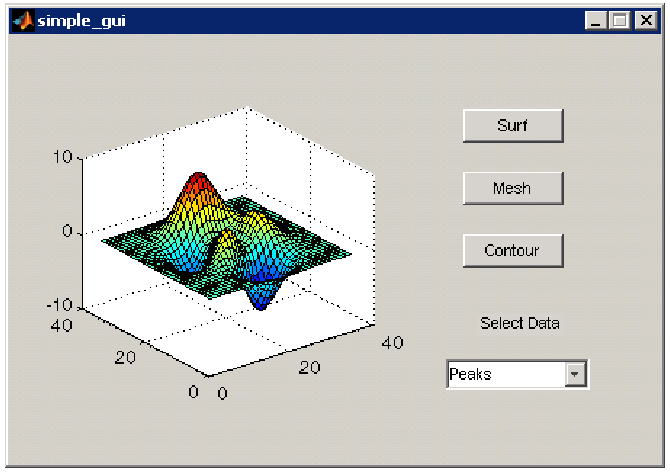
Computational Analysis of Physical Systems (Lecture 8)

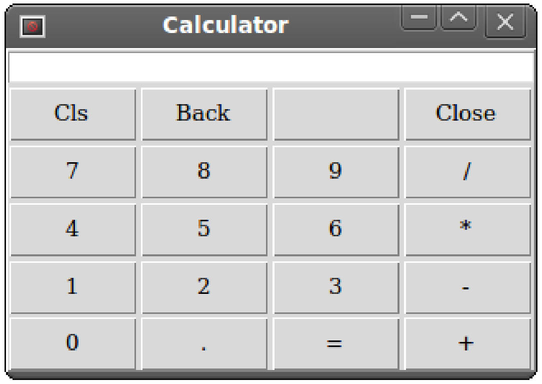
GUI Programming and Tkinter

What is a "GUI"?

Graphical **U**ser **I**nterface:

is a type of user interface that allows users to interact with electronic devices through **graphical icons** and **visual indicators**





What is Tkinter?

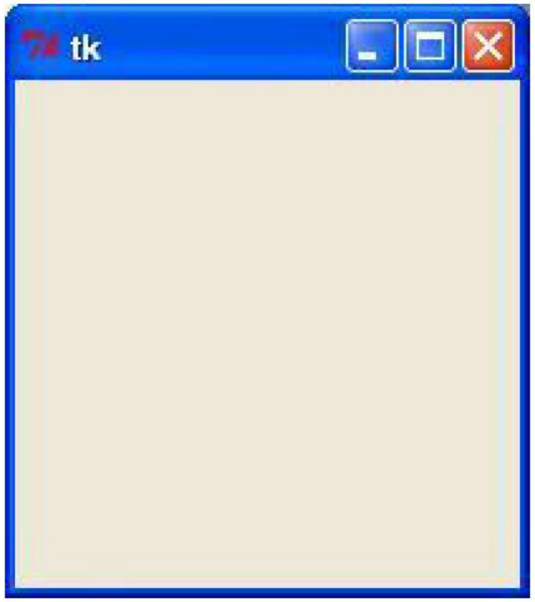
- Python’s standard GUI module.
- It is a layer for the “Tcl/Tk GUI Toolkit”.

There are other GUI modules for Python:

https://wiki.python.org/moin/GuiProgramming

Tkinter – import the module

```
#!/usr/bin/python

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

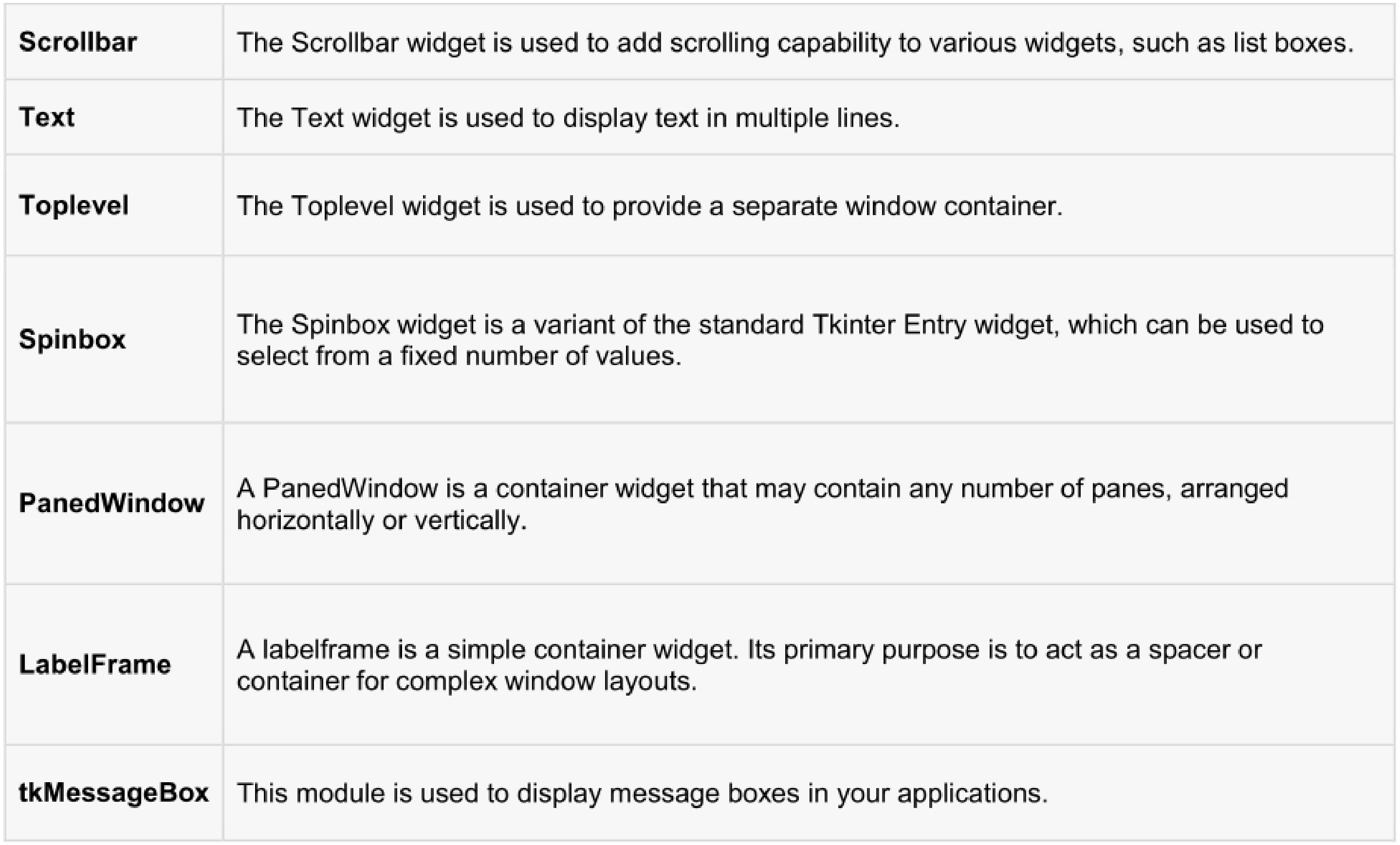


Widgets (1)

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

Operator	Description
Button	The Button widget is used to display buttons in your application.
Canvas	The Canvas widget is used to draw shapes, such as lines, ovals, polygons, and rectangles, in your application.
Checkbutton	The Checkbutton widget is used to display a number of options as checkboxes. The user can select multiple options at a time.
Entry	The Entry widget is used to display a single-line text field for accepting values from a user.
Frame	The Frame widget is used as a container widget to organize other widgets.
Label	The Label widget is used to provide a single-line caption for other widgets. It can also contain images.
Listbox	The Listbox widget is used to provide a list of options to a user.
Menubutton	The Menubutton widget is used to display menus in your application.
Menu	The Menu widget is used to provide various commands to a user. These commands are contained inside Menubutton.
Message	The Message widget is used to display multiline text fields for accepting values from a user.
Radiobutton	The Radiobutton widget is used to display a number of options as radio buttons. The user can select only one option at a time.
Scale	The Scale widget is used to provide a slider widget.

Widgets (2)



Scrollbar	The Scrollbar widget is used to add scrolling capability to various widgets, such as list boxes.
Text	The Text widget is used to display text in multiple lines.
Toplevel	The Toplevel widget is used to provide a separate window container.
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.
PanedWindow	A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.
LabelFrame	A labelframe is a simple container widget. Its primary purpose is to act as a spacer or container for complex window layouts.
tkMessageBox	This module is used to display message boxes in your applications.

Button (1)

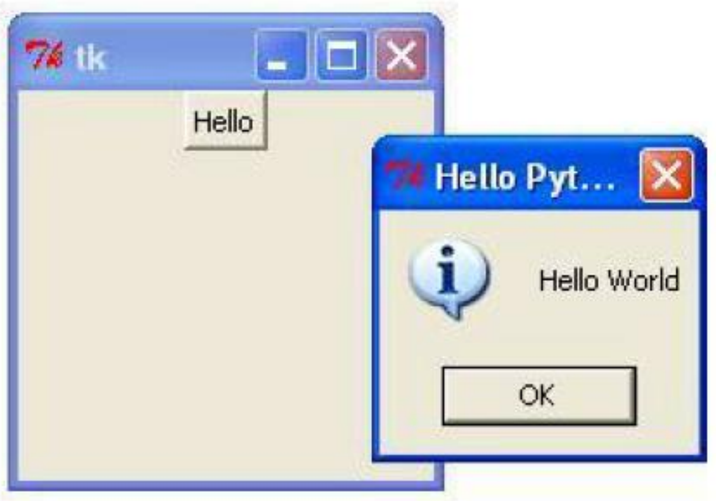
Button

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.

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

Button (2)

