Python  Programming

Day 13: Graphical User Interface

Graphical User Interface

GUI Library

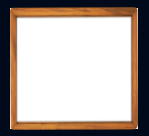
Widget

Color and symbol meaning

Hint

Preferred

Student's activity

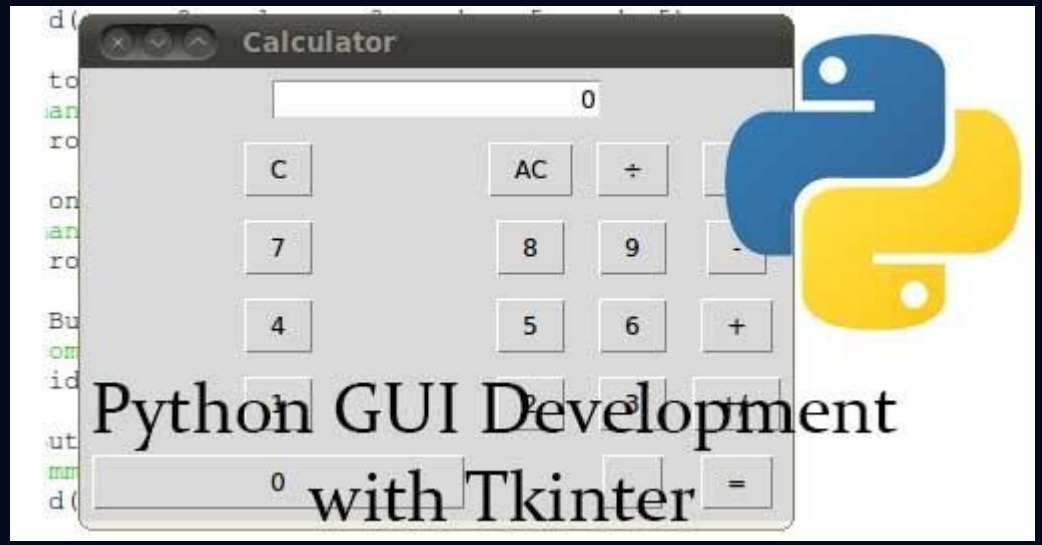


Practice code

	Keyword
	In-built functions
	Strings
	Output

Introduction to Python GUI

Until now we have focused on applications that can be run from the command line interface (CLI). Most users finds CLI programs unattractive. Python provides various options for developing **graphical user interfaces** (GUIs).



Introduction to Python GUI

Below is a list of some Python GUI Library

- Tkinter
- wxPython
- JPython
- GTK
- Qt

Some of the GUI libraries haven't been ported to Python 3 yet. At least **Tkinter** and Qt are known to be Python 3-compatible.

In this module we shall discuss **Tkinter**, a module in the Python standard library which serves as an interface to Tk, a simple toolkit.

Tkinter Library

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.

Tkinter Widgets

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

We present some of these widgets as well as a simple demonstration of their application.

Tkinter - Window Widget

A **window** is a GUI application main window upon which every other widgets are placed. Windows are primarily used as a container for other widgets. Windows are created using the **tkinter.Tk()** function

```
import tkinter
#create a new window widget
window = tkinter.Tk()
window.mainloop()
```


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
- Place

Geometry Management

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.

Geometry Management - place() Method

- **bordermode** : INSIDE (the default) to indicate that other options refer to the parent's inside (ignoring the parent's border); OUTSIDE otherwise.

- **height, width** : Height and width in pixels.

```
from tkinter import messagebox
import tkinter


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

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

B.pack()
B.place(bordermode=tkinter.OUTSIDE, height=100, width=100)
window.mainloop()
```

Geometry Management - pack() Method

- **anchor** : The exact spot of widget other options refer to: may be **N, E, S, W, NE, NW, SE, or SW**, compass directions indicating the corners and sides of widget; default is NW (the upper left corner of widget)

- **x, y** : Horizontal and vertical offset in pixels.

