PYTHON GUI

Introduction to Tkinter

- ϖ Python provides various options for developing graphical user interfaces (GUIs).
- ϖ Tkinter
  - ∞ Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.
  - ∞ Tk is called Tkinter in Python, or to be precise, Tkinter is the Python interface for Tk.
  - ∞ Tkinter is an acronym for "Tk interface".
  - ∞ Tk was developed as a GUI extension for the Tcl scripting language by John Ousterhout.
  - ∞ The first release was in 1991.
- ϖ Tk provides the following widgets

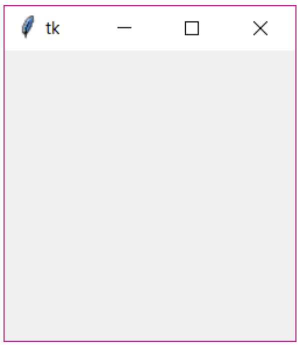
Button	canvas	checkbutton
combobox	entry	frame
Label	Labelframe	listbox
Menu	menubutton	message
Notebook	tk_optionMenu	panedwindow
progressbar	radiobutton	scale
Scrollbar	separator	sizegrip
Spinbox	text	treeview

- ϖ It provides the following top-level windows
  - ∞ tk_chooseColor
    - δ pops up a dialog box for the user to select a color.
  - ∞ tk_chooseDirectory
    - δ pops up a dialog box for the user to select a directory.
  - ∞ tk_dialog
    - δ creates a modal dialog and waits for a response.
  - ∞ tk_getOpenFile
    - δ pops up a dialog box for the user to select a file to open.
  - ∞ tk_getSaveFile
    - δ pops up a dialog box for the user to select a file to save.
  - ∞ tk_messageBox
    - δ pops up a message window and waits for a user response.
  - ∞ tk_popup

- δ posts a popup menu.
- ∞ toplevel
  - δ creates and manipulates toplevel widgets.
- ϖ Tk also provides three geometry managers
  - ∞ place
    - δ positions widgets at absolute locations
  - ∞ grid
    - δ arranges widgets in a grid
  - ∞ pack
    - δ packs widgets into a cavity
- ϖ In the GUI program, it has to be able to handle all sorts of different kinds of events and interactions.
  - ∞ This requirement is often implemented using a programming construct called an event loop.
  - ∞ An event loop is the central control structure of a program.
  - ∞ It waits for an event to happen, and then dispatches the appropriate handler.
  - ∞ To do this mainloop() has to be called once and only once, when the applications is ready to run.
    - δ It is an infinite loop used to run the application, wait for an event to occur and process the event till the window is not closed.

Tk Window

- ϖ tkinter has to be imported using
  - ∞ from tkinter import *
- ϖ To create a window
  - ∞ wind = Tk()
- ϖ To run the application
  - ∞ wind.mainloop()

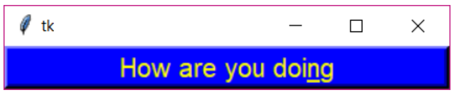
from tkinter import *	
w = Tk()	
mainloop()	

pack()

- ϖ This is an implementation class.
- ϖ Pack the widgets to the main application.

Labels

- ϖ A Label is a Tkinter Widget class, which is used to display text or an image.
- ϖ The label is a widget that the user just views but not interact with.
  - ∞ w = Label (master, option, ...)
    - δ master − represents the parent window got by calling Tk().
    - δ Options
      - ⇒ bg – sets background color for the label.
      - ⇒ cursor – can be "arrow‟, "circle‟, "clock‟, "cross‟, "dotbox‟, "exchange‟, "fleur‟, "heart‟, "man‟, "mouse‟, "pirate‟, "plus‟, "shuttle‟, "sizing‟, "spider‟, "spraycan‟, "star‟, "target‟, "tcross‟, "trek‟, "watch‟
      - ⇒ bd - size of the border around the indicator. Default is 2 pixels.
      - ⇒ font - specifies in what font that text will be displayed.
      - ⇒ text - string containing the text.
        - ₪ Internal newlines ("\n") will force a line break.
      - ⇒ width - width of the label in characters.
        - ₪ If this option is not set, the label will be sized to fit its contents.
      - ⇒ underline - Display an underline (_) below the nth letter of the text, counting from 0, by setting this option to n.
        - ₪ The default is underline=-1, which means no underlining.
      - ⇒ fg-specifies the color of the text
      - ⇒ relief - Relief specifies the type of the border.
        - ₪ Values are SUNKEN, RAISED, GROOVE, and RIDGE.

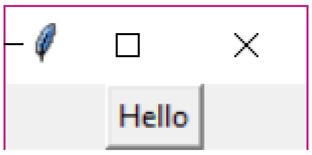
from tkinter import *	 tk How are you doing
root = Tk()	
var = StringVar()	
label = Label(root, text='How are you doing', relief=RAISED, bg='blue', cursor='cross', bd='4', font='Arial', width = 30, underline=15, fg='Yellow')	
label.pack()	
root.mainloop()	

Buttons

- ϖ The Button widget is used to add buttons in a Python application.
- ϖ These buttons can display text or images that convey the purpose of the buttons.

- ϖ A function or a method can be attached to a button which is called automatically when you click the button.
  - ∞ w = Button (master, option=value, ...)
    - δ Options
      - ⇒ activebackground - Background color when the button is under the cursor.
      - ⇒ activeforeground - Foreground color when the button is under the cursor.
      - ⇒ image - Image to be displayed on the button

```
from tkinter import *
from tkinter import messagebox
top = Tk()
def helloCallBack():
  messagebox.showinfo( "Hello Python", "Hello World")
B = Button(top, text ="Hello", activebackground="Red", activeforeground='Yellow', command = helloCallBack)
B.pack()
top.mainloop()
```

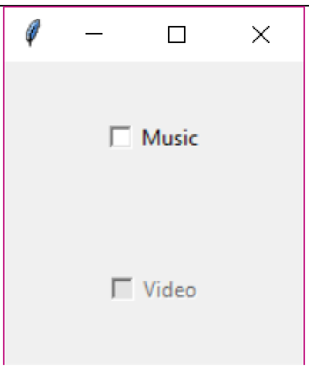


checkbutton

- ϖ The Checkbutton widget is used to display a number of options to a user as toggle buttons.
- ϖ The user can then select one or more options by clicking the button corresponding to each option.
- ϖ Can also display images in place of text.
  - ∞ w = Checkbutton (master, option, ...)
    - δ Options
      - ⇒ command - A procedure to be called every time the user changes the state of this checkbutton.
      - ⇒ offvalue - Normally, a checkbutton's associated control variable will be set to 0 when it is cleared (off).
        - ₪ An alternate value for the off state can be set by setting offvalue to that value.
      - ⇒ onvalue - Normally, a checkbutton's associated control variable will be set to 1 when it is set (on).
        - ₪ An alternate value for the on state can be set by setting onvalue to that value.

⇒ state - The default is state=NORMAL. state=DISABLED to gray out the control and make it unresponsive.

₪ If the cursor is currently over the checkbutton, the state is ACTIVE.

from tkinter import *	
from tkinter import messagebox	
top = 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, \	

canvas

ϖ The Canvas is a rectangular area intended for drawing pictures or other complex layouts.

ϖ Graphics, text, widgets or frames can be placed on a Canvas.

