Canvas

The Canvas is a rectangular area intended for drawing pictures or other complex layouts. You can place graphics, text, widgets2. or frames on a Canvas.

Syntax

Here is the simple syntax to create this widget:

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

Parameters

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

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.

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

The Canvas widget can support the following standard items:

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

```
coord = 10, 50, 240, 210

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

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

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

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

line . Creates a line item.

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

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.

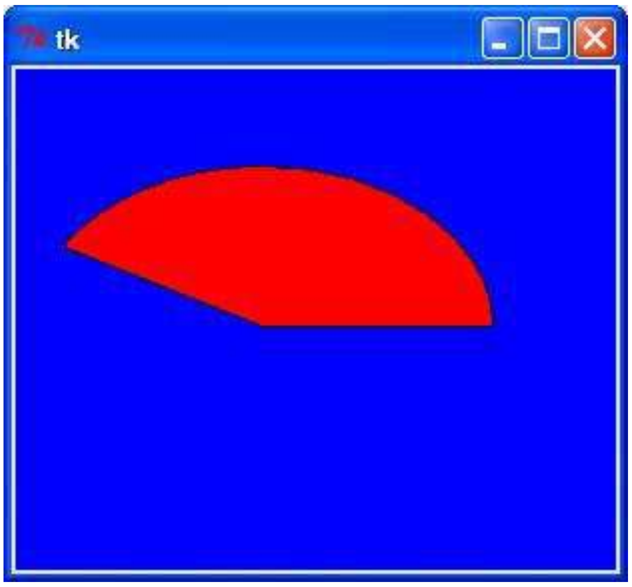
```
oval = canvas.create_polygon(x0, y0, x1, y1,...xn, yn, options)
```

Example

Try the following example yourself:

```
import Tkinter
import tkMessageBox

top = Tkinter.Tk()

C = Tkinter.Canvas(top, bg="blue", height=250, width=300)

coord = 10, 50, 240, 210
arc = C.create_arc(coord, start=0, extent=150, fill="red")

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

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



Checkbutton

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

You can also display images in place of text.

Syntax

Here is the simple syntax to create this widget:

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

Parameters

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

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

Following are commonly used methods for this widget:

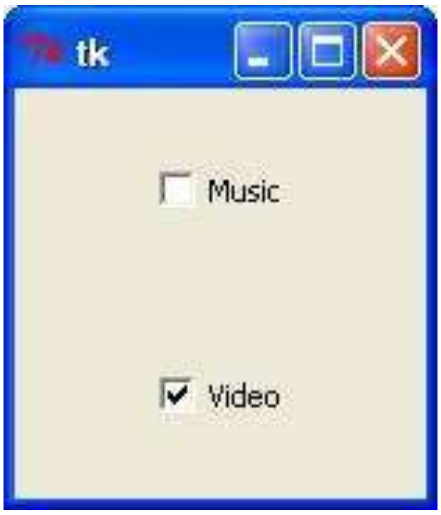
Medthod	**Description**
deselect()	Clears (turns off) the checkbutton.
flash()	Flashes the checkbutton a few times between its active and normal colors, but leaves it the way it started.
invoke()	You can call this method to get the same actions that would occur if the user clicked on the checkbutton to change its state.
select()	Sets (turns on) the checkbutton.
toggle()	Clears the checkbutton if set, sets it if cleared.

Example

Try the following example yourself:

```
from Tkinter import *
import tkMessageBox
import Tkinter

top = Tkinter.Tk()
CheckVar1 = IntVar()
CheckVar2 = IntVar()
C1 = Checkbutton(top, text = "Music", variable = CheckVar1, \
                 onvalue = 1, offvalue = 0, height=5, \
                 width = 20)
C2 = Checkbutton(top, text = "Video", variable = CheckVar2, \
                 onvalue = 1, offvalue = 0, height=5, \
                 width = 20)
C1.pack()
C2.pack()
top.mainloop()
```

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



Entry

The Entry widget is used to accept single-line text strings from a user.

- If you want to display multiple lines of text that can be edited, then you should use the *Text* widget.

- If you want to display one or more lines of text that cannot be modified by the user, then you should use the *Label* widget.

Syntax

Here is the simple syntax to create this widget:

```
w = Entry( 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.

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.
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.
font	The font used for the text.
exportselection	By default, if you select text within an Entry widget, it is automatically exported to the clipboard. To avoid this exportation, use exportselection=0.
fg	The color used to render 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.
xscrollcommand	If you expect that users will often enter more text than the onscreen size of the widget, you can link your entry widget to a scrollbar.

Methods

Following are commonly used methods for this widget:

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_from (index)	Sets the ANCHOR index position to the character selected by index, and selects that character.
select_present()	If there is a selection, returns true, else returns false.
select_range (start, end)	Sets the selection under program control. Selects the text starting at the start index, up to but not

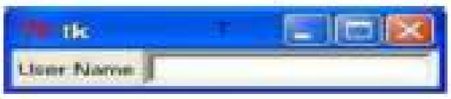
	including the character at the end index. The start position must be before the end position.
select_to (index)	Selects all the text from the ANCHOR position up to but not including the character at the given index.
xview (index)	This method is useful in linking the Entry widget to a horizontal scrollbar.
xview_scroll (number, what)	Used to scroll the entry horizontally. The what argument must be either UNITS, to scroll by character widths, or PAGES, to scroll by chunks the size of the entry widget. The number is positive to scroll left to right, negative to scroll right to left.

Example

Try the following example yourself:

```
from Tkinter import *


top = Tk()
L1 = Label(top, text="User Name")
L1.pack( side = LEFT)
E1 = Entry(top, bd =5)


E1.pack(side = RIGHT)


top.mainloop()
```

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


tk
User Name