```
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()
```



Checkbutton (1)

Checkbutton

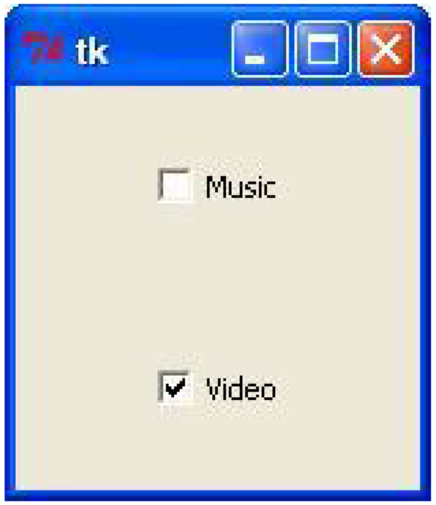
The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option.

You can also display images in place of text.

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

Checkbutton (2)

```
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()
```



Entry (1)

Entry

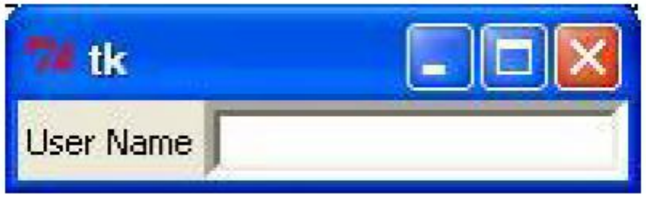
The Entry widget is used to accept single-line text strings from a user.

- If you want to display multiple lines of text that can be edited, then you should usethe *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.

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

Entry (2)

```
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 (1)

Frame

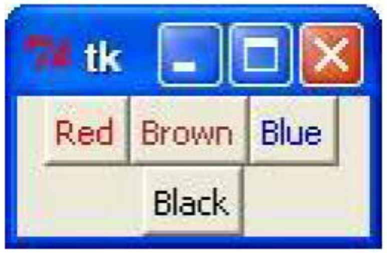
The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

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

Frame (2)

```
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()
```

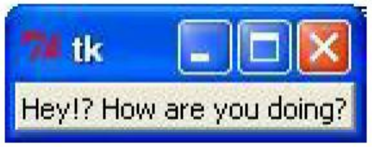


Label

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.

```
from Tkinter import *

root = Tk()

var = StringVar()
label = Label( root, textvariable=var, relief=RAISED )

var.set("Hey!? How are you doing?")
label.pack()
root.mainloop()
```



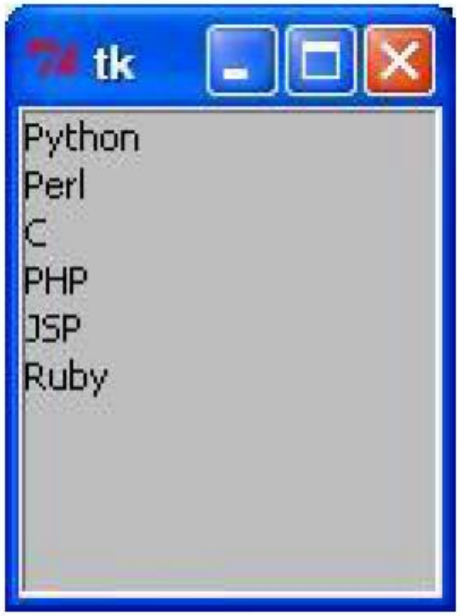
Hey!? How are you doing?

Listbox

Listbox

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

```
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()
```



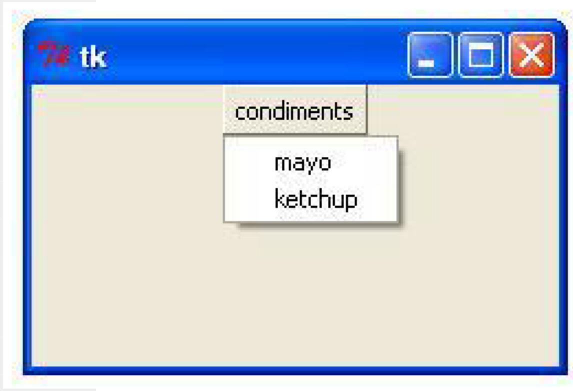
Python
Perl
C
PHP
JSP
Ruby

Menubutton

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.

```
from Tkinter import *
import tkMessageBox


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()
```



Menu (1)

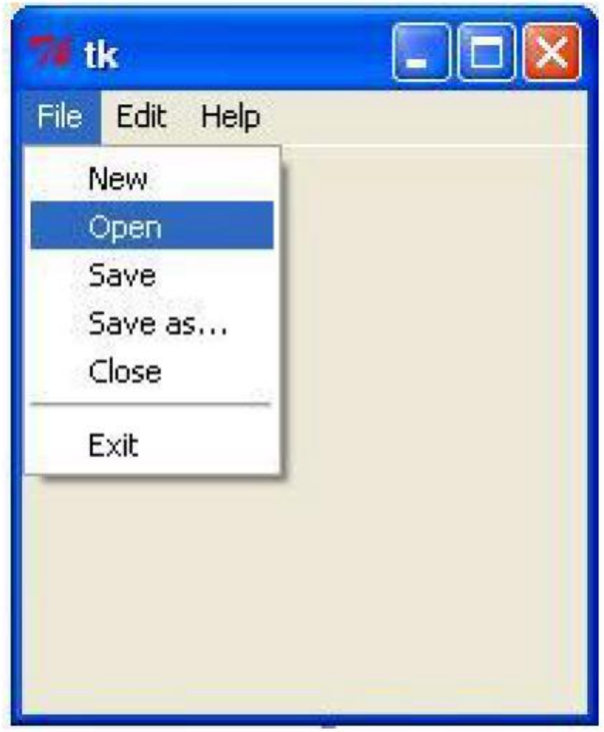
Menu

The goal of this widget is to allow us to create all kinds of menus that can be used by our applications. The core functionality provides ways to create three menu types: pop-up, toplevel, and pull-down.

```
from Tkinter import *
def donothing():
   filewin = Toplevel(root)
   button = Button(filewin, text="Do nothing button")
   button.pack()

root = Tk()
menubar = Menu(root)
filemenu = Menu(menubar, tearoff=0)
filemenu.add_command(label="New", command=donothing)
filemenu.add_command(label="Open", command=donothing)
filemenu.add_command(label="Save", command=donothing)
filemenu.add_command(label="Save as...", command=donothing)
filemenu.add_command(label="Close", command=donothing)

filemenu.add_separator()

filemenu.add_command(label="Exit", command=root.quit)
menubar.add_cascade(label="File", menu=filemenu)
editmenu = Menu(menubar, tearoff=0)
editmenu.add_command(label="Undo", command=donothing)

editmenu.add_separator()
```

Menu (2)

```
editmenu.add_command(label="Cut", command=donothing)
editmenu.add_command(label="Copy", command=donothing)
editmenu.add_command(label="Paste", command=donothing)
editmenu.add_command(label="Delete", command=donothing)
editmenu.add_command(label="Select All", command=donothing)

menubar.add_cascade(label="Edit", menu=editmenu)
helpmenu = Menu(menubar, tearoff=0)
helpmenu.add_command(label="Help Index", command=donothing)
helpmenu.add_command(label="About...", command=donothing)
menubar.add_cascade(label="Help", menu=helpmenu)

root.config(menu=menubar)
root.mainloop()
```



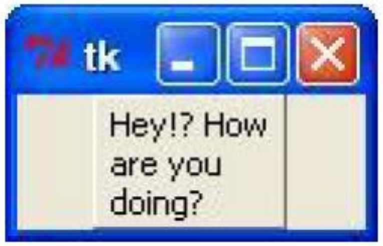
Message

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.

```
from Tkinter import *

root = Tk()

var = StringVar()
label = Message( root, textvariable=var, relief=RAISED )

var.set("Hey!? How are you doing?")
label.pack()
root.mainloop()
```



Hey!? How
are you
doing?

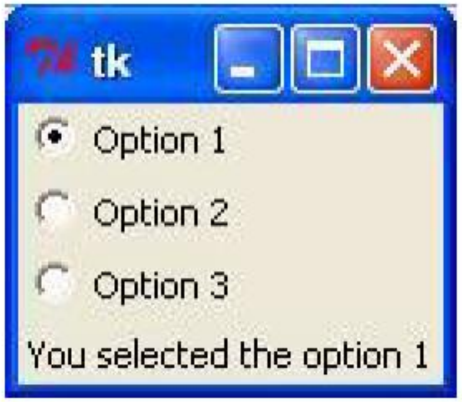
Radiobutton

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.

```
from Tkinter import *

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()
```



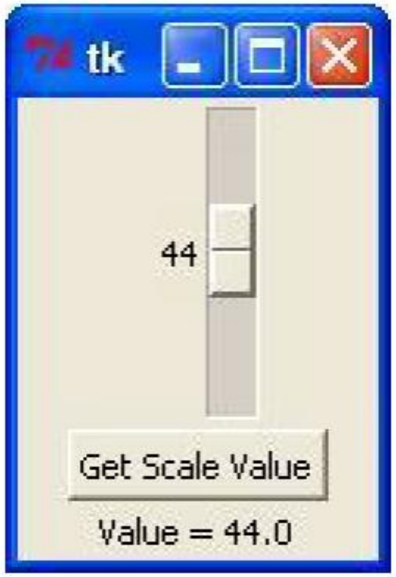
```
(●) Option 1
( ) Option 2
( ) Option 3
You selected the option 1
```

Scale

Scale

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

```
from Tkinter import *

def sel():
   selection = "Value = " + str(var.get())
   label.config(text = selection)

root = Tk()
var = DoubleVar()
scale = Scale( root, variable = var )
scale.pack(anchor=CENTER)

button = Button(root, text="Get Scale Value", command=sel)
button.pack(anchor=CENTER)

label = Label(root)
label.pack()

root.mainloop()
```



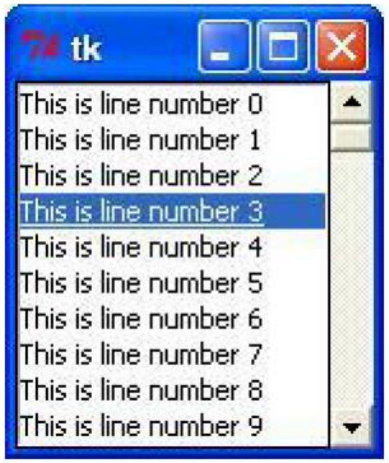
Value = 44.0

Scrollbar

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.

```
from Tkinter import *

root = Tk()
scrollbar = Scrollbar(root)
scrollbar.pack( side = RIGHT, fill=Y )

mylist = Listbox(root, yscrollcommand = scrollbar.set )
for line in range(100):
   mylist.insert(END, "This is line number " + str(line))

mylist.pack( side = LEFT, fill = BOTH )
scrollbar.config( command = mylist.yview )

mainloop()
```



```
This is line number 0
This is line number 1
This is line number 2
This is line number 3
This is line number 4
This is line number 5
This is line number 6
This is line number 7
This is line number 8
This is line number 9
```

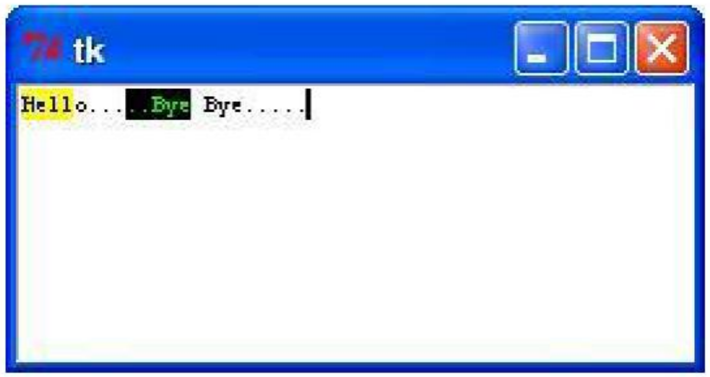
Text

Text

Text widgets provide advanced capabilities that allow you to edit a multiline text and format the way it has to be displayed, such as changing its color and font.

You can also use elegant structures like tabs and marks to locate specific sections of the text, and apply changes to those areas. Moreover, you can embed windows and images in the text because this widget was designed to handle both plain and formatted text.

```
from Tkinter import *

def onclick():
   pass

root = Tk()
text = Text(root)
text.insert(INSERT, "Hello.....")
text.insert(END, "Bye Bye.....")
text.pack()

text.tag_add("here", "1.0", "1.4")
text.tag_add("start", "1.8", "1.13")
text.tag_config("here", background="yellow", foreground="blue")
text.tag_config("start", background="black", foreground="green")
root.mainloop()
```



```
Hello.....Bye Bye.....
```

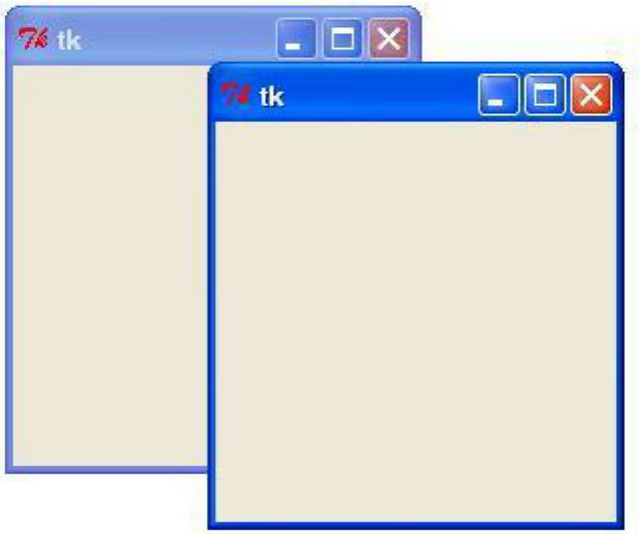
Toplevel

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.

```
from Tkinter import *

root = Tk()
top = Toplevel()

top.mainloop()
```

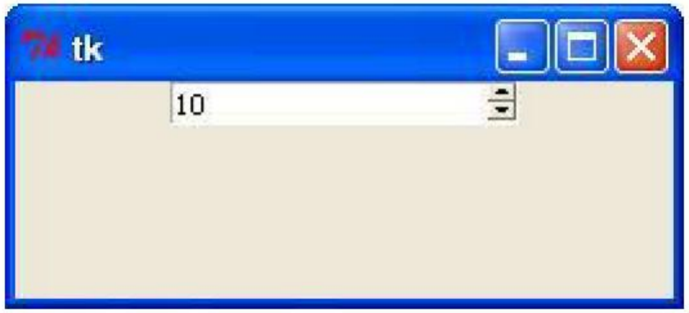


Spinbox

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.

```
from Tkinter import *

master = Tk()

w = Spinbox(master, from_=0, to=10)
w.pack()

mainloop()
```



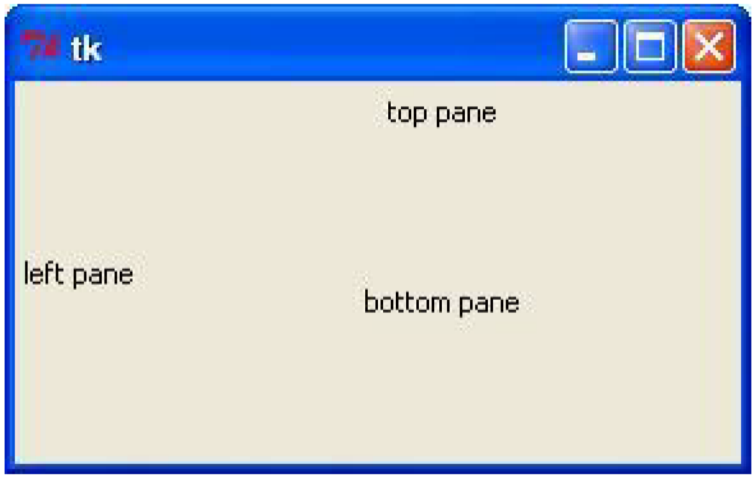
PanedWindow

PanedWindow

A PanedWindow is a container widget that may contain any number of panes, arranged horizontally or vertically.

Each pane contains one widget, and each pair of panes is separated by a moveable (via mouse movements) sash. Moving a sash causes the widgets on either side of the sash to be resized.

```
from Tkinter import *

m1 = PanedWindow()
m1.pack(fill=BOTH, expand=1)

left = Label(m1, text="left pane")
m1.add(left)

m2 = PanedWindow(m1, orient=VERTICAL)
m1.add(m2)

top = Label(m2, text="top pane")
m2.add(top)

bottom = Label(m2, text="bottom pane")
m2.add(bottom)

mainloop()
```



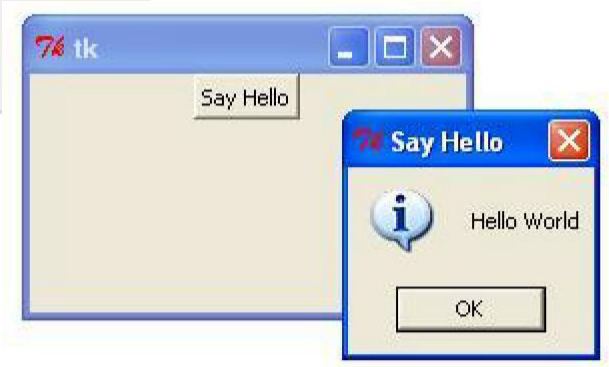
tkMessageBox

tkMessageBox

The tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are showinfo, showwarning, showerror, askquestion, askokcancel, askyesno, and askretryignore.

```
import Tkinter
import tkMessageBox

top = Tkinter.Tk()
def hello():
   tkMessageBox.showinfo("Say Hello", "Hello World")

B1 = Tkinter.Button(top, text = "Say Hello", command = hello)
B1.pack()

top.mainloop()
```



Example – Simple Clock

```
import Tkinter as tk
import time

def update_timeText():
    # Get the current time, note you can change the format as you wish
    current = time.strftime("%H:%M:%S")
    # Update the timeText Label box with the current time
    timeText.configure(text=current)
    # Call the update_timeText() function after 1 second
    root.after(1000, update_timeText)

root = tk.Tk()
root.wm_title("Simple Clock Example")

# Create a timeText Label (a text box)
timeText = tk.Label(root, text="", font=("Helvetica", 150))
timeText.pack()
update_timeText()
root.mainloop()
```

Example – Monte Carlo pi calc.

```
from Tkinter import *
from random import random

def montecarlo():
    all= int(E1.get()) #string to integer
    inside=0
    for i in range(all):
        x,y=random(),random()
        if (x**2+y**2)**(0.5)<1: inside=inside+1
    mypi=4.0*(float(inside)/all)
    print ("The value of pi for %d points is %f"%(all,mypi))

top = Tk()
L1 = Label(top, text="Number of points")
L1.pack( side = LEFT)
E1 = Entry(top, bd =5)
E1.pack(side = LEFT)
B1 = Button(top, text="FIND MY PI", width=10, command=montecarlo)
B1.pack(side=RIGHT)
mainloop()
```