∞ w = Canvas (master, option=value, ...)

δ Options

⇒ confine - If true (the default), the canvas cannot be scrolled outside of the scrollregion.

⇒ height - Size of the canvas in the Y dimension.

⇒ scrollregion - A tuple (w, n, e, s) that defines over how large an area the canvas can be scrolled, where w is the left side, n the top, e the right side, and s the bottom.

⇒ width - Size of the canvas in the X dimension.

⇒ xscrollincrement -Set this option to some positive dimension, the canvas can be positioned only on multiples of that distance, and the value will be used for scrolling by scrolling units, such as when the user clicks on the arrows at the ends of a scrollbar.

⇒ xscrollcommand - If the canvas is scrollable, this attribute should be the .set() method of the horizontal scrollbar.

⇒ yscrollincrement - Works like xscrollincrement, but governs vertical movement.

⇒ yscrollcommand - If the canvas is scrollable, this attribute should be the .set() method of the vertical scrollbar.

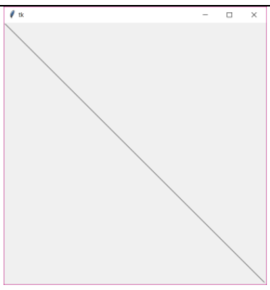
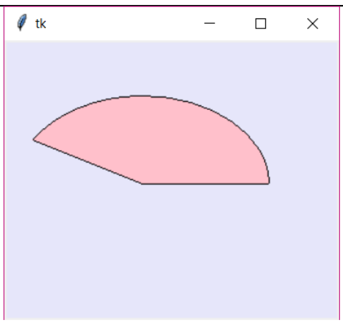
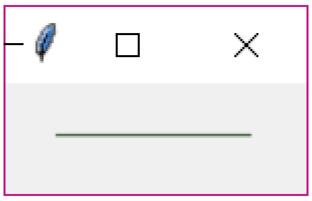
ϖ The Canvas widget can support the following standard items –

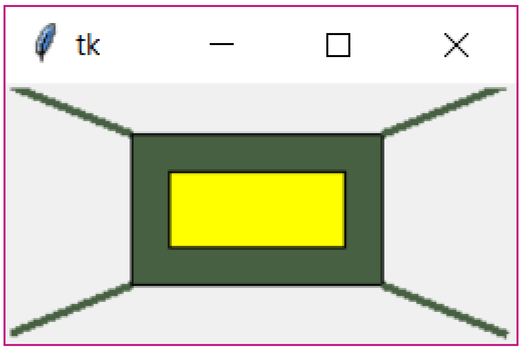
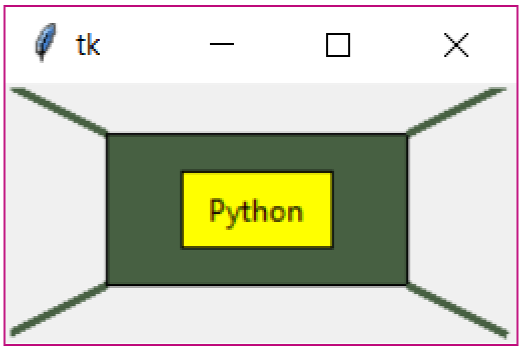
∞ arc − Creates an arc item, which can be a chord, a pieslice or a simple arc.

δ coord = 10, 50, 240, 210

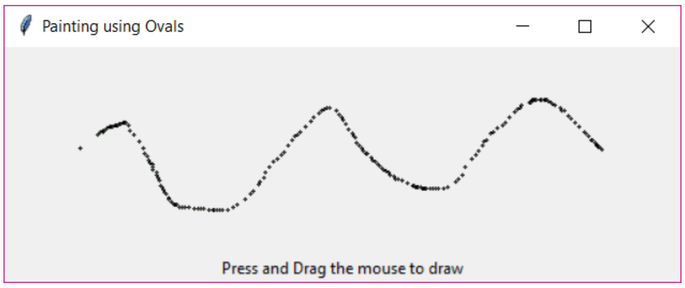
δ arc = canvas.create_arc(coord, start=0, extent=150, fill="blue")

- 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)

from tkinter import *	
tk = Tk()	
canvas = Canvas(tk, width=500, height=500)	
canvas.pack()	
canvas.create_line(0, 0, 500, 500)	
mainloop()	
import tkinter	
top = Tk()	
C = Canvas(top, bg="lavender", height=250, width=300)	
coord = 10, 50, 240, 210	
arc = C.create_arc(coord, start=0, extent=150, fill="pink")	
C.pack()	
top.mainloop()	
from tkinter import *	
master = Tk()	
canvas_width = 80	
canvas_height = 40	
w = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
w.pack()	
y = int(canvas_height / 2)	
w.create_line(0, y, canvas_width, y, fill="#476042")	
mainloop()	

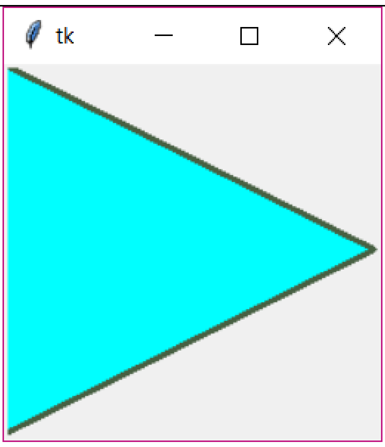
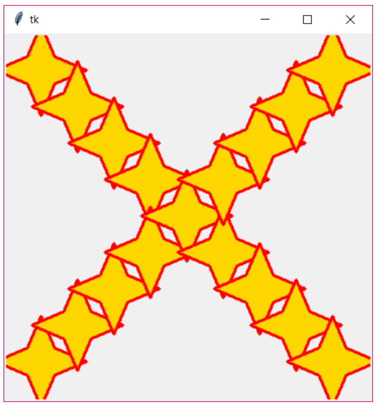
from tkinter import *	
master = Tk()	
w = Canvas(master, width=200, height=100)	
w.pack()	
w.create_rectangle(50, 20, 150, 80, fill="#476042")	
w.create_rectangle(65, 35, 135, 65, fill="yellow")	
w.create_line(0, 0, 50, 20, fill="#476042", width=3)	
w.create_line(0, 100, 50, 80, fill="#476042", width=3)	
w.create_line(150,20, 200, 0, fill="#476042", width=3)	
w.create_line(150, 80, 200, 100, fill="#476042", width=3)	
mainloop()	
from tkinter import *	
canvas_width = 200	
canvas_height = 100	
colours = ("#476042", "yellow")	
box=[]	
for ratio in (0.2, 0.35):	
box.append((canvas_width * ratio,	
canvas_height * ratio,	
canvas_width * (1 - ratio),	
canvas_height * (1 - ratio)))	
master = Tk()	
w = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
w.pack()	
for i in range(2):	
w.create_rectangle(box[i][0], box[i][1],box[i][2],box[i][3], fill=colours[i])	
w.create_line(0, 0, # origin of canvas	
box[0][0], box[0][1], # coordinates of left upper corner of the box[0]	
fill=colours[0],	
width=3)	
w.create_line(0, canvas_height, # lower left corner of canvas	

