

Tkinter

What is it?

- Tkinter is a Python interface to the Tk graphics library.
 - Tk is a graphics library widely used and available everywhere
- Tkinter is included with Python as a library. To use it:
 - `from Tkinter import *`

What can it do?

- Tkinter gives you the ability to create Windows with widgets in them
- Definition: **widget** is a graphical component on the screen (button, text label, drop-down menu, scroll bar, picture, etc...)
- GUIs are built by arranging and combining different widgets on the screen.

First Tkinter Window

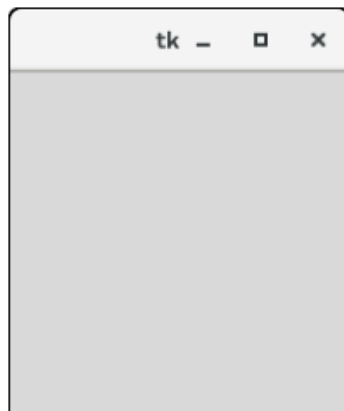
```
# File: hello1.py
from Tkinter import *
root = Tk() # Create the root (base) window where all widgets go
w = Label(root, text="Hello, world!") # Create a label with words
w.pack() # Put the label into the window
root.mainloop() # Start the event loop
```



Example

```
#!/usr/bin/python3
from tkinter import *
#creating the application main window.
top = Tk()
#Entering the event main loop
top.mainloop()
```

Output:



Tkinter widgets

There are various widgets like button, canvas, checkbutton, entry, etc. that are used to build the python GUI applications.

SN	Widget	Description
1	Button	The Button is used to add various kinds of buttons to the python application.
2	Canvas	The canvas widget is used to draw the canvas on the window.
3	Checkbutton	The Checkbutton is used to display the CheckButton on the window.
4	Entry	The entry widget is used to display the single-line text field to the user. It is commonly used to accept user values.
5	Frame	It can be defined as a container to which, another widget can be added and organized.
6	Label	A label is a text used to display some message or information about the other widgets.
7	ListBox	The ListBox widget is used to display a list of options to the user.
8	Menubutton	The Menubutton is used to display the menu items to the user.

9	Menu	It is used to add menu items to the user.
10	Message	The Message widget is used to display the message-box to the user.
11	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.
12	Scale	It is used to provide the slider to the user.
13	Scrollbar	It provides the scrollbar to the user so that the user can scroll the window up and down.
14	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.
14	Toplevel	It is used to create a separate window container.
15	Spinbox	It is an entry widget used to select from options of values.
16	PanedWindow	It is like a container widget that contains horizontal or vertical panes.
17	LabelFrame	A LabelFrame is a container widget that acts as the container
18	MessageBox	This module is used to display the message-box in the desktop based applications.

Python Tkinter Geometry

The Tkinter geometry specifies the method by using which, the widgets are represented on display. The python Tkinter provides the following geometry methods.

1. The pack() method
2. The grid() method
3. The place() method

Python Tkinter pack() method

The pack() widget is used to organize widget in the block. The positions widgets added to the python application using the pack() method can be controlled by using the various options specified in the method call.

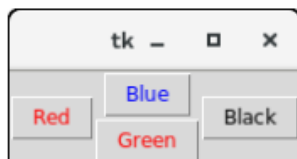
syntax

```
widget.pack(options)
```

Example

```
# !/usr/bin/python3
from tkinter import *
parent = Tk()
redbutton = Button(parent, text = "Red", fg = "red")
redbutton.pack( side = LEFT)
greenbutton = Button(parent, text = "Black", fg = "black")
greenbutton.pack( side = RIGHT )
bluebutton = Button(parent, text = "Blue", fg = "blue")
bluebutton.pack( side = TOP )
blackbutton = Button(parent, text = "Green", fg = "red")
blackbutton.pack( side = BOTTOM)
parent.mainloop()
```

Output:



Python Tkinter grid() method

The grid() geometry manager organizes the widgets in the tabular form. We can specify the rows and columns as the options in the method call. We can also specify the column span (width) or rowspan(height) of a widget.

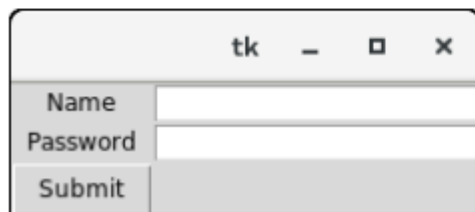
Syntax

```
widget.grid(options)
```

Example

```
# !/usr/bin/python3
from tkinter import *
parent = Tk()
name = Label(parent, text = "Name").grid(row = 0, column = 0)
e1 = Entry(parent).grid(row = 0, column = 1)
password = Label(parent, text = "Password").grid(row = 1, column = 0)
e2 = Entry(parent).grid(row = 1, column = 1)
submit = Button(parent, text = "Submit").grid(row = 4, column = 0)
parent.mainloop()
```

Output:



The screenshot shows a Tkinter window titled "tk" with standard window controls (minimize, maximize, close). Inside the window, there is a form layout. The first row contains a label "Name" and an empty text entry field. The second row contains a label "Password" and another empty text entry field. The third row contains a "Submit" button.

Python Tkinter place() method

The place() geometry manager organizes the widgets to the specific x and y coordinates.

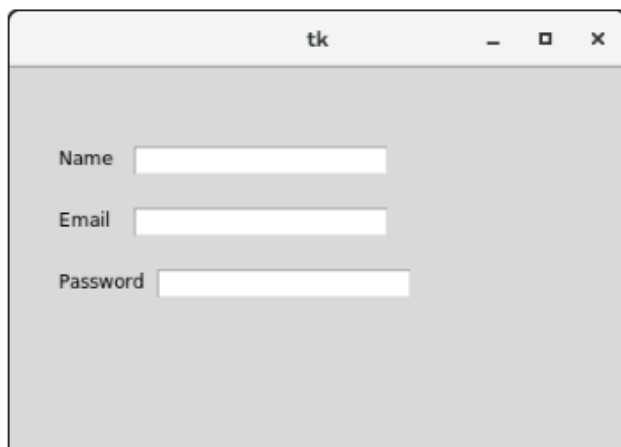
Syntax

```
widget.place(options)
```

Example

```
# !/usr/bin/python3
from tkinter import *
top = Tk()
top.geometry("400x250")
name = Label(top, text = "Name").place(x = 30,y = 50)
email = Label(top, text = "Email").place(x = 30, y = 90)
password = Label(top, text = "Password").place(x = 30, y = 130)
e1 = Entry(top).place(x = 80, y = 50)
e2 = Entry(top).place(x = 80, y = 90)
e3 = Entry(top).place(x = 95, y = 130)
top.mainloop()
```

Output:



Explain the code



File: hello1.py

```
from Tkinter import *
```

```
root = Tk()
```

Create the parent window. All applications have a "root" window. This is the parent of all other widgets, you should create only one!

```
w = Label(root, text="Hello, world!")
```

A Label is a widget that holds text
This one has a parent of "root"
That is the mandatory first argument
to the Label's constructor

```
w.pack()
```

Tell the label to place itself into the
root window and display. Without
calling pack the Label will NOT be
displayed!!!

```
root.mainloop()
```

Windows go into an "event loop" where they wait for things to
happen (buttons pushed, text entered, mouse clicks, etc...) or
Windowing operations to be needed (redraw, etc...). You must tell
the root window to enter its event loop or the window won't be
displayed!

Tkinter objects

- Label is a class, w is an object

- `w = Label(root, text="Hello, world!")`

Build it (called
instantiation)

- Call the “pack” operation:

- `w.pack()`
 - Hint: An operation is just a function... nothing more, nothing less.. it is just defined inside the class to act upon the object's current data.

Objects usually hide their data from anyone else and let other programmers access the data only through operations. (This is an OO concept called encapsulation)

More objects we can build

```
#Button1.py  
from Tkinter import *
```

```
root = Tk() # Create the root (base) window where all widgets go
```

```
w = Label(root, text="Hello, world!") # Create a label with words  
w.pack() # Put the label into the window
```

```
myButton = Button(root, text="Exit")  
myButton.pack()
```

```
root.mainloop() # Start the event loop
```

But nothing happens when we push the button! Lets fix that with an event!



Making the button do something

```
#Button2.py
from Tkinter import *

def buttonPushed():
    print "Button pushed!"
```

This says, whenever someone pushes the button, call the buttonPushed function. (Generically any function called by an action like this is a "callback")

```
root = Tk() # Create the root (base) window where all widgets go

w = Label(root, text="Hello, world!") # Create a label with words
w.pack() # Put the label into the window

myButton = Button(root, text="Exit", command=buttonPushed)
myButton.pack()

root.mainloop() # Start the event loop
```



The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the `global` keyword.

Example

If you use the `global` keyword, the variable belongs to the global scope:

```
def myfunc():
    global x
    x = "fantastic"

myfunc()

print("Python is " + x)
```

Python is fantastic

Output: -

Creating text entry box

General form for all widgets:

1. # Create the widget
 `widget = <widgetname>(parent, attributes...)`
2. `widget.pack()`
 pack the widget to make it show up

```
def createTextBox(parent):  
    tBox = Entry(parent)  
    tBox.pack()
```

From main call:
`createTextBox(root)`



Using a text entry box

To use a text entry box you must be able to get information from it when you need it. (Generally in response to an event)

For us, this means make the entry box global so we can get the info when a button is pressed

Using a text entry box

```
#Textentrybox1.py  
from Tkinter import *
```

```
# Hold onto a global reference for the root window  
root = None
```

```
# Hold onto the Text Entry Box also  
entryBox = None
```

```
def buttonPushed():  
    global entryBox  
    txt = entryBox.get()  
    print "The text is:",txt
```

Call the get() operation on the entry box
to get the text when button is pushed

```
def createTextBox(parent):  
    global entryBox  
    entryBox = Entry(parent)  
    entryBox.pack()
```

Create the global entry box!

```
def main():  
    global root  
    root = Tk() # Create the root (base) window where all widgets go  
  
    myButton = Button(root, text="Show Text",command=buttonPushed)  
    myButton.pack()  
    createTextBox(root)  
    root.mainloop() # Start the event loop
```

```
main()
```



Creating a label you can change

```
#changeable_label.py
# Use a StringVar to create a changeable label
from Tkinter import *

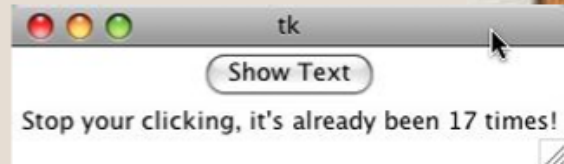
# Hold onto a global reference for the root window
root = None

# Changeable text that will go inside the Label
myText = None
count = 0 # Click counter

def buttonPushed():
    global myText
    global count
    count += 1
    myText.set("Stop your clicking, it's already been %d times!" %(count))

def addTextLabel(root):
    global myText
    myText = StringVar()
    myText.set("")
    myLabel = Label(root, textvariable=myText)
    myLabel.pack()

def main():
    global root
    root = Tk() # Create the root (base) window where all widgets go
    myButton = Button(root, text="Show Text",command=buttonPushed)
    myButton.pack()
    addTextLabel(root)
    root.mainloop() # Start the event loop
```



Set the text in the label
(call set method with a
string actual parameter)

Create a StringVar to hold text

Link the label to the StringVar

Layout management

- You may have noticed as we pack widgets into the window they always go under the previous widget
- What if we want to get them to go side-by-side or some other place?
- Most windowing toolkits have layout management systems to help you arrange widgets!

Layout management

- You've been using one – the packer is called when you pack()
- pack can have a side to pack on:
 - `myWidget.pack(side=LEFT)`
 - this tells pack to put this widget to the left of the next widget

Pack Examples

```
#pack_sample.py
from Tkinter import *

# Hold onto a global reference for the root window
root = None
count = 0 # Click counter

def addButton(root, sideToPack):
    global count
    name = "Button "+ str(count) +" "+sideToPack
    button = Button(root, text=name)
    button.pack(side=sideToPack)
    count +=1

def main():
    global root
    root = Tk() # Create the root (base) window where all widgets go
    addButton(root, LEFT) # Put the left side of the next widget close to me
    addButton(root, BOTTOM) # Put bottom of next widget close to me
    addButton(root, RIGHT) # Put right of next widget close to me
    addButton(root, BOTTOM) # Put bottom of next widget close to me
    root.mainloop() # Start the event loop

main()
```



Geometry Method in Python Tkinter

Tkinter provides many methods, one of them is the **geometry()** method. This method is used to set the dimensions of the Tkinter window and is used to set the position of the main window on the user's desktop.

Examples of Tkinter Geometry Method in Python

```
# importing only those functions which
# are needed
from tkinter import Tk, mainloop, TOP
from tkinter.ttk import Button

# creating tkinter window
root = Tk()

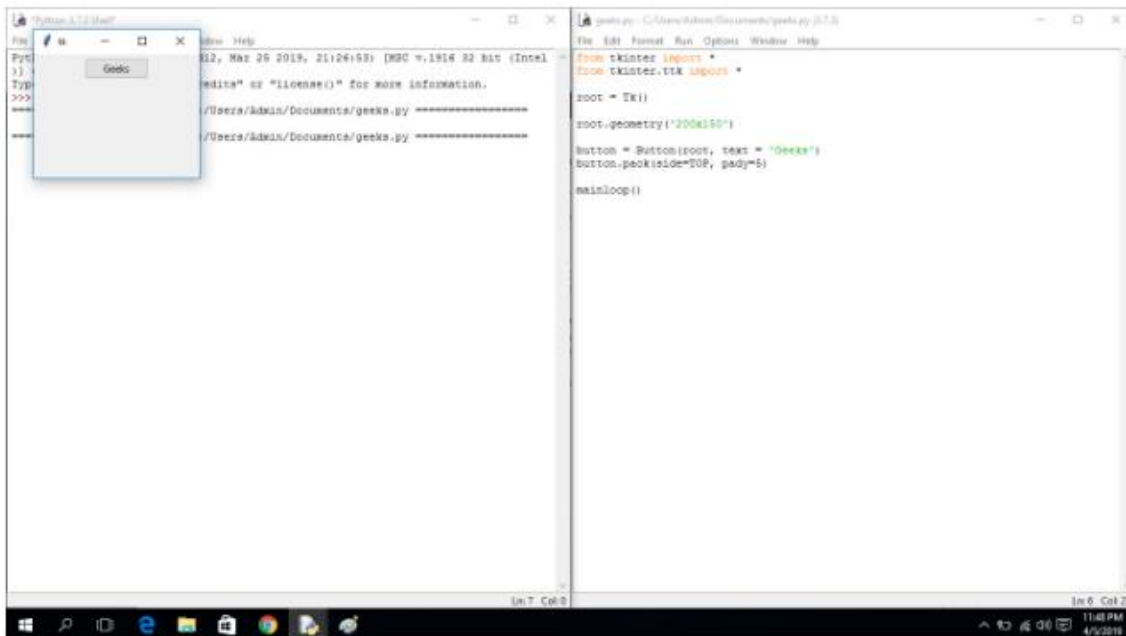
# creating fixed geometry of the
# tkinter window with dimensions 150x200
root.geometry('200x150')

# Create Button and add some text
button = Button(root, text = 'Geeks')
button.pack(side = TOP, pady = 5)

# Execute Tkinter
root.mainloop()
```

Output:

After running the application, you'll see that the size of the Tkinter window has changed, but the position on the screen is the same.



Using the geometry method to set the dimensions and positions of the Tkinter window.

```
# importing only those functions which
# are needed
from tkinter import Tk, mainloop, TOP
from tkinter.ttk import Button

# creating tkinter window
root = Tk()

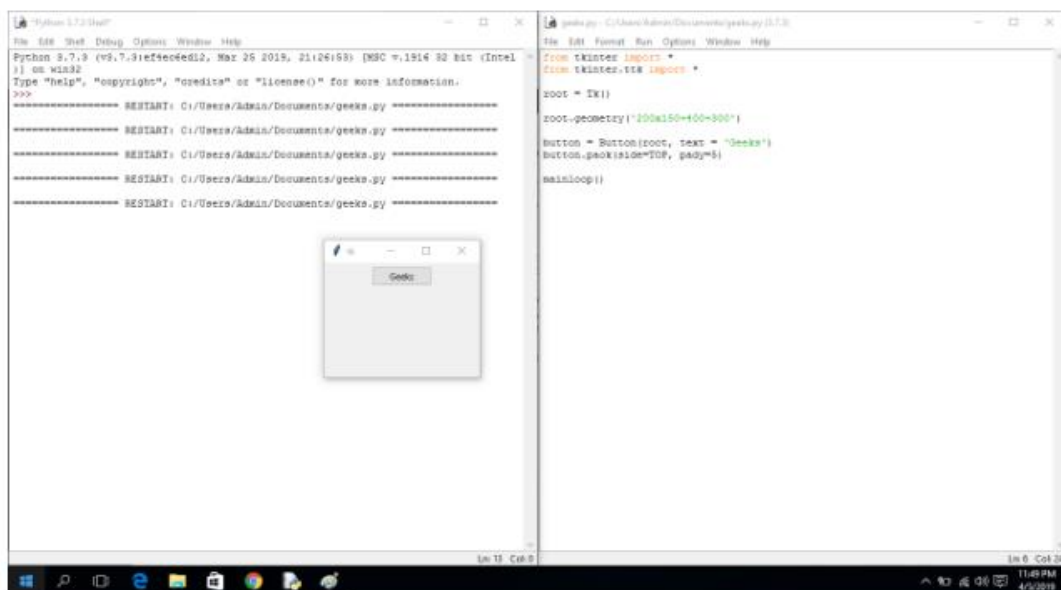
# creating fixed geometry of the
# tkinter window with dimensions 150x200
root.geometry('200x150+400+300')

# Create Button and add some text
button = Button(root, text = 'Geeks')
button.pack(side = TOP, pady = 5)

# Execute Tkinter
root.mainloop()
```

Output:

When you run the application, you'll observe that the position and size both are changed. Now the Tkinter window is appearing at a different position (400 shifted on X-axis and 300 shifted on Y-axis).



Python Tkinter Checkbutton

The Checkbutton is used to track the user's choices provided to the application. In other words, we can say that Checkbutton is used to implement the on/off selections.

Example

```
from tkinter import *

top = Tk()

top.geometry("200x200")

checkvar1 = IntVar()

checkvar2 = IntVar()

checkvar3 = IntVar()

chkbtn1 = Checkbutton(top, text = "C", variable = checkvar1, onvalue = 1, offvalue = 0, height = 2, width = 10)

chkbtn2 = Checkbutton(top, text = "C++", variable = checkvar2, onvalue = 1, offvalue = 0, height = 2, width = 10)

chkbtn3 = Checkbutton(top, text = "Java", variable = checkvar3, onvalue = 1, offvalue = 0, height = 2, width = 10)

chkbtn1.pack()

chkbtn2.pack()

chkbtn3.pack()

top.mainloop()
```

Output:



Python Tkinter Entry

The Entry widget is used to provide the single line text-box to the user to accept a value from the user. We can use the Entry widget to accept the text strings from the user. It can only be used for one line of text from the user. For multiple lines of text, we must use the text widget.

Example

```
# !/usr/bin/python3

from tkinter import *

top = Tk()

top.geometry("400x250")

name = Label(top, text = "Name").place(x = 30,y = 50)

email = Label(top, text = "Email").place(x = 30, y = 90)

password = Label(top, text = "Password").place(x = 30, y = 130)

sbmitbtn = Button(top, text = "Submit",activebackground = "pink", activeforeground = "blue").place(x = 30, y = 170)

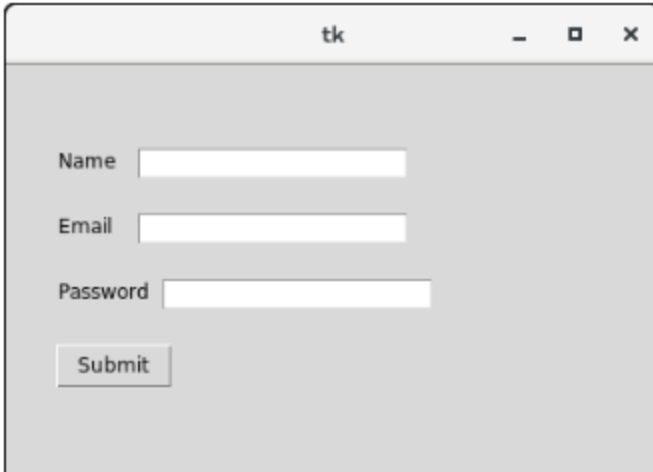
e1 = Entry(top).place(x = 80, y = 50)

e2 = Entry(top).place(x = 80, y = 90)

e3 = Entry(top).place(x = 95, y = 130)

top.mainloop()
```

Output:

A screenshot of a Tkinter window titled "tk". The window has a standard macOS-style title bar with a minus button, a maximize button, and a close button. The main content area is light gray. It contains three text input fields stacked vertically, each with a label to its left: "Name", "Email", and "Password". Below the "Password" field is a "Submit" button with a light gray background and a thin border.

Python Tkinter 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.

A list of possible options is given below.

SN	Option	Description
1	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.
2	bg	The background color displayed behind the widget.
3	bitmap	It is used to set the bitmap to the graphical object specified so that, the label can represent the graphics instead of text.
4	bd	It represents the width of the border. The default is 2 pixels.
5	cursor	The mouse pointer will be changed to the type of the cursor specified, i.e., arrow, dot, etc.
6	font	The font type of the text written inside the widget.
7	fg	The foreground color of the text written inside the widget.
8	height	The height of the widget.
9	image	The image that is to be shown as the label.
10	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.

Example 1

```
#!/usr/bin/python3

from tkinter import *

top = Tk()

top.geometry("400x250")

#creating label
uname = Label(top, text = "Username").place(x = 30,y = 50)

#creating label
password = Label(top, text = "Password").place(x = 30, y = 90)

sbmitbtn = Button(top, text = "Submit",activebackground = "pink", activeforeground = "blue").place(x = 30, y = 120)

e1 = Entry(top,width = 20).place(x = 100, y = 50)

e2 = Entry(top, width = 20).place(x = 100, y = 90)

top.mainloop()
```

Output:

