. Set this option to the number of the letter under which the line will be displayed.
* **width**: The width of the widget. It is specified as the number of characters.
* **wraplength**: Instead of having only one line as the label text, we can break it to the number of lines where each line has the number of characters specified to this option.

Example:



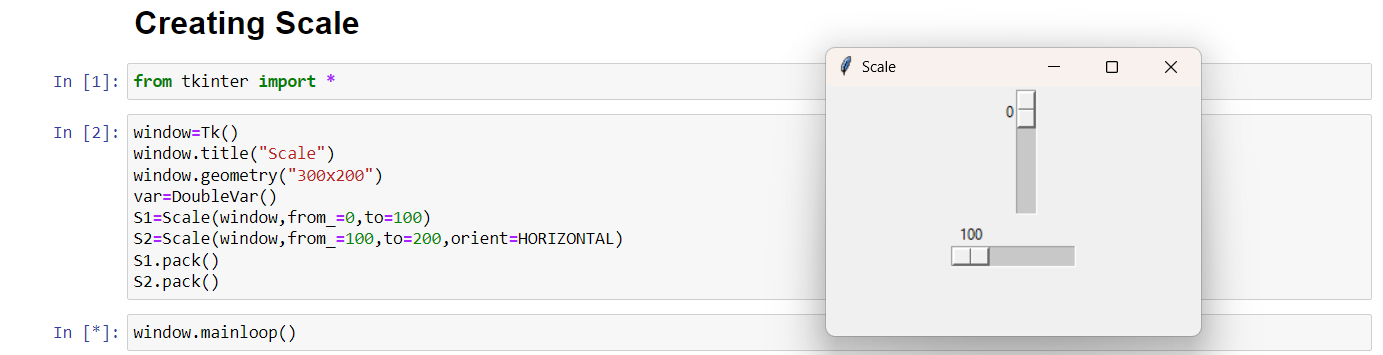
**Scale:**

The Scale widget is used to implement the graphical slider to the python application so that the user can slide through the range of values shown on the slider and select the one among them.

Syntax:

**W = Scale(parent,options)**

Example:



List of possible options used in Scale widget:

* **activebackground**: The background color of the widget when it has the focus.
* **bg**: The background color of the widget.
* **bd**: The border size of the widget. The default is 2 pixel.
* **command**: It is set to the procedure which is called each time when we move the slider. If the slider is moved rapidly, the callback is done when it settles.
* **cursor**: The mouse pointer is changed to the cursor type assigned to this option. It can be an arrow, dot, etc.
* **digits**: If the control variable used to control the scale data is of string type, this option is used to specify the number of digits when the numeric scale is converted to a string.
* **font**: The font type of the widget text.
* **fg**: The foreground color of the text.
* **from\_**: It is used to represent one end of the widget range.
* **highlightbackground**: The highlight color when the widget doesn't have the focus.
* **highlighcolor**: The highlight color when the widget has the focus.
* **label**: This can be set to some text which can be shown as a label with the scale. It is shown in the top left corner if the scale is horizontal or the top right corner if the scale is vertical.
* **length**: It represents the length of the widget. It represents the X dimension if the scale is horizontal or y dimension if the scale is vertical.
* **orient**: It can be set to horizontal or vertical depending upon the type of the scale.
* **relief**: It represents the type of the border. The default is FLAT.
* **repeatdelay**: This option tells the duration up to which the button is to be pressed before the slider starts moving in that direction repeatedly. The default is 300 ms.
* **resolution**: It is set to the smallest change which is to be made to the scale value.
* **showvalue**: The value of the scale is shown in the text form by default. We can set this option to 0 to suppress the label.
* **sliderlength**: It represents the length of the slider window along the length of the scale. The default is 30 pixels. However, we can change it to the appropriate value.
* **state**: The scale widget is active by default. We can set this to DISABLED to make it unresponsive.
* **takefocus**: The focus cycles through the scale widgets by default. We can set this option to 0 if we don't want this to happen.
* **tickinterval**: The scale values are displayed on the multiple of the specified tick interval. The default value of the tickinterval is 0.
* **to**: It represents a float or integer value that specifies the other end of the range represented by the scale.
* **troughcolor**: It represents the color of the through.
* **variable**: It represents the control variable for the scale.
* **width**: It represents the width of the through part of the widget.