Geometry Management - grid() Method

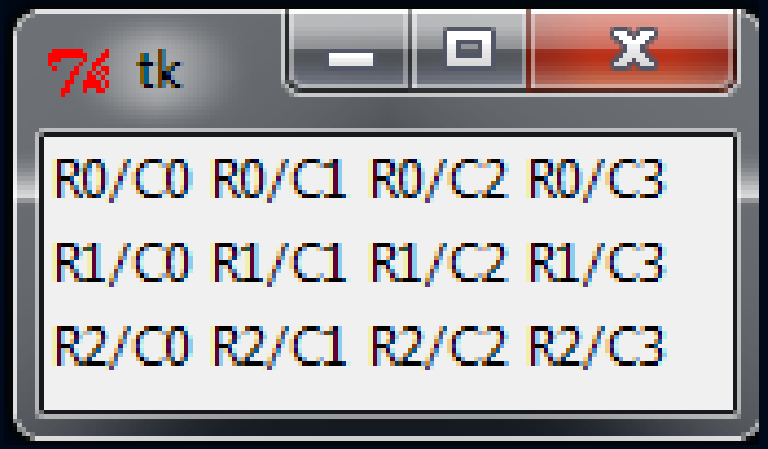
- **column :** The column to put widget in; default 0 (leftmost column).
- **columnspan:** How many columns widget occupies; default 1.
- **ipadx, ipady :**How many pixels to pad widget, horizontally and vertically, inside widget's borders.

```
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),
         borderwidth=1 ).grid(row=r,column=c)
root.mainloop( )
```


Geometry Management - grid() Method

- **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.

The code above outputs 12 labels arrayed in a 3 x 4 grid



Geometry Management - pack() Method

- **expand**: When set to true, widget expands to fill any space not otherwise used in widget's parent.

- **fill**: Determines whether widget fills any extra space allocated to it by the packer, or keeps its own minimal dimensions: NONE (default), X (fill only horizontally), Y (fill only vertically), or BOTH (fill both horizontally and vertically).

- **side**: Determines which side of the parent widget packs against: TOP (default), BOTTOM, LEFT, or RIGHT.

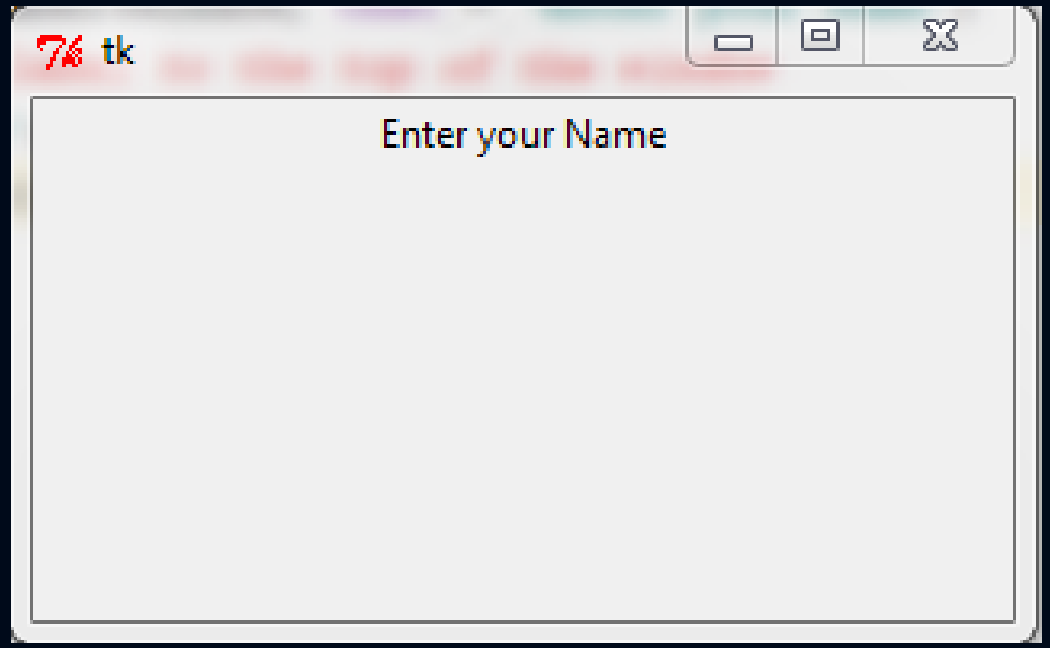
Tkinter - Label Widget

A **label** is a widget that displays text or images, typically that the user will just view but not otherwise interact with. Labels are used for such things as identifying controls or other parts of the user interface, providing textual feedback or results, etc.

```
import tkinter
#create a new window widget
window = tkinter.Tk()

lbl = tkinter.Label(window, text = 'Enter your Name')
# attaches the label to the top of the window
lbl.pack({'side':'top'})

window.mainloop()
```


Tkinter - Label Widget

Output



Tkinter - Entry Widget

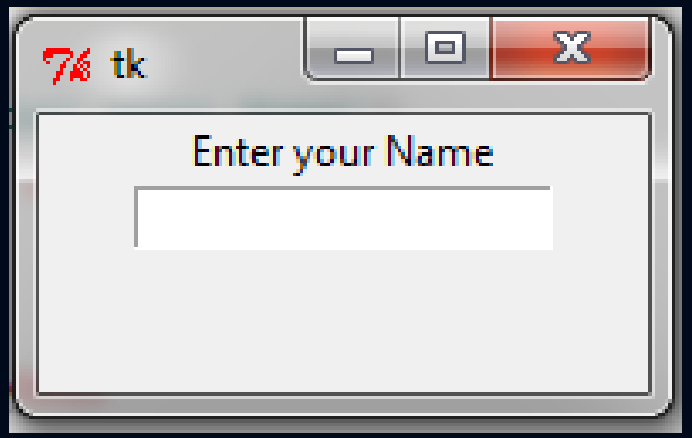
An **entry** presents the user with a single line text field that they can use to type in a string value. These can be just about anything: their name, a city, a password, social security number, and so on

```
#create a text entry widget called entName
entName = tkinter.Entry(window)
entName.pack({'side':'top'})
window.mainloop()
```

Entries are created using the **tkinter.Entry** function

Tkinter – Entry Widget

Output



Tkinter – Entry Options

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.

Tkinter – Entry Options

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.**

Tkinter – Button Widget

A **button**, unlike a frame or label, is very much designed for the user to interact with, and in particular, press to perform some action. Like labels, they can display text or images, but also have a whole range of new options used to control their behaviour.

Syntax

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

Tkinter – Button Widget

Buttons are created using the **tkinter.Button** function, and typically their contents and command callback are set up at the same time

```
#create button widget
btnSubmit =
tkinter.Button(window,
text='Submit',
command='submitForm')
btnSubmit.pack({'side':'top',
'pady':20})
window.mainloop()
```


Tkinter – Button Options

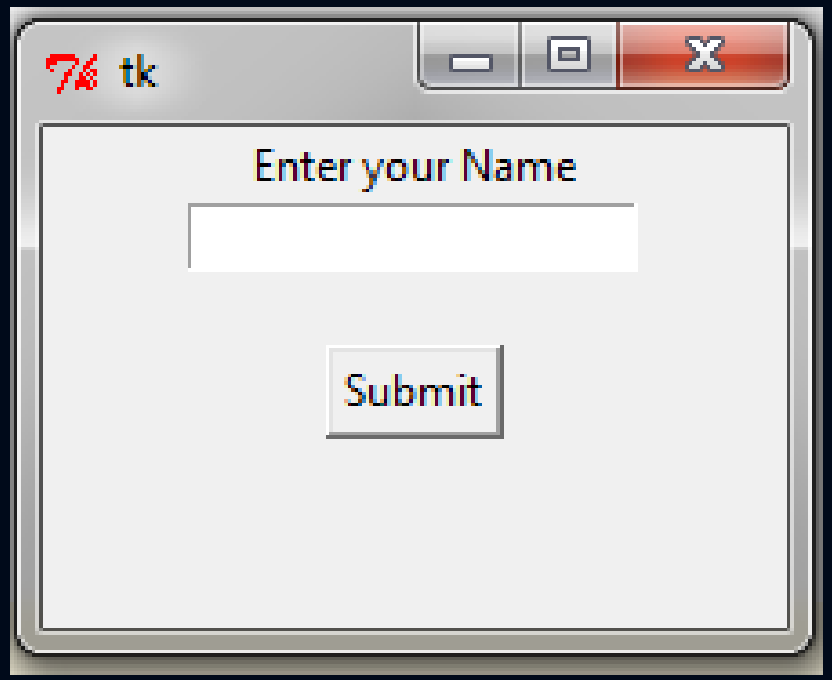
Option	Description
activebackground	Background color when the button is under the cursor.
activeforeground	Foreground color when the button is under the cursor.
bd	Border width in pixels. Default is 2.
bg	Normal background color.
command	Function or method to be called when the button is clicked.
fg	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 (for images).
highlightcolor	The color of the focus highlight when the widget has focus.

Tkinter – Button Options

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.**

Tkinter – Button Widget

Output



Tkinter – 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.

Syntax

Here is the simple syntax to create this widget –

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


Tkinter – Frame Widget

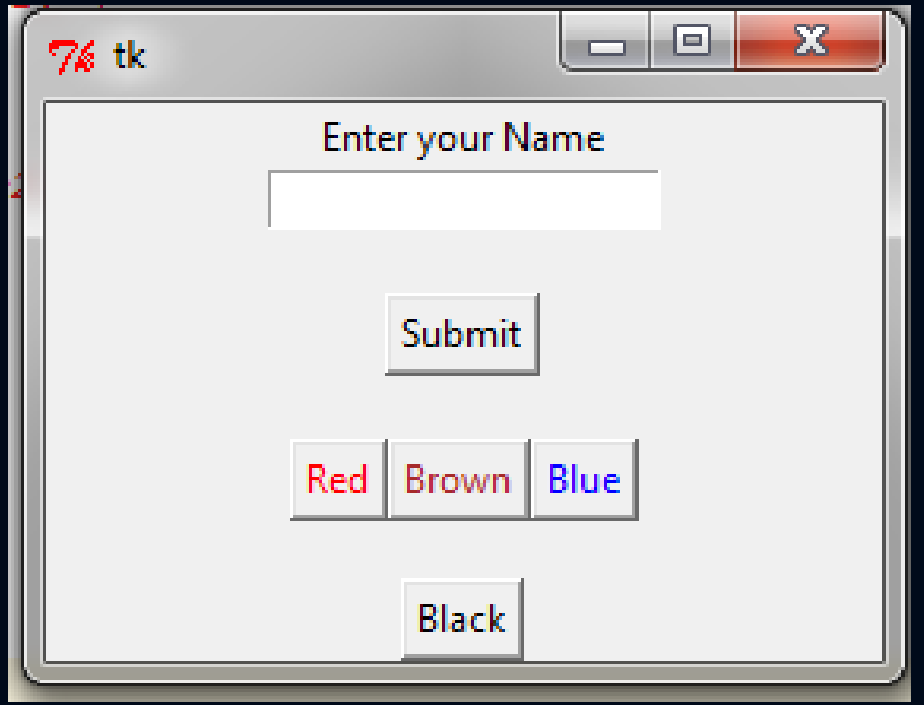
Here is the list of most commonly used options for this widget

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.

Tkinter – Frame Widget

```
#create a frame widget
from tkinter import *
frame = Frame(window)
frame.pack()

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

redbutton = Button(frame, text="Red", 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 = BOTTOM)
window.mainloop()
```

Output



Tkinter – Checkbox Widget

The **Checkbutton** widget is used to display a number of options as checkboxes. The user can select multiple options at a time. Checkbuttons are created using the **tkinter.Checkbutton** function, and typically set up at the same time

Syntax

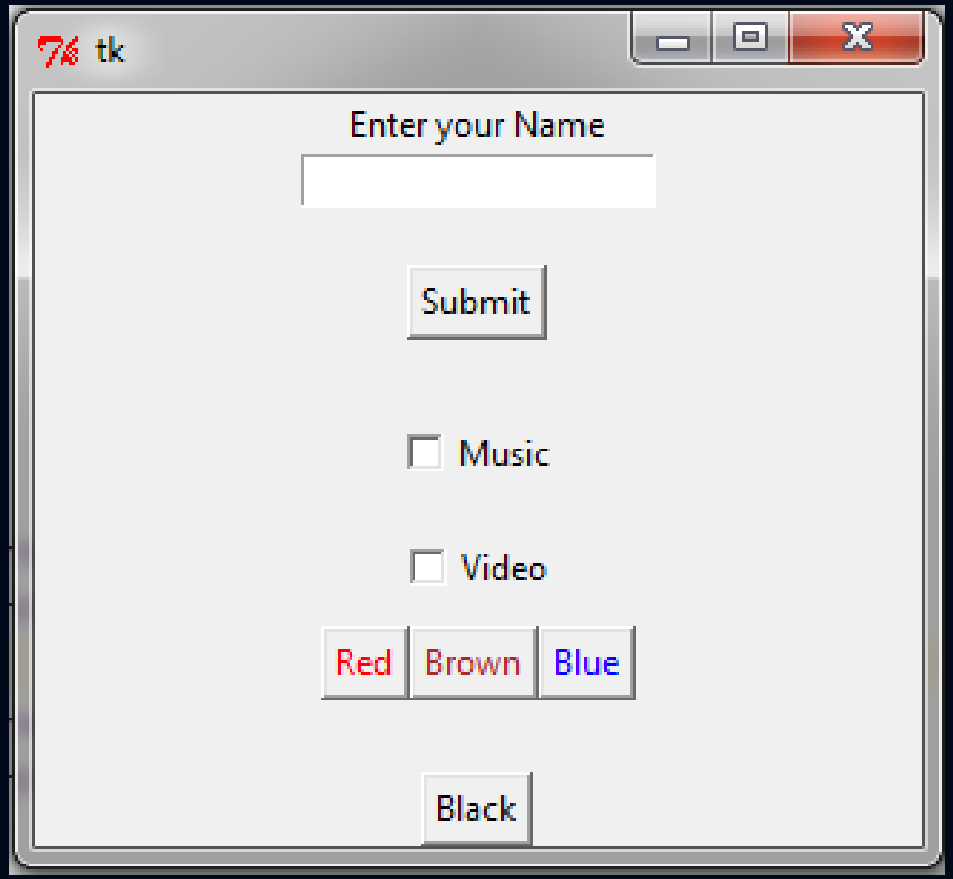
Here is the simple syntax to create this widget

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

Tkinter – Checkbox Widget

```
CheckVar1 = tkinter.IntVar()
CheckVar2 = tkinter.IntVar()

C1 = tkinter.Checkbutton(window, text = "Music",
variable = CheckVar1, \
            onvalue = 1, offvalue = 0, height=2, \
            width = 20)
C2 = tkinter.Checkbutton(window, text = "Video",
variable = CheckVar2, \
            onvalue = 1, offvalue = 0, height=2, \
            width = 20)
C1.pack()
C2.pack()

window.mainloop()
```

Output



Tkinter – Checkbox Options

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.

Tkinter – Checkbox Options

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**

Tkinter – Checkbox Options

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.**
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.**

Class Activity 1

Experiment with the frame widget and the 3 geometry management methods to position every other widgets on the main form:

Tkinter – Radiobutton Widget

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

Syntax

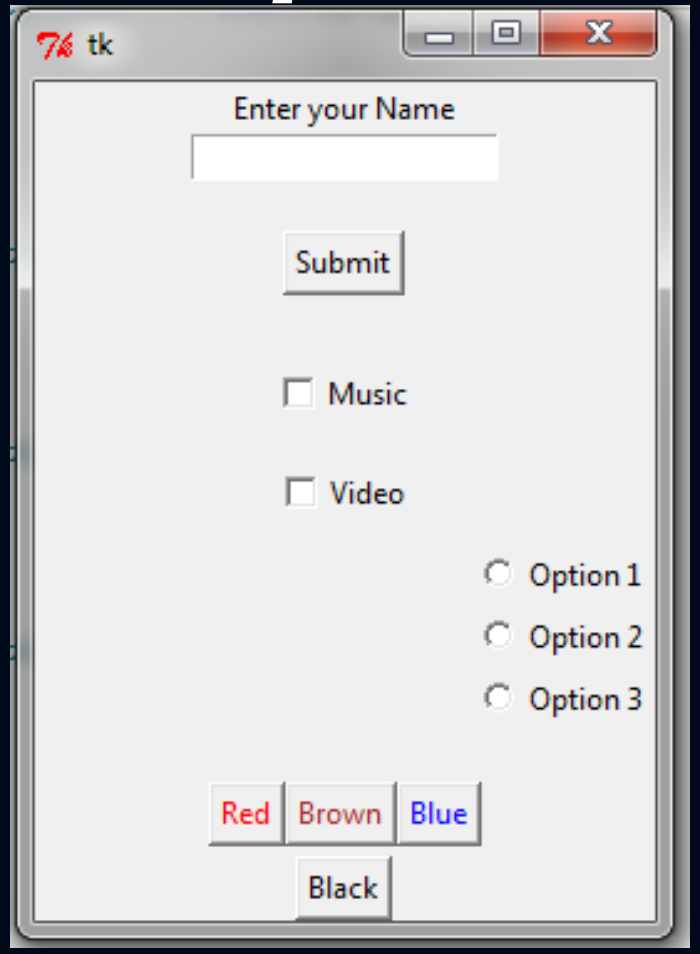
Here is the simple syntax to create this widget

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

Tkinter – Radiobutton Widget

```
#create a set of radiobuttons
def sel():
  selection = "You selected the option " + str(var.get())
  label.config(text = selection)

from tkinter import *
var = IntVar()
R1 = Radiobutton(window, text="Option 1", variable=var, value=1,
            command=sel)
R1.pack( anchor = E )

R2 = Radiobutton(window, text="Option 2", variable=var, value=2,
            command=sel)
R2.pack( anchor = E )

R3 = Radiobutton(window, text="Option 3", variable=var, value=3,
            command=sel)
R3.pack( anchor = E)

label = Label(window)
label.pack()
```

Output



Tkinter – Radiobutton Options

It has similar options as checkbox widget, below are peculiar options

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

Class Activity 2

Experiment with the configuration options of label, Button, Entry, Checkbutton, Radiobutton and scale widgets.

Next Lecture ...

Python Programming

Day 14: Graphical User Interface (2)