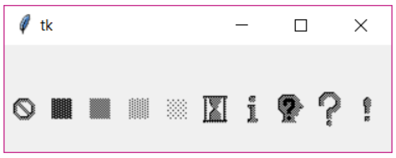
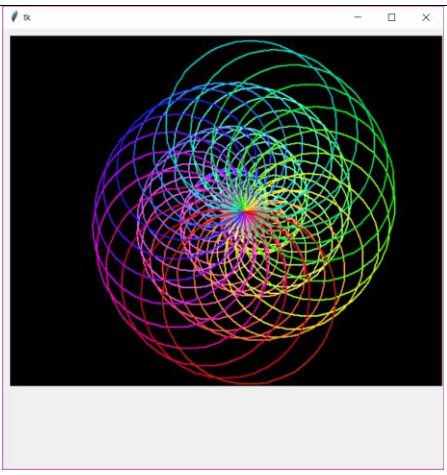
```
        box[0][0], box[0][3], # lower left corner of box[0]
        fill=colours[0],
        width=3)
w.create_line(box[0][2],box[0][1],  # right upper corner of box[0]
        canvas_width, 0,     # right upper corner of canvas
        fill=colours[0],
        width=3)
w.create_line(box[0][2], box[0][3], # lower right corner pf box[0]
        canvas_width, canvas_height, # lower right corner of canvas
        fill=colours[0], width=3)
w.create_text(canvas_width / 2,
        canvas_height / 2,
        text="Python")
mainloop()
```



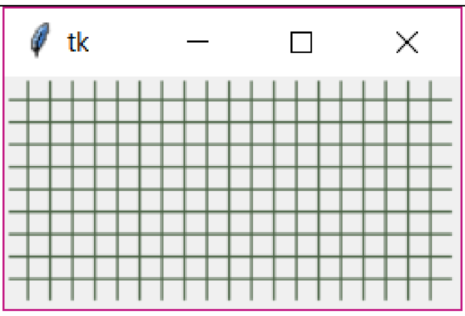
```
from tkinter import *
canvas_width = 500
canvas_height = 150
def paint( event ):
  python_green = "#476042"
  x1, y1 = ( event.x - 1 ), ( event.y - 1 )
  x2, y2 = ( event.x + 1 ), ( event.y + 1 )
  w.create_oval( x1, y1, x2, y2, fill = python_green )
master = Tk()
master.title( "Painting using Ovals" )
w = Canvas(master,
        width=canvas_width,
        height=canvas_height)
w.pack(expand = YES, fill = BOTH)
w.bind( "<B1-Motion>", paint )
message = Label( master, text = "Press and Drag the mouse to draw" )
message.pack( side = BOTTOM )
mainloop()
```

```
from tkinter import *
canvas_width = 200
```

canvas_height =200	 tk
python_green = "#476042"	
master = Tk()	
w = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
w.pack()	
points = [0,0,canvas_width,canvas_height/2, 0, canvas_height]	
w.create_polygon(points, outline=python_green,	
fill='cyan', width=3)	
mainloop()	
from tkinter import *	 tk
canvas_width = 400	
canvas_height =400	
python_green = "#476042"	
def polygon_star(canvas, x,y,p,t, outline=python_green, fill='yellow', width = 1):	
points = []	
for i in (1,-1):	
points.extend((x,	
points.extend((x + i*t, y + i*t))	
points.extend((x + i*p, y))	
points.extend((x + i*t, y - i * t))	
print(points)	
canvas.create_polygon(points, outline=outline,	
fill=fill, width=width)	
master = Tk()	
w = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
w.pack()	
p = 50	
t = 15	
nsteps = 10	
step_x = int(canvas_width / nsteps)	

step_y = int(canvas_height / nsteps)	
for i in range(1, nsteps):	
polygon_star(w,i*step_x,i*step_y,p,t,outline='red',fill='gold', width=3)	
polygon_star(w,i*step_x,canvas_height - i*step_y,p,t,outline='red',fill='gold', width=3)	
mainloop()	
from tkinter import *	
canvas_width = 300	
canvas_height =80	
master = Tk()	
canvas = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
canvas.pack()	
bitmaps = ["error", "gray75", "gray50", "gray25", "gray12", "hourglass", "info", "questhead", "question", "warning"]	
nsteps = len(bitmaps)	
step_x = int(canvas_width / nsteps)	
for i in range(0, nsteps):	
canvas.create_bitmap((i+1)*step_x - step_x/2,50, bitmap=bitmaps[i])	
mainloop()	
from tkinter import *	
canvas_width = 600	
canvas_height =600	
master = Tk()	
canvas = Canvas(master,	
width=canvas_width,	
height=canvas_height)	
canvas.pack()	
img = PhotoImage(file="Circles-3.gif")	
canvas.create_image(10,10, anchor=NW, image=img)	
mainloop()	
from tkinter import *	

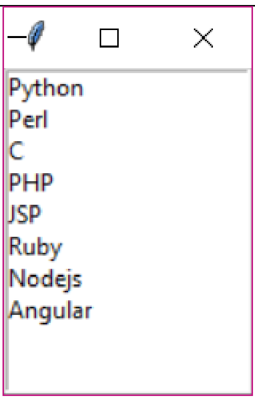
```
def checkered(canvas, line_distance):
  # vertical lines at an interval of "line_distance" pixel
  for x in range(line_distance,canvas_width,line_distance):
    canvas.create_line(x, 0, x, canvas_height, fill="#476042")
  # horizontal lines at an interval of "line_distance" pixel
  for y in range(line_distance,canvas_height,line_distance):
    canvas.create_line(0, y, canvas_width, y, fill="#476042")
master = Tk()
canvas_width = 200
canvas_height = 100
w = Canvas(master,
        width=canvas_width,
        height=canvas_height)
w.pack()
```



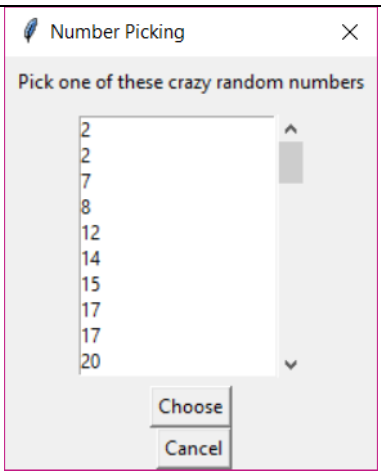
Listbox

- The Listbox widget is used to display a list of items from which a user can select a number of items.
  - w = Listbox (master, option, ...)
    - selectmode - Determines how many items can be selected, and how mouse drags affect the selection
      - ⇒ BROWSE – If an item is clicked and then dragged to a different line, the selection will follow the mouse. This is the default.
      - ⇒ SINGLE – Can only select one line and can't drag the mouse.
      - ⇒ MULTIPLE – Select any number of lines at once. Clicking on any line toggles whether or not it is selected.
      - ⇒ EXTENDED – Select any adjacent group of lines at once by clicking on the first line and dragging to the last line.

```
from tkinter import *
top = Tk()
Lb1 = Listbox(top, selectmode=BROWSE)
Lb1.insert(1, "Python")
Lb1.insert(2, "Perl")
Lb1.insert(3, "C")
Lb1.insert(4, "PHP")
Lb1.insert(5, "JSP")
```



```
Python
Perl
C
PHP
JSP
Ruby
Nodejs
Angular
```

Lb1.insert(6, "Ruby")	
Lb1.insert(7, "Nodejs")	
Lb1.insert(8, "Angular")	
Lb1.pack()	
top.mainloop()	
from tkinter import *	 Number Picking Pick one of these crazy random numbers 2 2 7 8 12 14 15 17 17 20 Choose Cancel
class ListBoxChoice(object):	
def __init__(self, master=None, title=None, message=None, list=[]):	
self.master = master	
self.value = None	
self.list = list[:]	
self.modalPane = Toplevel(self.master)	
self.modalPane.transient(self.master)	
self.modalPane.grab_set()	
self.modalPane.bind("<Return>", self._choose)	
self.modalPane.bind("<Escape>", self._cancel)	
if title:	
self.modalPane.title(title)	
if message:	
Label(self.modalPane, text=message).pack(padx=5, pady=5)	
listFrame = Frame(self.modalPane)	
listFrame.pack(side=TOP, padx=5, pady=5)	
scrollBar = Scrollbar(listFrame)	
scrollBar.pack(side=RIGHT, fill=Y)	
self.listBox = Listbox(listFrame, selectmode=SINGLE)	
self.listBox.pack(side=LEFT, fill=Y)	
scrollBar.config(command=self.listBox.yview)	
self.listBox.config(yscrollcommand=scrollBar.set)	
self.list.sort()	
for item in self.list:	
self.listBox.insert(END, item)	
buttonFrame = Frame(self.modalPane)	

```
        buttonFrame.pack(side=BOTTOM)
        chooseButton = Button(buttonFrame, text="Choose", command=self._choose)
        chooseButton.pack()
        cancelButton = Button(buttonFrame, text="Cancel", command=self._cancel)
        cancelButton.pack(side=RIGHT)
    def _choose(self, event=None):
        try:
            firstIndex = self.listBox.curselection()[0]
            self.value = self.list[int(firstIndex)]
        except IndexError:
            self.value = None
        self.modalPane.destroy()
    def _cancel(self, event=None):
        self.modalPane.destroy()
    def returnValue(self):
        self.master.wait_window(self.modalPane)
        return self.value
if __name__ == '__main__':
    import random
    root = Tk()
    returnValue = True
    list = [random.randint(1,100) for x in range(50)]
    while returnValue:
        returnValue = ListBoxChoice(root, "Number Picking", "Pick one of these crazy random numbers", list).returnValue()
        print(returnValue)
```

Tk MenuBar

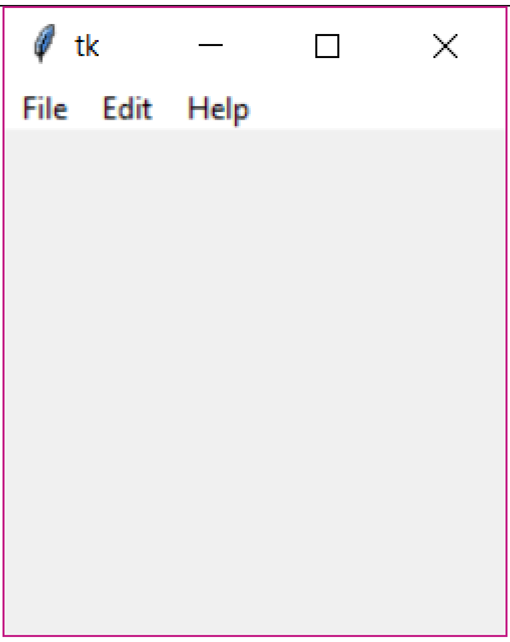
- To create all kinds of menus that can be used by the applications.
- The core functionality provides ways to create three menu types
  - pop-up

- ∞ toplevel
- ∞ pull-down
- ∞ w = Menu (master, option, ...)
  - δ postcommand
    - ⇒ Set this option to a procedure, and that procedure will be called every time someone brings up this menu.
  - δ tearoff
    - ⇒ A menu can be torn off, the first position (position 0) in the list of choices is occupied by the tear-off element, and the additional choices are added starting at position 1.
  - δ title
    - ⇒ The title of a tear-off menu window will be the same as the text of the menubutton or cascade that lead to this menu. To be changed the title of that window, set the title option to that string.

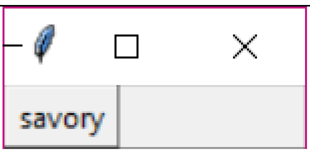
ϖ Tk MenuBar – Methods

- ∞ add_command (options)
  - δ Adds a menu item to the menu.
- ∞ add_radiobutton(options)
  - δ Creates a radio button menu item.
- ∞ add_checkbutton(options)
  - δ Creates a check button menu item.
- ∞ add_cascade(options)
  - δ Creates a new hierarchical menu by associating a given menu to a parent menu
- ∞ add_separator()
  - δ Adds a separator line to the menu.
- ∞ add(type, options)
  - δ Adds a specific type of menu item to the menu.
- ∞ delete(startindex [, endindex])
  - δ Deletes the menu items ranging from startindex to endindex.
- ∞ entryconfig(index, options)
  - δ Allows you to modify a menu item, which is identified by the index, and change its options.
- ∞ index(item)
  - δ Returns the index number of the given menu item label.
- ∞ insert_separator (index)
  - δ Insert a new separator at the position specified by index.
- ∞ invoke (index)
  - δ Calls the command callback associated with the choice at position index. If a checkbutton, its state is toggled between set and cleared; if a radiobutton, that choice is set.
- ∞ type (index)

- δ Returns the type of the choice specified by index: either "cascade", "checkbutton", "command", "radiobutton", "separator", or "tearoff".

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()	
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()	
from tkinter import *	

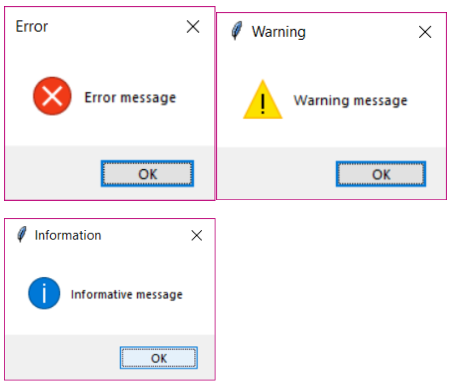
```
import tkinter
top = Tk()
mb = Menubutton ( top, text = "savory", relief = RAISED )
mb.grid()
mb.menu = Menu ( mb, tearoff = 0 )
mb["menu"] = mb.menu
mayoVar = IntVar()
ketchVar = IntVar()
mb.menu.add_checkbutton ( label = "Bajji", variable = mayoVar )
mb.menu.add_checkbutton ( label = "Pakoda", variable = ketchVar )
top.mainloop()
```



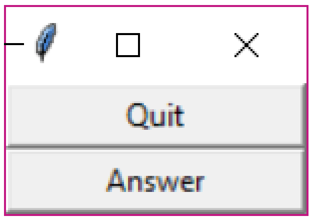
Tk MessageBox

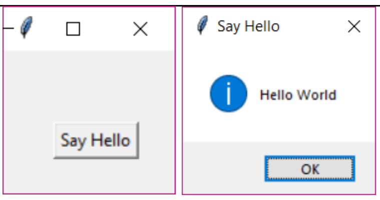
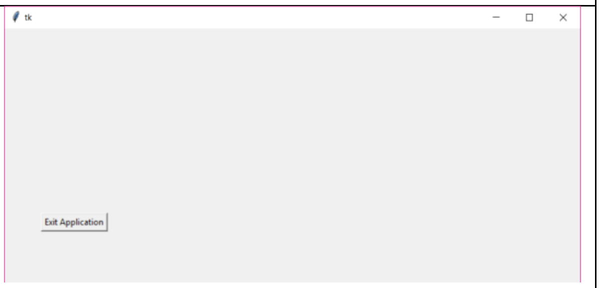
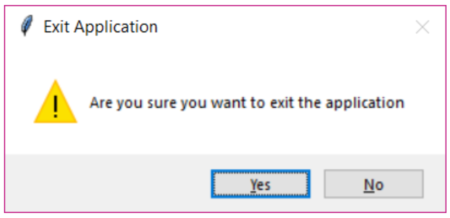
- This widget provides a multiline and noneditable object that displays texts, automatically breaking lines and justifying their contents.
- Its functionality is very similar to the one provided by the Label widget, except that it can also automatically wrap the text, maintaining a given width or aspect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

```
import tkinter
from tkinter import messagebox
root = tkinter.Tk()
root.withdraw()
messagebox.showerror("Error", "Error message")
messagebox.showwarning("Warning","Warning
message")
messagebox.showinfo("Information","Informative
message")
```



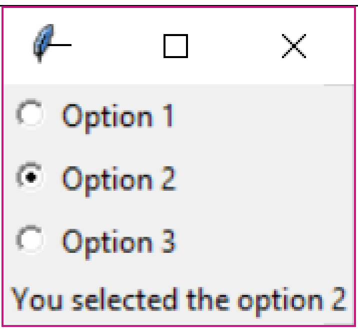
```
import sys
from tkinter import *
from tkinter import messagebox
def answer():
    showerror("Answer", "Sorry, no answer available")
def callback():
    if askyesno('Verify', 'Really quit?'):
        showwarning('Yes', 'Not yet implemented')
```



`    else:`	
`        showinfo('No', 'Quit has been cancelled')`	
`Button(text='Quit', command=callback).pack(fill=X)`	
`Button(text='Answer', command=answer).pack(fill=X)`	
`mainloop()`	
`from tkinter import *`	 Say Hello / Hello World
`from tkinter import messagebox`	
`top = Tk()`	
`top.geometry("100x100")`	
`def hello():`	
`   messagebox.showinfo("Say Hello", "Hello World")`	
`B1 = Button(top, text = "Say Hello", command = hello)`	
`B1.place(x = 35,y = 50)`	
`top.mainloop()`	
`import tkinter as tk`	  Exit Application / Are you sure you want to exit the application
`from tkinter import messagebox`	
`root= tk.Tk() # create window`	
`canvas1 = tk.Canvas(root, width = 800, height = 350)`	
`canvas1.pack()`	
`def ExitApplication():`	
`    MsgBox = tk.messagebox.askquestion ('Exit Application','Are you sure you want to exit the application',icon = 'warning')`	
`    if MsgBox == 'yes':`	
`       root.destroy()`	
`    else:`	
`        tk.messagebox.showinfo('Return','You will now return to the application screen')`	
`button1 = tk.Button (root, text='Exit Application',command=ExitApplication)`	
`canvas1.create_window(97, 270, window=button1)`	
`root.mainloop()`	

radiobutton

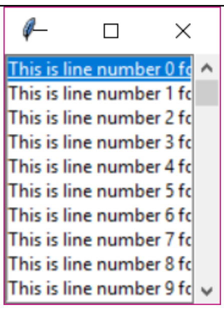
- This widget implements a multiple-choice button, which is a way to offer many possible selections to the user and lets user choose only one of them.
- In order to implement this functionality, each group of radiobuttons must be associated to the same variable and each one of the buttons must symbolize a single value.
- Can use the Tab key to switch from one radionbutton to another.
  - w = Radiobutton (master, option, ...)

`from tkinter import *`	 Option 1 Option 2 Option 3 You selected the option 2
`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()`	

Scroll bar

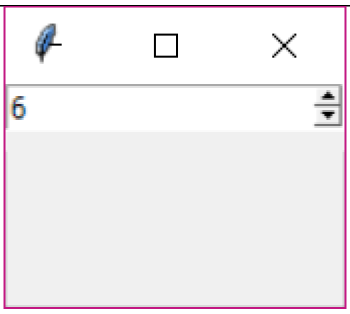
- This widget provides a slide controller that is used to implement vertical scrolled widgets, such as Listbox, Text and Canvas.
  - w = Scrollbar (master, option, ...)

`from tkinter import *`	
`root = Tk()`	
`scrollbar = Scrollbar(root)`	
`scrollbar.pack( side = RIGHT, fill = Y )`	

mylist = Listbox(root, yscrollcommand = scrollbar.set)	 This is line number 0 fo This is line number 1 fo This is line number 2 fo This is line number 3 fo This is line number 4 fo This is line number 5 fo This is line number 6 fo This is line number 7 fo This is line number 8 fo This is line number 9 fo
for line in range(100):	
mylist.insert(END, "This is line number " + str(line)+ " for scrolling")	
mylist.pack(side = LEFT, fill = BOTH)	
scrollbar.config(command = mylist.yview)	
mainloop()	

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.
  - w = Spinbox(master, option, ...)
    - from_
      - ⇒ The minimum value. Used together with to limit the spinbox range.
    - repeatdelay
      - ⇒ Together with repeatinterval, this option controls button auto-repeat. Both values are given in milliseconds.
    - state
      - ⇒ One of NORMAL, DISABLED, or "readonly". Default is NORMAL.
    - values
      - ⇒ A tuple containing valid values for this widget. Overrides from/to/increment.

from tkinter import *	 6
master = Tk()	
w = Spinbox(master, values = (1,2,4,6,8,0,-1,-23))	
#from_ = 0, to = 10)	
w.pack()	
mainloop()	

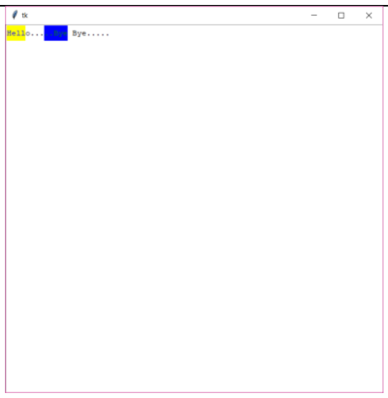
Text

- Text widgets provide advanced capabilities that allow the editing of a multiline text and format the way it has to be displayed, such as changing its color and font.
- Can also use elegant structures like tabs and marks to locate specific sections of the text, and apply changes to those areas.

- ϖ Can embed windows and images in the text because this widget was designed to handle both plain and formatted text.
  - ∞ w = Text (master, option, ...)
    - δ spacing1
      - ⇒ This option specifies how much extra vertical space is put above each line of text.
      - ⇒ If a line wraps, this space is added only before the first line it occupies on the display. Default is 0.
    - δ spacing2
      - ⇒ This option specifies how much extra vertical space to add between displayed lines of text when a logical line wraps. Default is 0.
    - δ spacing3
      - ⇒ This option specifies how much extra vertical space is added below each line of text.
      - ⇒ If a line wraps, this space is added only after the last line it occupies on the display. Default is 0.
- ϖ Methods
  - ∞ delete(startindex [,endindex])
    - δ This method deletes a specific character or a range of text.
  - ∞ get(startindex [,endindex])
    - δ This method returns a specific character or a range of text.
  - ∞ index(index)
    - δ Returns the absolute value of an index based on the given index.
  - ∞ insert(index [,string]...)
    - δ This method inserts strings at the specified index location.
  - ∞ see(index)
    - δ This method returns true if the text located at the index position is visible.
- ϖ Text – Helper Structures – Marks - Methods
  - ∞ Marks are used to bookmark positions between two characters within a given text.
    - δ index(mark)
      - ⇒ Returns the line and column location of a specific mark.
    - δ mark_gravity(mark [,gravity])
      - ⇒ Returns the gravity of the given mark. If the second argument is provided, the gravity is set for the given mark.
    - δ mark_names()
      - ⇒ Returns all marks from the Text widget.
    - δ mark_set(mark, index)
      - ⇒ Informs a new position to the given mark.
    - δ mark_unset(mark)
      - ⇒ Removes the given mark from the Text widget.
- ϖ Text – Helper Structures – Tags - Methods

- ∞ Tags are used to associate names to regions of text which makes easy the task of modifying the display settings of specific text areas.
- ∞ Tags are also used to bind event callbacks to specific ranges of text.
  - δ tag_add(tagname, startindex[,endindex] ...)
    - ⇒ This method tags either the position defined by startindex, or a range delimited by the positions startindex and endindex.
  - δ tag_config
    - ⇒ Use this method to configure the tag properties, which include, justify(center, left, or right), tabs(this property has the same functionality of the Text widget tabs's property), and underline(used to underline the tagged text).
  - δ tag_delete(tagname)
    - ⇒ This method is used to delete and remove a given tag.
  - δ tag_remove(tagname [,startindex[.endindex]] ...)
    - ⇒ After applying this method, the given tag is removed from the provided area without deleting the actual tag definition.

```
from tkinter import *
root = Tk()
text = Text(root, spacing1 = 5, spacing2 = 5, spacing3 = 5)
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 = "blue", foreground = "green")
root.mainloop()
```



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

Scale

- ϖ The Scale widget provides a graphical slider object that allows you to select values from a specific scale.
  - ∞ w = Scale (master, option, ...)
    - δ digits
      - ⇒ The way the program reads the current value shown in a scale widget is through a control variable.
      - ⇒ The control variable for a scale can be an IntVar, a DoubleVar (float), or a StringVar.
      - ⇒ If it is a string variable, the digits option controls how many digits to use when the numeric scale value is converted to a string.

- δ from_
  - ⇒ A float or integer value that defines one end of the scale's range.
- δ label
  - ⇒ Can display a label within the scale widget by setting this option to the label's text.
  - ⇒ The label appears in the top left corner if the scale is horizontal, or the top right corner if vertical. The default is no label.
- δ length
  - ⇒ The length of the scale widget.
  - ⇒ This is the x dimension if the scale is horizontal, or the y dimension if vertical.
  - ⇒ The default is 100 pixels.
- δ orient
  - ⇒ Set orient = HORIZONTAL if the scale should run along the x dimension, or orient = VERTICAL to run parallel to the y-axis. Default is horizontal.
- δ resolution
  - ⇒ Normally, the user will only be able to change the scale in whole units.
  - ⇒ Set this option to some other value to change the smallest increment of the scale's value.
  - ⇒ For example, if from_ = -1.0 and to = 1.0 and resolution = 0.5, the scale will have 5 possible values: -1.0, -0.5, 0.0, +0.5, and +1.0.
- δ showvalue
  - ⇒ Normally, the current value of the scale is displayed in text form by the slider (above it for horizontal scales, to the left for vertical scales).
  - ⇒ Set this option to 0 to suppress that label.
- δ sliderlength
  - ⇒ Normally the slider is 30 pixels along the length of the scale.
  - ⇒ Can change that length by setting the sliderlength option to your desired length.
- δ takefocus
  - ⇒ Normally, the focus will cycle through scale widgets.
  - ⇒ Set this option to 0 if not needed.
- δ tickinterval
  - ⇒ To display periodic scale values, set this option to a number, and ticks will be displayed on multiples of that value.
  - ⇒ For example, if from_ = 0.0, to = 1.0, and tickinterval = 0.25, labels will be displayed along the scale at values 0.0, 0.25, 0.50, 0.75, and 1.00.
  - ⇒ These labels appear below the scale if horizontal, to its left if vertical.
  - ⇒ Default is 0, which suppresses display of ticks.
- δ to
  - ⇒ A float or integer value that defines one end of the scale's range; the other end is defined by the from_ option, discussed above.
  - ⇒ The to value can be either greater than or less than the from_ value.

⇒ For vertical scales, the to value defines the bottom of the scale; for horizontal scales, the right end.

δ troughcolor

⇒ The color of the trough.

δ variable

⇒ The control variable for this scale, if any.

⇒ Control variables may be from class IntVar, DoubleVar (float), or StringVar.

⇒ In the latter case, the numerical value will be converted to a string.

δ width

⇒ The width of the trough part of the widget.

⇒ This is the x dimension for vertical scales and the y dimension if the scale has orient = HORIZONTAL.

⇒ Default is 15 pixels.

∞ Methods

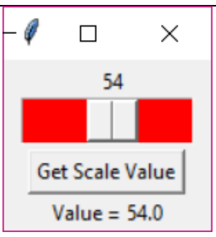
δ get()

⇒ This method returns the current value of the scale.

δ set (value)

⇒ Sets the scale's value.

```
from tkinter import *
def sel():
   selection = "Value = " + str(var.get())
   label.config(text = selection)
root = Tk()
var = DoubleVar()
scale = Scale( root, variable = var, troughcolor = 'red', width = 25, orient = HORIZONTAL)
scale.pack(anchor = CENTER)
button = Button(root, text = "Get Scale Value", command = sel)
button.pack(anchor = CENTER)
label = Label(root)
label.pack()
root.mainloop()
```



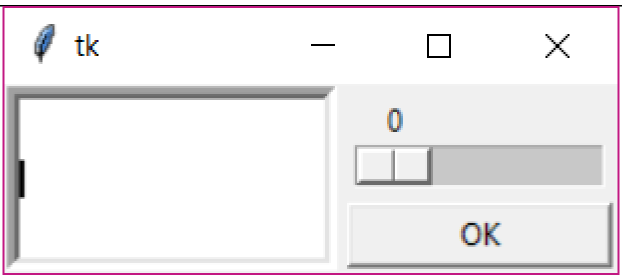
```
Value = 54.0
```

Paned Window

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

ϖ Each pane contains one widget and each pair of panes is separated by a moveable (via mouse movements) sash.

- ϖ Moving a sash causes the widgets on either side of the sash to be resized.
  - ∞ w = PanedWindow(master, option, ...)
    - δ handlepad
      - ⇒ Default is 8.
    - δ handlesize
      - ⇒ Default is 8.
- ϖ PanedWindow objects have these methods –
  - ∞ add(child, options)
    - δ Adds a child window to the paned window.
  - ∞ get(startindex [,endindex])
    - δ This method returns a specific character or a range of text.
  - ∞ config(options)
    - δ Modifies one or more widget options. If no options are given, the method returns a dictionary containing all current option values.

from tkinter import *	
m1 = PanedWindow()	
m1.pack(fill = BOTH, expand = 1)	
left = Entry(m1, bd = 5)	
m1.add(left)	
m2 = PanedWindow(m1, orient = VERTICAL)	
m1.add(m2)	
top = Scale(m2, orient = HORIZONTAL)	
m2.add(top)	
bottom = Button(m2, text = "OK")	
m2.add(bottom)	
mainloop()	

Label Frame

- ϖ A labelframe is a simple container widget.
- ϖ Its primary purpose is to act as a spacer or container for complex window layouts.
- ϖ This widget has the features of a frame plus the ability to display a label.
  - ∞ w = LabelFrame(master, option, ...)
    - δ labelAnchor
      - ⇒ Specifies where to place the label.
      - ⇒ Must be e, en, es, n, ne, nw, s, se, sw, w, wn, or ws
    - δ highlightbackground

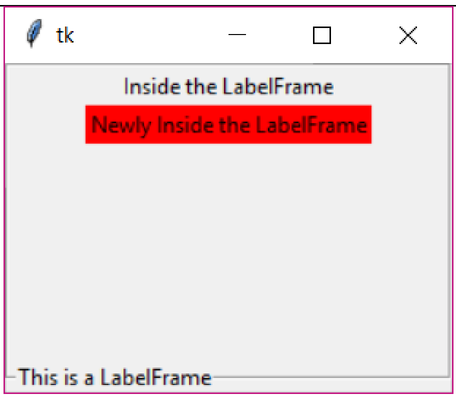
⇒ Color of the focus highlight when the frame does not have focus.

δ highlightcolor

⇒ Color shown in the focus highlight when the frame has the focus.

δ highlightthickness

⇒ Thickness of the focus highlight.

from tkinter import *	 tk Inside the LabelFrame Newly Inside the LabelFrame This is a LabelFrame
root = Tk()	
labelframe = LabelFrame(root, labelanchor = 'sw' , text = "This is a LabelFrame", highlightbackground = 'red', highlightcolor = 'yellow')	
labelframe.pack(fill = "both", expand = "yes")	
left = Label(labelframe, text = "Inside the LabelFrame")	
left.pack()	
right = Label(labelframe, text = "Newly Inside the LabelFrame", bg='red')	
right.pack()	
root.mainloop()	

Toplevel

ϖ Toplevel widgets work as windows that are directly managed by the window manager.

ϖ They do not necessarily have a parent widget on top of them.

ϖ The application can use any number of top-level windows.

∞ w = Toplevel (option, ...)

δ class_

⇒ Normally, text selected within a text widget is exported to be the selection in the window manager.

⇒ Set exportselection = 0 if not needed.

ϖ Toplevel - Methods

∞ deiconify()

δ Displays the window, after using either the iconify or the withdraw methods.

∞ frame()

δ Returns a system-specific window identifier.

∞ group(window)

δ Adds the window to the window group administered by the given window.

∞ iconify()

δ Turns the window into an icon, without destroying it.

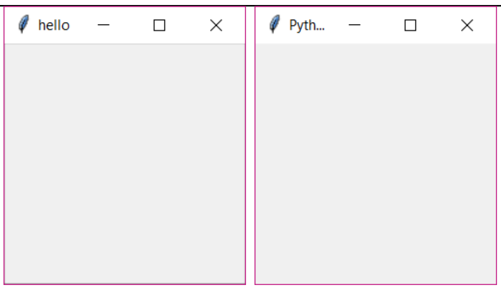
∞ protocol(name, function)

δ Registers a function as a callback which will be called for the given protocol.

∞ state()

- δ Returns the current state of the window. Possible values are normal, iconic, withdrawn and icon.
- ∞ transient([master])
  - δ Turns the window into a temporary(transient) window for the given master or to the window's parent, when no argument is given.
- ∞ withdraw()
  - δ Removes the window from the screen, without destroying it.
- ∞ maxsize(width, height)
  - δ Defines the maximum size for this window.
- ∞ minsize(width, height)
  - δ Defines the minimum size for this window.
- ∞ positionfrom(who)
  - δ Defines the position controller.
- ∞ resizable(width, height)
  - δ Defines the resize flags, which control whether the window can be resized.
- ∞ sizefrom(who)
  - δ Defines the size controller.
- ∞ title(string)
  - δ Defines the window title.

```
from tkinter import *
root = Tk()
root.title("hello")
top = Toplevel()
top.title("Python")
root.iconify()
top.mainloop()
```



Entry

- ϖ The Entry widget is used to accept single-line text strings from a user.
- ϖ For display multiple lines of text that can be edited, then use the Text widget.
- ϖ For display of one or more lines of text that cannot be modified by the user, use the Label widget.
  - ∞ w = Entry(master, option, ...)
    - δ exportselection
      - ⇒ By default, when selecting text within an Entry widget, it is automatically exported to the clipboard. To avoid this exportation, use exportselection = 0.
    - δ justify

⇒ If the text contains multiple lines, this option controls how the text is justified: CENTER, LEFT, or RIGHT.

δ selectbackground

⇒ The background color to use displaying selected text.

δ selectborderwidth

⇒ The width of the border to use around selected text. The default is one pixel.

δ selectforeground

⇒ The foreground (text) color of selected text.

δ 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 = "*".

δ 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. Sets the selection under program control. Selects the text starting at the start index, up to but not including the character at the end index. The start position must be before the end position.

∞ Entry – Methods

δ delete (first, last = None)

⇒ Deletes characters from the widget, starting with the one at index first, up to but not including the character at position last. If the second argument is omitted, only the single character at position first is deleted.

δ get()

⇒ Returns the entry's current text as a string.

δ icursor (index)

⇒ Set the insertion cursor just before the character at the given index.

δ index (index)

⇒ Shift the contents of the entry so that the character at the given index is the leftmost visible character. Has no effect if the text fits entirely within the entry.

δ insert (index, s)

⇒ Inserts string s before the character at the given index.

δ select_adjust (index)

⇒ This method is used to make sure that the selection includes the character at the specified index.

δ select_clear()

⇒ Clears the selection. If there isn't currently a selection, has no effect.

δ select_from (index)

⇒ Sets the ANCHOR index position to the character selected by index, and selects that character.

δ select_present()

⇒ If there is a selection, returns true, else returns false.

δ select_range (start, end)

⇒ Sets the selection under program control. Selects the text starting at the start index, up to but not including the character at the end index. The start position must be before the end position.

δ select_to (index)

⇒ Selects all the text from the ANCHOR position up to but not including the character at the given index.

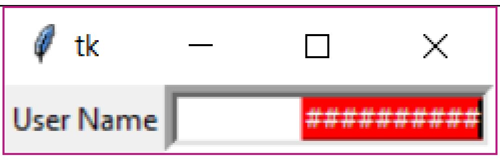
δ xview (index)

⇒ This method is useful in linking the Entry widget to a horizontal scrollbar.

δ xview_scroll (number, what)

⇒ Used to scroll the entry horizontally. The what argument must be either UNITS, to scroll by character widths, or PAGES, to scroll by chunks the size of the entry widget. The number is positive to scroll left to right, negative to scroll right to left.

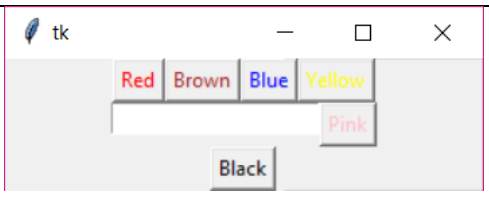
```
from tkinter import *
top = Tk()
L1 = Label(top, text = "User Name")
L1.pack( side = LEFT)
E1 = Entry(top, bd = 5, justify = RIGHT, selectbackground = 'red',
selectborderwidth=2, selectforeground='white', show='#')
E1.pack(side = RIGHT)
top.mainloop()
```



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, ...)

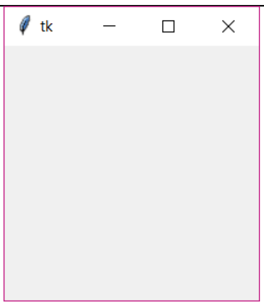
```
from tkinter import *
root = Tk()
frame = Frame(root)
frame.pack()
```

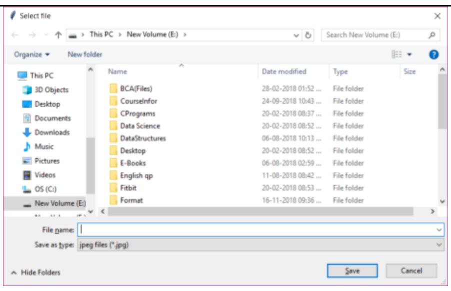
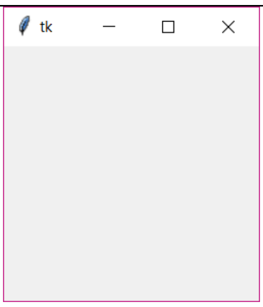
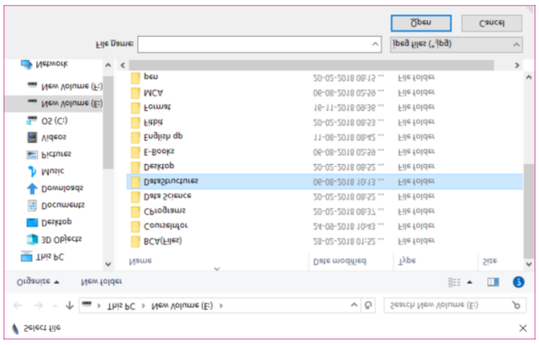
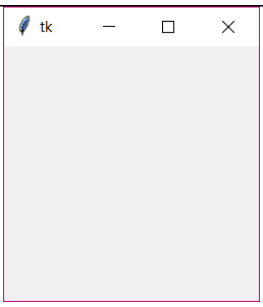
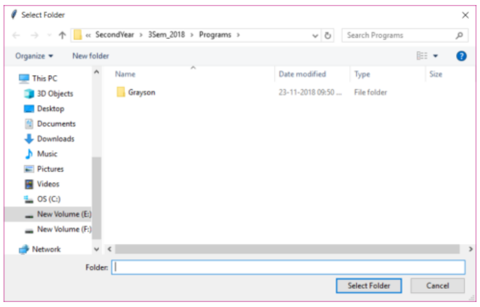


bottomframe = Frame(root)	
bottomframe.pack(side = BOTTOM)	
redbutton = Button(frame, text = "Red", fg = "red")	
redbutton.pack(side = LEFT)	
greenbutton = Button(frame, text = "Brown", fg="brown")	
greenbutton.pack(side = LEFT)	
bluebutton = Button(frame, text = "Blue", fg = "blue")	
bluebutton.pack(side = LEFT)	
yellowbutton = Button(frame, text = "Yellow", fg="yellow")	
yellowbutton.pack(side = RIGHT)	
blackbutton = Button(bottomframe, text = "Black", fg = "black")	
blackbutton.pack(side = BOTTOM)	
pinkbutton = Button(bottomframe, text = "Pink", fg = "pink")	
pinkbutton.pack(side = RIGHT)	
entry = Entry(bottomframe)	
entry.pack()	
root.mainloop()	

Tk File Dialogs

- ϖ tkFileDialog is a module with open and save dialog functions.
  - ∞ .askopenfilename
    - δ Needs Directory, Title, Extension
    - δ Dialog that requests selection of an existing file.
  - ∞ .asksaveasfilename
    - δ Needs Directory, Title, Extension
    - δ Dialog that requests creation or replacement of a file.
  - ∞ .askdirectory
    - δ To open directory

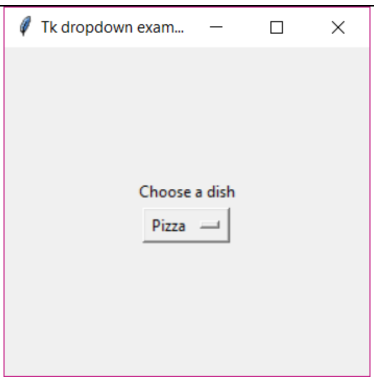
from tkinter import filedialog	
from tkinter import *	
root = Tk()	
root.filename = filedialog.asksaveasfilename(initialdir = "/",title = "Select file", filetypes = (("jpeg files","*.jpg"),("all files","*.*")))	
print (root.filename)	
mainloop()	

	 Select file This PC > New Volume (E:) Search New Volume (E:) Organize New folder Name Date modified Type Size BCA(Files) CourseInfor CPrograms Data Science DataStructures Desktop E-Books English qp Fitbit Format File name: Save as type: jpeg files (*.jpg) Hide Folders Save Cancel
from tkinter import filedialog	
from tkinter import *	
root = Tk()	
root.filename = filedialog.askopenfilename(initialdir = "/",title = "Select file",filetypes = (("jpeg files","*.jpg"),("all files","*.*")))	
print (root.filename)	
root.mainloop()	
from tkinter import *	
root = Tk()	
root.directory = filedialog.askdirectory()	
print (root.directory)	
mainloop()	 Select Folder « SecondYear > 3Sem_2018 > Programs Search Programs Organize New folder Grayson Folder: Select Folder Cancel

Tk DropDown

- Dropdown menus is similar to standard combobox.
- The widget is called OptionMenu and the parameters needed are: frame, tk variable and a dictionary with choices.

```
from tkinter import *
def change_dropdown(*args):
    print( tkvar.get() )
root = Tk()
root.title("Tk dropdown example")
mainframe = Frame(root)
mainframe.grid(column=0,row=0, sticky=(N,W,E,S) )
mainframe.columnconfigure(0, weight = 1)
mainframe.rowconfigure(0, weight = 1)
mainframe.pack(pady = 100, padx = 100)
tkvar = StringVar(root)
choices = { 'Pizza','Lasagne','Fries','Fish','Potatoe'}
tkvar.set('Pizza') # set the default option
popupMenu = OptionMenu(mainframe, tkvar, *choices)
Label(mainframe, text="Choose a dish").grid(row = 1, column = 1)
popupMenu.grid(row = 2, column =1)
tkvar.trace('w', change_dropdown)
mainloop()
```


Tk dropdown exam...
Choose a dish
Pizza