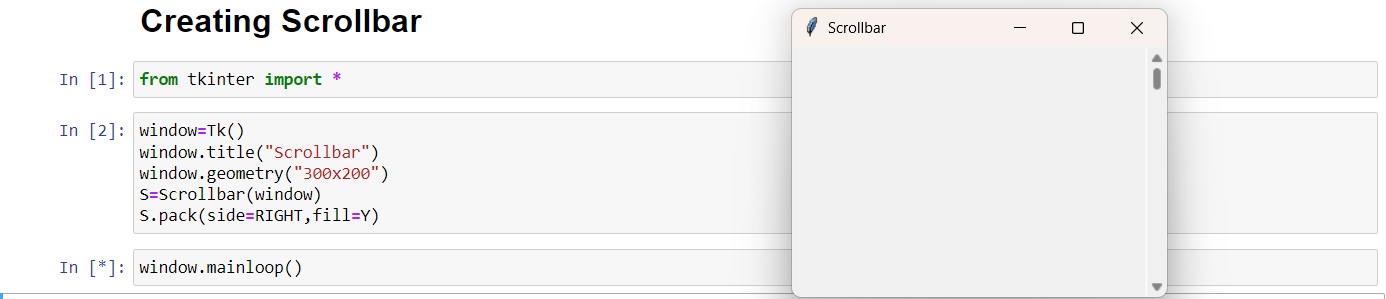
**Scrollbar:**

The scrollbar widget is used to scroll down the content of the other widgets like listbox, text, and canvas. However, we can also create the horizontal scrollbars to the Entry widget.

Syntax:

**W = Scrollbar(parent,options)**

Example:



List of possible options used in Scrollbar widget:

* **activebackground**: The background color of the widget when it has the focus.
* **bg**: The background color of the widget.
* **bd**: The border width of the widget.
* **command**: It can be set to the procedure associated with the list which can be called each time when the scrollbar is moved.
* **cursor**: The mouse pointer is changed to the cursor type set to this option which can be an arrow, dot, etc.
* **elementborderwidth**: It represents the border width around the arrow heads and slider. The default value is -1.
* **Highlightbackground**: The focus highlighcolor when the widget doesn't have the focus.
* **highlighcolor**: The focus highlighcolor when the widget has the focus.
* **highlightthickness**: It represents the thickness of the focus highlight.
* **jump**: It is used to control the behavior of the scroll jump. If it set to 1, then the callback is called when the user releases the mouse button.
* **orient**: It can be set to HORIZONTAL or VERTICAL depending upon the orientation of the scrollbar.
* **repeatdelay**: This option tells the duration up to which the button is to be pressed before the slider starts moving in that direction repeatedly. The default is 300 ms.
* **repeatinterval**: The default value of the repeat interval is 100.
* **takefocus**: We can tab the focus through this widget by default. We can set this option to 0 if we don't want this behavior.
* **troughcolor**: It represents the color of the trough.
* **width**: It represents the width of the scrollbar.

Example:



**Text:**

In Tkinter, the standard Python library for creating graphical user interfaces (GUI), you can use the **Text** widget to display and edit multiline text. The **Text** widget provides a lot of functionality for manipulating and formatting text, including inserting and deleting text, selecting and modifying text ranges, and applying various formatting options.

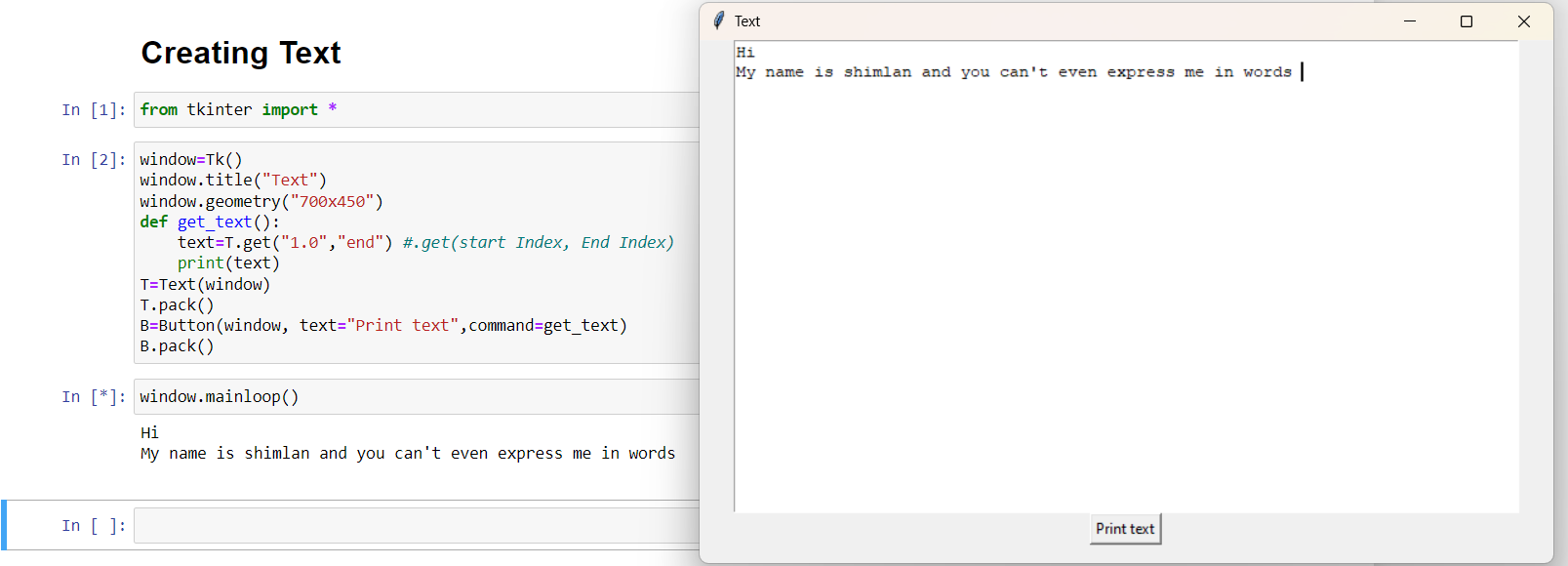
Syntax:

**W = Text(parent,options)**

List of possible options used in Text widget:

* **bg**: The background color of the widget.
* **bd**: It represents the border width of the widget.
* **cursor**: The mouse pointer is changed to the specified cursor type, i.e. arrow, dot, etc.
* **exportselection**: The selected text is exported to the selection in the window manager. We can set this to 0 if we don't want the text to be exported.
* **font**: The font type of the text.
* **fg**: The text color of the widget.
* **height**: The vertical dimension of the widget in lines.
* **highlightbackground**: The highlightcolor when the widget doesn't has the focus.
* **highlightthickness**: The thickness of the focus highlight. The default value is 1.
* **highlighcolor**: The color of the focus highlight when the widget has the focus.
* **insertbackground**: It represents the color of the insertion cursor.
* **insertborderwidth**: It represents the width of the border around the cursor. The default is 0.
* **insertofftime**: The time amount in Milliseconds during which the insertion cursor is off in the blink cycle.
* **insertontime**: The time amount in Milliseconds during which the insertion cursor is on in the blink cycle.
* **insertwidth**: It represents the width of the insertion cursor.
* **padx**: The horizontal padding of the widget.
* **pady**: The vertical padding of the widget.
* **relief**: The type of the border. The default is SUNKEN.
* **selectbackground**: The background color of the selected text.
* **selectborderwidth**: The width of the border around the selected text.
* **spacing1**: It specifies the amount of vertical space given above each line of the text. The default is 0.
* **spacing2**: This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. The default is 0.
* **spacing3**: It specifies the amount of vertical space to insert below each line of the text.
* **state**: It the state is set to DISABLED, the widget becomes unresponsive to the mouse and keyboard unresponsive.
* **tabs**: This option controls how the tab character is used to position the text.
* **width**: It represents the width of the widget in characters.
* **wrap**: This option is used to wrap the wider lines into multiple lines. Set this option to the WORD to wrap the lines after the word that fit into the available space. The default value is CHAR which breaks the line which gets too wider at any character.
* **xscrollcommand**: To make the Text widget horizontally scrollable, we can set this option to the set() method of Scrollbar widget.
* **yscrollcommand**: To make the Text widget vertically scrollable, we can set this option to the set() method of Scrollbar widget.

