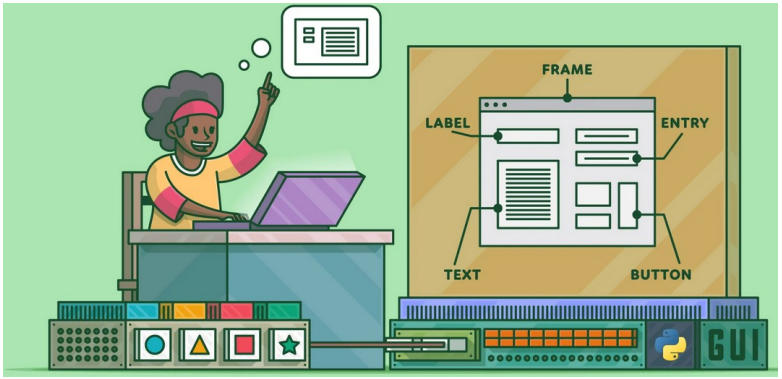
GUI Development using Python [Tkinter]

Tushar B. Kute,
http://tusharkute.com



Python GUI Programming

- Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below:
  - Tkinter: Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
  - wxPython: This is an open-source Python interface for wxWindows http://wxpython.org.
  - JPython: JPython is a Python port for Java, which gives Python scripts seamless access to Java class libraries on the local machine http://www.jython.org.

How to install tkinter ?

- It is by default available with standard python interpreter with idle IDE.
- Search for tkinter in Anaconda for installation.
- Ubuntu Linux:
  - `sudo apt install python3-tk`

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:
  - Example: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.

The Tkinter snippet

```
import Tkinter
top = Tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

The Button

- The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button, which is called automatically when you click the button.
- Syntax:

  w = Button (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.

The 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.
- Syntax:

  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.

The Canvas items

- The Canvas widget can support the following standard items:
  - **arc** . Creates an arc item.

```
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)
```

The Canvas items

- **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

```
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)
```

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:

  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.

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:

    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.

Label

- This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.
- It is also possible to underline part of the text (like to identify a keyboard shortcut), and span the text across multiple lines.
  - Syntax:

    w = Label (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.

ListBox

- The Listbox widget is used to display a list of items from which a user can select a number of items
  - Syntax:

    w = Listbox (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.

Menubutton

- A menubutton is the part of a drop-down menu that stays on the screen all the time. Every menubutton is associated with a Menu widget that can display the choices for that menubutton when the user clicks on it.
  - Syntax:

    w = Menubutton (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.

Message

- This widget provides a multiline and noneditable object that displays texts, automatically breaking lines and justifying their contents.
- Its functionality is very similar to the one provided by the Label widget, except that it can also automatically wrap the text, maintaining a given width or aspect ratio.
  - Syntax:

    w = Message (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.

RadioButton

- This widget implements a multiple-choice button, which is a way to offer many possible selections to the user, and let user choose only one of them.
- In order to implement this functionality, each group of radiobuttons must be associated to the same variable, and each one of the buttons must symbolize a single value. You can use the Tab key to switch from one radionbutton to another.
  - Syntax:

    w = Radiobutton (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.

Scale

- The Scale widget provides a graphical slider object that allows you to select values from a specific scale.
  - Syntax:

    w = Scale (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.

Scrollbar

- This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text, and Canvas. Note that you can also create horizontal scrollbars on Entry widgets.
  - Syntax:

    w = Scrollbar (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.

Toplevel

- Toplevel widgets work as windows that are directly managed by the window manager. They do not necessarily have a parent widget on top of them.
- Your application can use any number of top-level windows.
  - Syntax:

    w = Toplevel (option, ...)
  - Parameters:

    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.

Spinbox

- The Spinbox widget is a variant of the standard Tkinter Entry widget, which can be used to select from a fixed number of values.
  - Syntax:

    w = Spinbox(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.

Events and binding

<Button-1>	- left mouse button
<Button-2>	- middle mouse button (on 3 button mouse)
<Button-3>	- rightmost mouse button
<B1-Motion>	- mouse moved with left button depressed
<ButtonRelease-1>	- left button released
<Double-Button-1>	- double click on button 1
<Enter>	- mouse pointer entered widget
<Leave>	- mouse pointer left the widget
<FocusIn>	- Keyboard focus moved to a widget
<FocusOut>	- Keyboard focus moved to another widget
<Return>	- Enter key depressed
<Key>	- A key was depressed
<Shift-Up>	- Up arrow while holding Shift key
<Configure>	- widget changed size or location

Mouse events

Keyboard events

Event handling

- Event sources (widgets) can specify their handlers
  - Command handlers
  - Callbacks

Command handlers

- Use the 'command=' keyword followed by the command you want executed

```
from Tkinter import *
root = Tk()
Button (root, text='Press Me',
command=root.quit).pack(side=LEFT)
root.mainloop()
```

Callback

- A callback is the name of the function that is to be run in response of an event. It can be defined as a free standing function in our program or as a class member.

```
from Tkinter import *
def quit():
  print 'Hello, getting out of here'
  import sys; sys.exit()
widget = Button(None, text='Press me to quit' ,
command=quit)
widget.pack()
widget.mainloop()
```

Binding events

```
from tkinter import *
def hello(event):
    print('Double click to exit')
def quit(event):
    print('Caught a double click, leaving')
    import sys
    sys.exit()
def right(event):
    print('Right Clicked')

widget = Button(None, text='Hello Event World')
widget.pack()
widget.bind('<Button-1>', hello)
widget.bind('<Double-1>', quit)
widget.bind('<Button-3>', right)
widget.mainloop()
```

Geometry Managers

- Geometry manager is the layout manager for placing various widgets on the visible window.
- Tkinter possess three layout managers:
  - pack
  - grid
  - place
- The three layout managers pack, grid, and place should never be mixed in the same master window! Geometry managers serve various functions. They:
  - Arrange widgets on the screen
  - Register widgets with the underlying windowing system
  - Manage the display of widgets on the screen

pack

- Pack is the easiest to use of the three geometry managers of Tk and Tkinter.
- Instead of having to declare precisely where a widget should appear on the display screen, we can declare the positions of widgets with the pack command relative to each other.
- The pack command takes care of the details. Though the pack command is easier to use, this layout managers is limited in its possibilities compared to the grid and place mangers.
- For simple applications it is definitely the manager of choice. For example simple applications like placing a number of widgets side by side, or on top of each other.

pack - Example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
         bg='red', fg='white')
w.pack()
w = tk.Label(root, text='Green Grass',
         bg='green', fg='black')
w.pack()
w = tk.Label(root, text='Blue Sky',
         bg='blue', fg='white')
w.pack()

tk.mainloop()
```

pack with fill- Example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
         bg='red', fg='white')
w.pack(fill=tk.X)
w = tk.Label(root, text='Green Grass',
         bg='green', fg='black')
w.pack(fill=tk.X)
w = tk.Label(root, text='Blue Sky',
         bg='blue', fg='white')
w.pack(fill=tk.X)

tk.mainloop()
```

pack with padding- Example

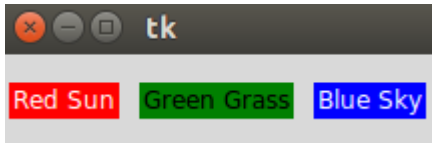
```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
         bg='red', fg='white')
w.pack(fill=tk.X, padx=10)
w = tk.Label(root, text='Green Grass',
         bg='green', fg='black')
w.pack(fill=tk.X, padx=10)
w = tk.Label(root, text='Blue Sky',
         bg='blue', fg='white')
w.pack(fill=tk.X, padx=10)

tk.mainloop()
```

pack with padding- Example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
        bg='red', fg='white')
w.pack(fill=tk.X, pady=10)
w = tk.Label(root, text='Green Grass',
        bg='green', fg='black')
w.pack(fill=tk.X, pady=10)
w = tk.Label(root, text='Blue Sky',
        bg='blue', fg='white')
w.pack(fill=tk.X, pady=10)

tk.mainloop()
```

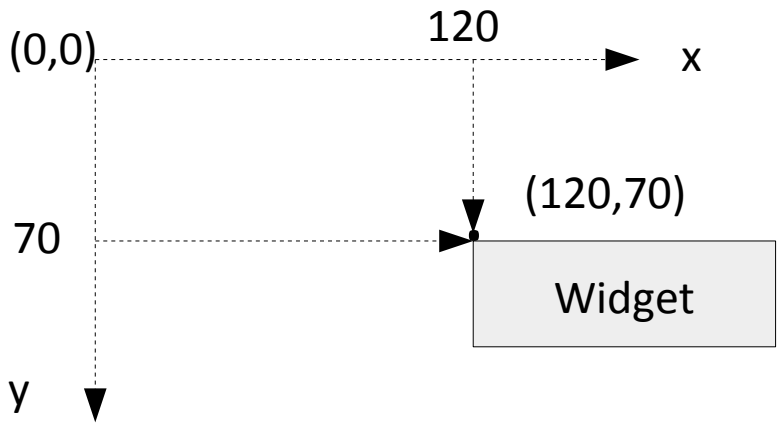
pack with side- Example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
        bg='red', fg='white')
w.pack(padx=5, pady=10, side=tk.LEFT)
w = tk.Label(root, text='Green Grass',
        bg='green', fg='black')
w.pack(padx=5, pady=10, side=tk.LEFT)
w = tk.Label(root, text='Blue Sky',
        bg='blue', fg='white')
w.pack(padx=5, pady=10, side=tk.LEFT)

tk.mainloop()
```



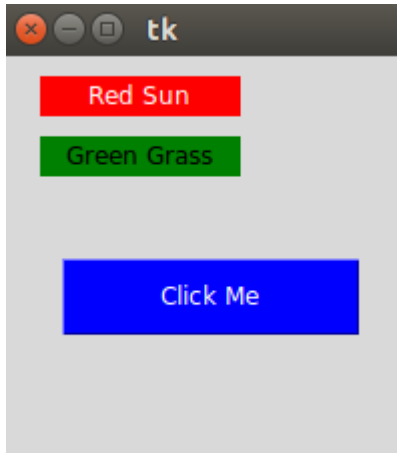
place

- The Place geometry manager allows you explicitly set the position and size of a window, either in absolute terms, or relative to another window.
- The place manager can be accessed through the place method.
- It can be applied to all standard widgets.



place – example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
        bg='red', fg='white')
w.place(x=20, y=10, width=100, height=20)
w = tk.Label(root, text='Green Grass',
        bg='green', fg='black')
w.place(x=20, y=40, width=100, height=20)
w = tk.Button(root, text='Click Me',
        bg='blue', fg='white')
w.place(x=30, y=100, width=150, height=40)

tk.mainloop()
```

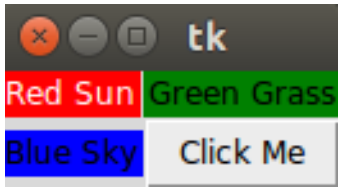


grid

- The Grid geometry manager places the widgets in a 2-dimensional table, which consists of a number of rows and columns.
- The position of a widget is defined by a row and a column number.
- Widgets with the same column number and different row numbers will be above or below each other.
- Correspondingly, widgets with the same row number but different column numbers will be on the same "line" and will be beside of each other, i.e. to the left or the right.

grid – example

```
import tkinter as tk

root = tk.Tk()

w = tk.Label(root, text='Red Sun',
         bg='red', fg='white')
w.grid(row=0, column=0)
w = tk.Label(root, text='Green Grass',
         bg='green', fg='black')
w.grid(row=0, column=1)
w = tk.Label(root, text='Blue Sky',
         bg='blue', fg='black')
w.grid(row=1, column=0)
w = tk.Button(root, text='Click Me',
         bg='yellow', fg='black')
w.grid(row=1, column=1)

tk.mainloop()
```



Adding Fonts

```
import tkinter as tk

root = tk.Tk()
root.geometry('200x200')

w = tk.Button(root, text='Click Me',
          bg='yellow', fg='black',
          font=('Arial', 25, 'bold'))

#styles- bold, underline, italic

w.pack()

tk.mainloop()
```

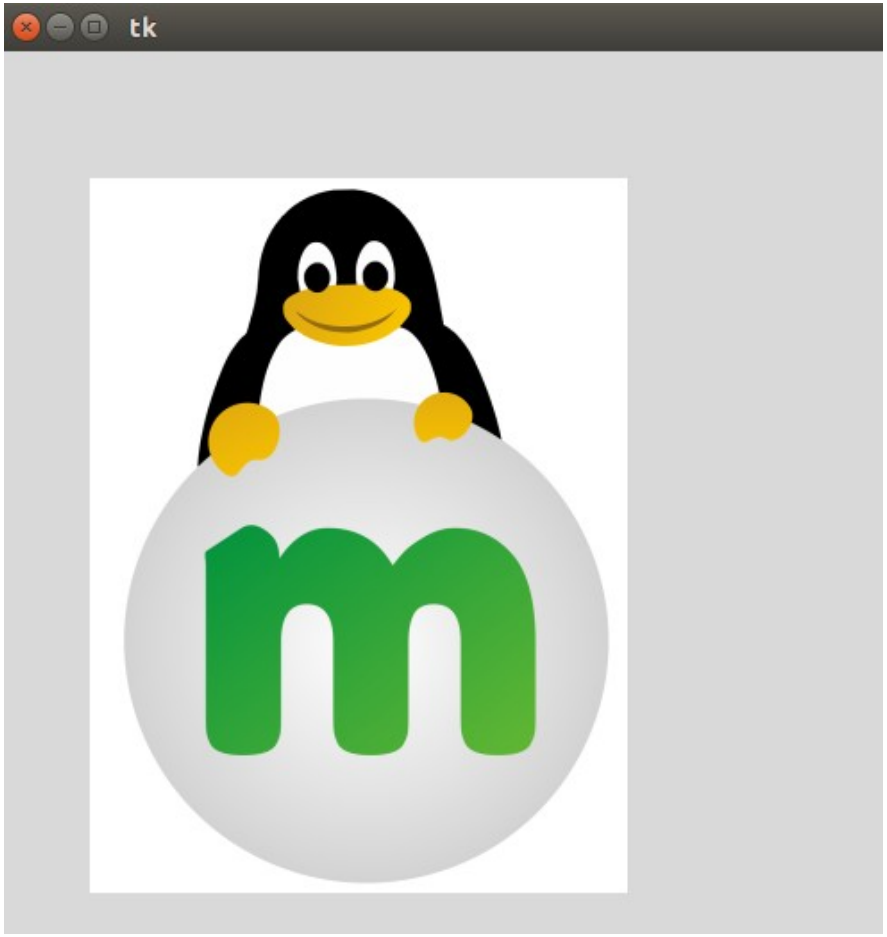
Inserting images

- In oder to insert the image, you need to install pil or pillow package.
- You can install it by,
  - `sudo pip3 install pillow -U`

Example:

```
import tkinter as tk

root = tk.Tk()
root.geometry('500x500')

from PIL import ImageTk, Image
img = ImageTk.PhotoImage(Image.open('logo.png'))
#logo.png is saved in current directory

tk.Label(root, image = img).place(x=50,y=70)

tk.mainloop()
```

Output



Useful resources

- https://www.python-course.eu
- https://mitu.co.in
- https://tutorialspoint.com

Thank you

@mitu_skillologies

/mITuSkillologies

@mitu_group

/company/mitu-skillologies

MITUSkillologies

Web Resources
https://mitu.co.in
http://tusharkute.com

contact@mitu.co.in

tushar@tusharkute.com