

# GUI Using Python

- Recommended Version: 2.7 or latest
- Source :

[http://www.tutorialspoint.com/python/python\\_gui\\_programming.htm](http://www.tutorialspoint.com/python/python_gui_programming.htm)

- **Tkinter:** Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
- **wxPython:** This is an open-source Python interface for wxWindows
- **JPython:** JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine

# Tkinter Programming

- 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 of the above-mentioned widgets to the GUI application.
  - Enter the main event loop to take action against each event triggered by the user.

- Program1

```
#!/usr/bin/python
```

```
import Tkinter
```

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

```
# Code to add widgets will go here...
```

```
top.mainloop()
```

# Tkinter Widgets

- Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.
- There are currently 15 types of widgets in Tkinter. We present these widgets as well as a brief description in the following table –

Operator	Description
<a href="#"><u>Button</u></a>	The Button widget is used to display buttons in your application.
<a href="#"><u>Canvas</u></a>	The Canvas widget is used to draw shapes, such as lines, ovals, polygons and rectangles, in your application.
<a href="#"><u>Checkbutton</u></a>	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
<a href="#"><u>Entry</u></a>	The Entry widget is used to display a single-line text field for accepting values from a user.
<a href="#"><u>Frame</u></a>	The Frame widget is used as a container widget to organize other widgets.
<a href="#"><u>Label</u></a>	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
<a href="#"><u>Listbox</u></a>	The Listbox widget is used to provide a list of options to a user.
<a href="#"><u>Menubutton</u></a>	The Menubutton widget is used to display menus in your application.
<a href="#"><u>Menu</u></a>	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
<a href="#"><u>Message</u></a>	The Message widget is used to display multiline text fields for accepting values from a user.
<a href="#"><u>Radiobutton</u></a>	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
<a href="#"><u>Scale</u></a>	The Scale widget is used to provide a slider widget.
<a href="#"><u>Scrollbar</u></a>	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
<a href="#"><u>Text</u></a>	The Text widget is used to display text in multiple lines.
<a href="#"><u>Toplevel</u></a>	The Toplevel widget is used to provide a separate window container.
<a href="#"><u>Spinbox</u></a>	The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
<a href="#"><u>PanedWindow</u></a>	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
<a href="#"><u>LabelFrame</u></a>	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
<a href="#"><u>tkMessageBox</u></a>	This module is used to display message boxes in your applications.

# Standard attributes

- Let us take a look at how some of their common attributes such as sizes, colors and fonts are specified.
  - [Dimensions](#)
  - [Colors](#)
  - [Fonts](#)
  - [Anchors](#)
  - [Relief styles](#)
  - [Bitmaps](#)
  - [Cursors](#)



# Dimensions

Various lengths, widths, and other dimensions of widgets can be described in many different units.

- If you set a dimension to an integer, it is assumed to be in pixels.
- You can specify units by setting a dimension to a string containing a number followed by.

Character	Description
c	Centimeters
i	Inches
m	Millimeters
p	Printer's points (about 1/72")

# Fonts

## Font object Fonts

You can create a "font object" by importing the `tkFont` module and using the `Font` class constructor –

```
import tkFont  
  
font = tkFont.Font ( option, ... )
```

Here is the list of options:

- **family:** The font family name as a string.
- **size:** The font height as an integer in points. To get a font n | use -n.
- **weight:** "bold" for boldface, "normal" for regular weight.
- **slant:** "italic" for italic, "roman" for unslanted.
- **underline:** 1 for underlined text, 0 for normal.
- **overstrike:** 1 for overstruck text, 0 for normal.

## Example

```
helv36 = tkFont.Font(family="Helvetica",size=36,weight="bold")
```

# Relief Styles

Here is list of possible constants which can be used for relief attribute.

- ▣ FLAT
- ▣ RAISED
- ▣ SUNKEN
- ▣ GROOVE
- ▣ RIDGE

## Example

```
from Tkinter import *
import Tkinter

top = Tkinter.Tk()

B1 = Tkinter.Button(top, text = "FLAT", relief=FLAT )
B2 = Tkinter.Button(top, text = "RAISED", relief=RAISED )
B3 = Tkinter.Button(top, text = "SUNKEN", relief=SUNKEN )
B4 = Tkinter.Button(top, text = "GROOVE", relief=GROOVE )
B5 = Tkinter.Button(top, text = "RIDGE", relief=RIDGE )

B1.pack()
B2.pack()
B3.pack()
B4.pack()
B5.pack()
```

# BIT MAPS



# 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.

# Button

- Program2

```
import Tkinter
import tkMessageBox
top = Tkinter.Tk()
def helloCallBack():
    tkMessageBox.showinfo( "Hello Python", "Hello World")
    B = Tkinter.Button(top, text ="Hello", command = helloCallBack)
    B.pack()
top.mainloop()
```

# Canvas

- 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.

- [program3](#)

```
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()
```

**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)
```



# Entry

- [program4](#)

```
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()
```

# Frame

- [program5](#)

```
from Tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
bottomframe = Frame(root)
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)

root.mainloop()
```

# Listbox

- Program6

```
from Tkinter import *  
import tkMessageBox  
import Tkinter
```

```
top = Tk()
```

```
Lb1 = Listbox(top)  
Lb1.insert(1, "Python")  
Lb1.insert(2, "Perl")  
Lb1.insert(3, "C")  
Lb1.insert(4, "PHP")  
Lb1.insert(5, "JSP")  
Lb1.insert(6, "Ruby")
```

```
Lb1.pack()  
top.mainloop()
```

# Radiobutton

```
from Tkinter import *
```

- program7

```
def sel():
```

```
    selection = "You selected the option " + str(var.get())
```

```
    label.config(text = selection)
```

```
root = Tk()
```

```
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 = Label(root)
```

```
label.pack()
```

```
root.mainloop()
```

# Menubutton

- [program8](#)

```
from Tkinter import *
import tkMessageBox
import Tkinter
top = Tk()
mb= Menubutton ( top, text="condiments",relief=RAISED )
mb.grid()
mb.menu = Menu ( mb, tearoff = 0 )
mb["menu"] = mb.menu

mayoVar = IntVar()
ketchVar = IntVar()

mb.menu.add_checkbutton ( label="mayo",
                           variable=mayoVar )
mb.menu.add_checkbutton ( label="ketchup",
                           variable=ketchVar )

mb.pack()
top.mainloop()
```

# Check button

## program9

```
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=15, width = 50)  
C1.pack()  
C2.pack()  
top.mainloop()
```

# Bring Image

# Putting a gif image on a canvas with Tkinter

## Program

```
from Tkinter import *
root=Tk()
    # create the canvas, size in pixels
canvas = Canvas(width = 300, height = 200, bg = 'yellow')
    # pack the canvas into a frame/form
canvas.pack(expand = YES, fill = BOTH)
    # load the .gif image file
    # put in your own gif file here, may need to add full path
gif1 = PhotoImage(file = 'dw.gif')
    # put gif image on canvas
    # pic's upper left corner (NW) on the canvas is at x=50 y=10
canvas.create_image(50, 10, image = gif1, anchor = NW)
    # run it ...
root.mainloop()
```

Thank You !!!