Example:



Methods:

* **delete(startindex, endindex)**: This method is used to delete the characters of the specified range.
* **get(startindex, endindex)**: It returns the characters present in the specified range.
* **index(index)**: It is used to get the absolute index of the specified index.
* **insert(index, string)**: It is used to insert the specified string at the given index.
* **see(index)**: It returns a boolean value true or false depending upon whether the text at the specified index is visible or not.

**Toplevel:**

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

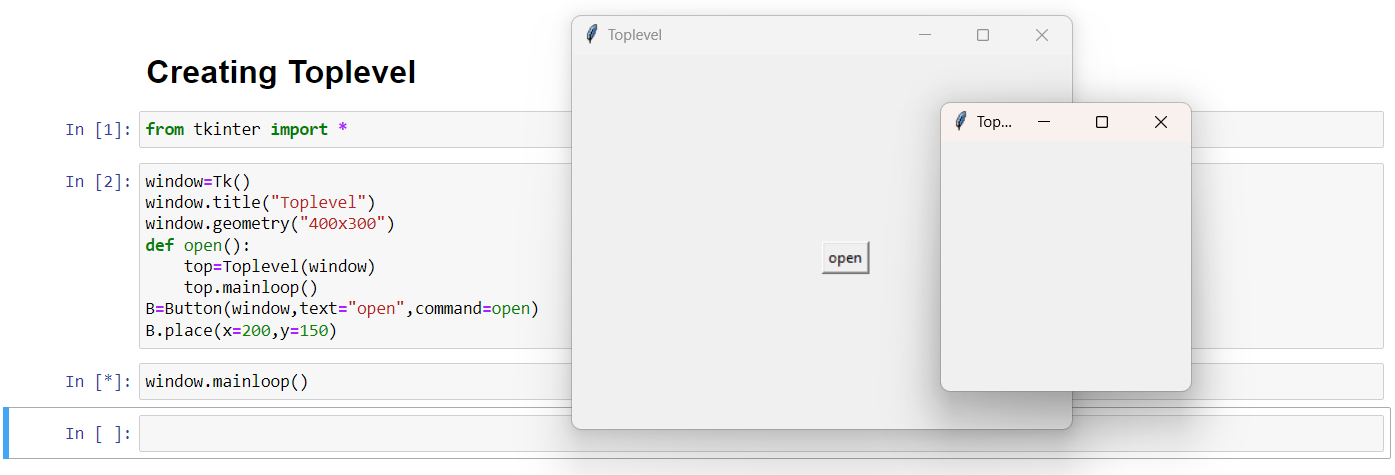
Syntax:

**W = Toplevel(options)**

List of possible options used in Toplevel widget:

* **bg**: It represents the background color of the window.
* **bd**: It represents the border size of the window.
* **cursor**: The mouse pointer is changed to the cursor type set to the arrow, dot, etc. when the mouse is in the window.
* **class\_**: The text selected in the text widget is exported to be selected to the window manager. We can set this to 0 to make this behavior false.
* **font**: The font type of the text inserted into the widget.
* **fg**: The foreground color of the widget.
* **height**: It represents the height of the window.
* **relief**: It represents the type of the window.
* **width**: It represents the width of the window.

Example:



**Spinbox:**

The Spinbox widget is an alternative to the Entry widget. It provides the range of values to the user, out of which, the user can select the one.

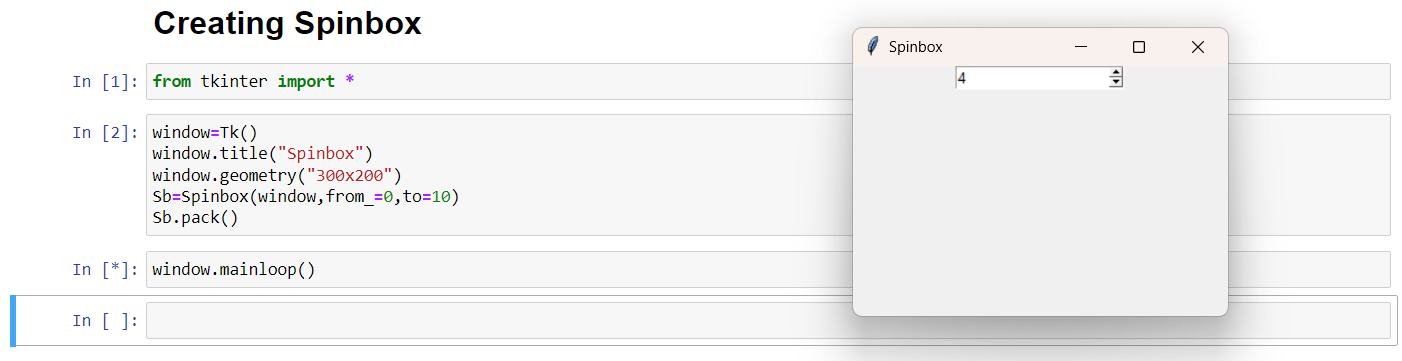
Syntax:

**W = Spinbox(parent,options)**

List of possible options used in Spinbox widget:

* **activebackground**: The background color of the widget when it has the focus.
* **bg**: The background color of the widget.
* **bd**: The border width of the widget.
* **command**: The associated callback with the widget which is called each time the state of the widget is called.
* **cursor**: The mouse pointer is changed to the cursor type assigned to this option.
* **disabledbackground**: The background color of the widget when it is disabled.
* **disabledforeground**: The foreground color of the widget when it is disabled.
* **fg**: The normal foreground color of the widget.
* **font**: The font type of the widget content.
* **format**: This option is used for the format string. It has no default value.
* **from**\_: It is used to show the starting range of the widget.
* **justify**: It is used to specify the justification of the multi-line widget content. The default is LEFT.
* **relief**: It is used to specify the type of the border. The default is SUNKEN.
* **repeatdelay**: This option is used to control the button auto repeat. The value is given in milliseconds.
* **repeatinterval**: It is similar to repeatdelay. The value is given in milliseconds.
* **state**: It represents the state of the widget. The default is NORMAL. The possible values are NORMAL, DISABLED, or "readonly".
* **textvariable**: It is like a control variable which is used to control the behaviour of the widget text.
* **to**: It specify the maximum limit of the widget value. The other is specified by the from\_ option.
* **validate**: This option controls how the widget value is validated.
* **validatecommand**: It is associated to the function callback which is used for the validation of the widget content.
* **values**: It represents the tuple containing the values for this widget.
* **vcmd**: It is same as validation command.
* **width**: It represents the width of the widget.
* **wrap**: This option wraps up the up and down button the Spinbox.
* **xscrollcommand**: This options is set to the set() method of scrollbar to make this widget horizontally scrollable.

Example:



**PanedWindow:**

The PanedWindow widget acts like a Container widget which contains one or more child widgets (panes) arranged horizontally or vertically. The child panes can be resized by the user, by moving the separator lines known as sashes by using the mouse. Each pane contains only one widget. The PanedWindow is used to implement the different layouts in the python applications.

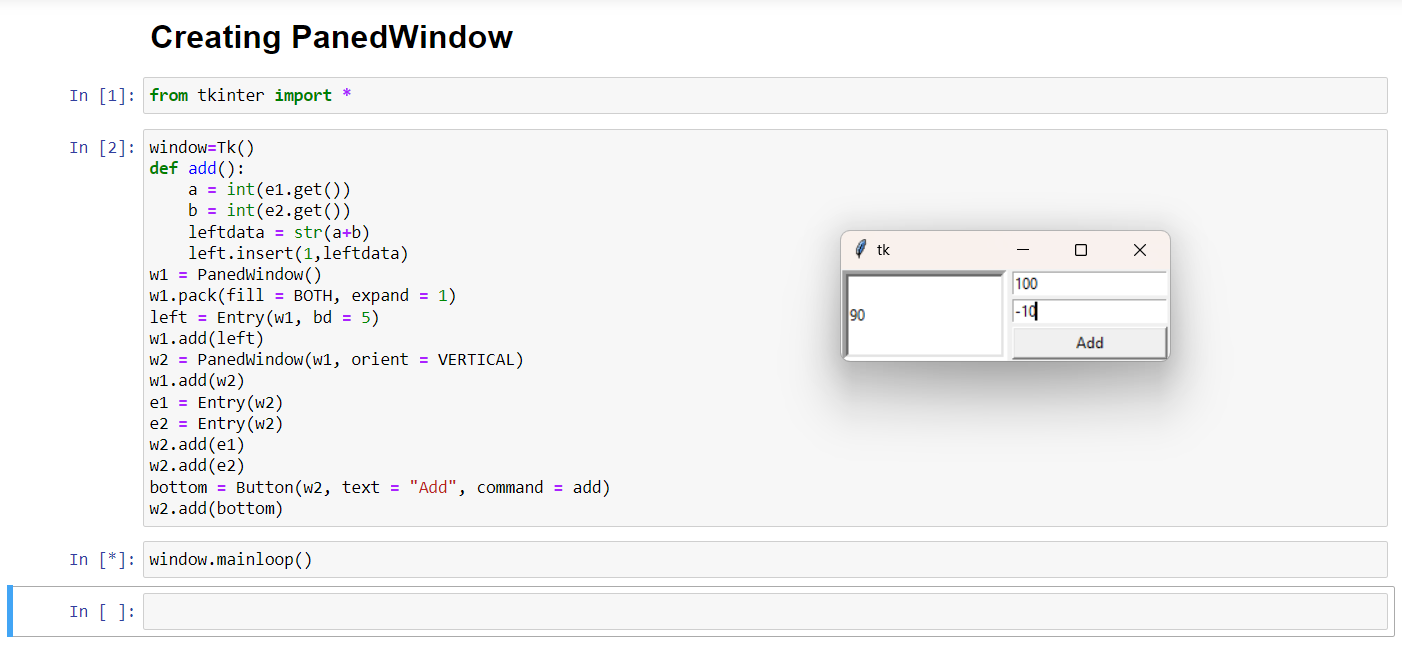
Syntax:

**W = PanedWindow(parent,options)**

List of possible options used in PanedWindow widget:

* **bg**: It represents the background color of the widget when it doesn't have the focus.
* **bd**: It represents the 3D border size of the widget. The default option specifies that the trough contains no border whereas the arrowheads and slider contain the 2-pixel border size.
* **borderwidth**: It represents the border width of the widget. The default is 2 pixel.
* **cursor**: The mouse pointer is changed to the specified cursor type when it is over the window.
* **handlepad**: This option represents the distance between the handle and the end of the sash. For the horizontal orientation, it is the distance between the top of the sash and the handle. The default is 8 pixels.
* **handlesize**: It represents the size of the handle. The default size is 8 pixels. However, the handle will always be a square.
* **height**: It represents the height of the widget. If we do not specify the height, it will be calculated by the height of the child window.
* **orient**: The orient will be set to HORIZONTAL if we want to place the child windows side by side. It can be set to VERTICAL if we want to place the child windows from top to bottom.
* **relief**: It represents the type of the border. The default is FLAT.
* **sashpad**: It represents the padding to be done around each sash. The default is 0.
* **sashrelief**: It represents the type of the border around each of the sash. The default is FLAT.
* **sashwidth**: It represents the width of the sash. The default is 2 pixels.
* **showhandle**: It is set to True to display the handles. The default value is false.
* **Width**: It represents the width of the widget. If we don't specify the width of the widget, it will be calculated by the size of the child widgets.

Example:



**LabelFrame:**

The LabelFrame widget is used to draw a border around its child widgets. We can also display the title for the LabelFrame widget. It acts like a container which can be used to group the number of interrelated widgets such as Radiobuttons.

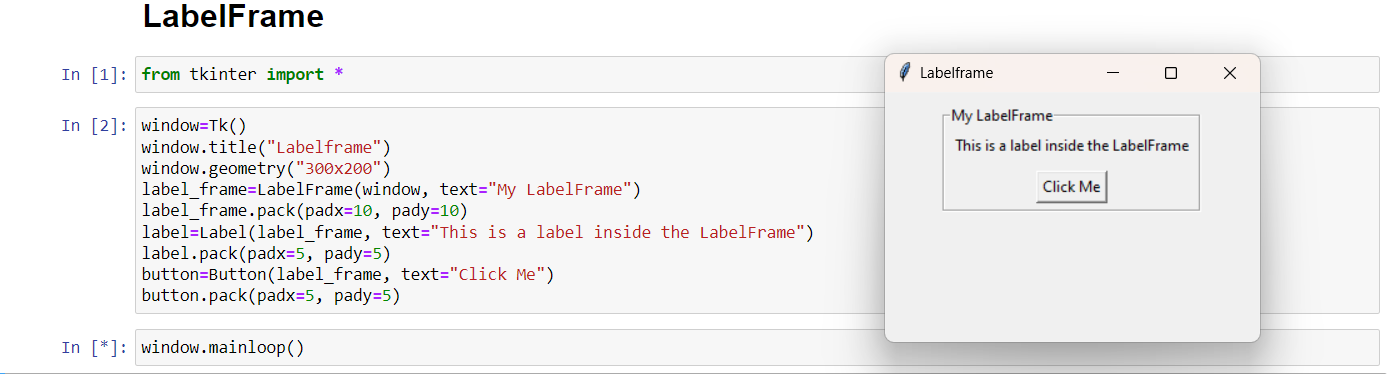
Syntax:

**W = LabelFrame(parent,options)**

List of possible options used in LabelFrame widget:

* **anchor**: It specifies the exact position of the text within the size provided to the widget. The default value is CENTER, which is used to center the text within the specified space.
* **bg**: The background color displayed behind the widget.
* **bitmap**: It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
* **bd**: It represents the width of the border. The default is 2 pixels.
* **cursor**: The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
* **font**: The font type of the text written inside the widget.
* **fg**: The foreground color of the text written inside the widget.
* **height**: The height of the widget.
* **image**: The image that is to be shown as the label.
* **justify**: It is used to represent the orientation of the text if the text contains multiple lines. It can be set to LEFT for left justification, RIGHT for right justification, and CENTER for center justification.
* **padx**: The horizontal padding of the text. The default value is 1.
* **pady**: The vertical padding of the text. The default value is 1.
* **relief**: The type of the border. The default value is FLAT.
* **text**: This is set to the string variable which may contain one or more line of text.
* **textvariable**: The text written inside the widget is set to the control variable StringVar so that it can be accessed and changed accordingly.
* **underline**: We can display a line under the specified letter of the text. Set this option to the number of the letter under which the line will be displayed.
* **width**: The width of the widget. It is specified as the number of characters.
* **wraplength**: Instead of having only one line as the label text, we can break it to the number of lines where each line has the number of characters specified to this option.

Example:



Messagebox:

The messagebox module is used to display the message boxes in the python applications. There are the various functions which are used to display the relevant messages depending upon the application requirements.

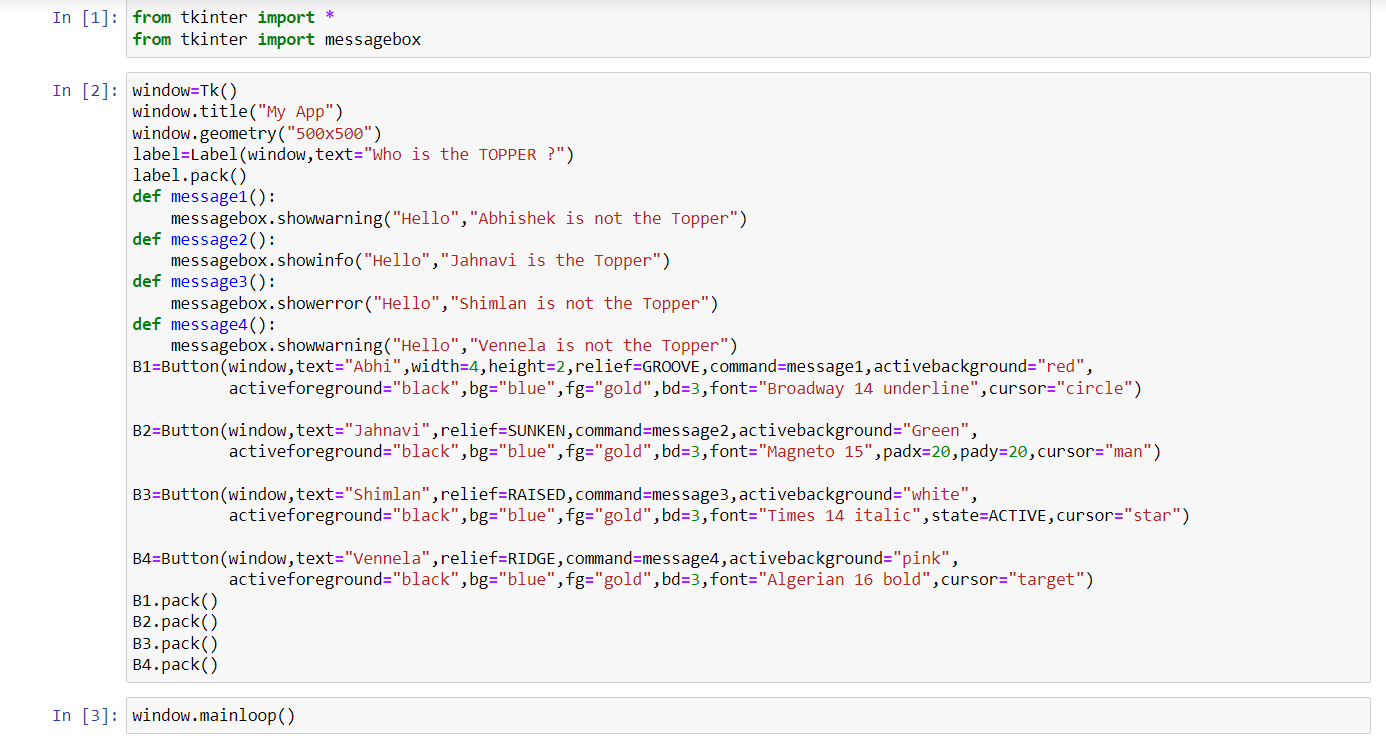
Syntax:

**W = Messagebox.function\_name(title,message,options)**

List of different types of messagebox:

* showinfo( ): The showinfo() messagebox is used where we need to show some relevant information to the user.
* Showwarning( ): This method is used to display the warning to the user.
* showerror( ): This method is used to display the error message to the user. Consider the following example.
* askquestion( ): This method is used to ask some question to the user which can be answered in yes or no.
* askokcancel( ): This method is used to confirm the user's action regarding some application activity.
* askyesno( ): This method is used to ask the user about some action to which, the user can answer in yes or no.
* askretrycancel( ): This method is used to ask the user about doing a particular task again or not.

Example:



Output:

