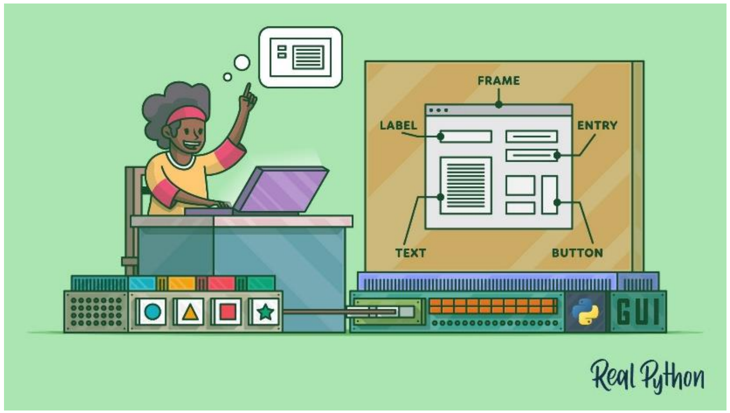
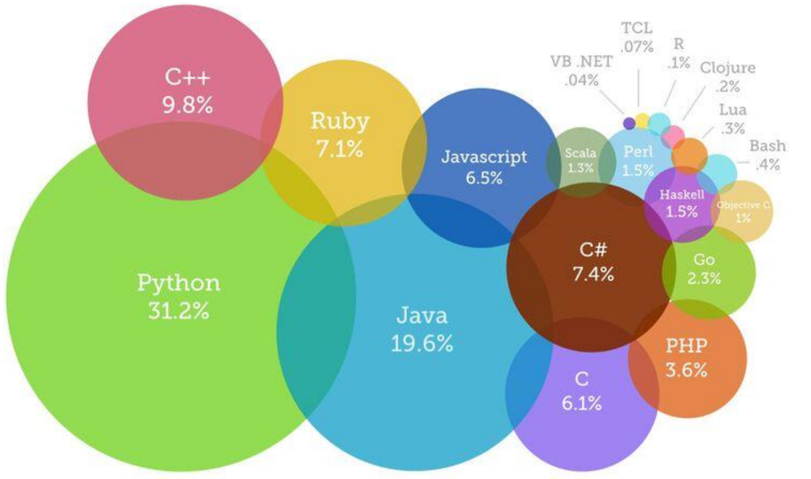


Introduction
to tkinter / Python

https://www.python.org



UNIVERSITÉ TOULOUSE III PAUL SABATIER

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 *`
  - similar to `from math import sin`
    `# Lets you use sin without math.sin`

What can it do?

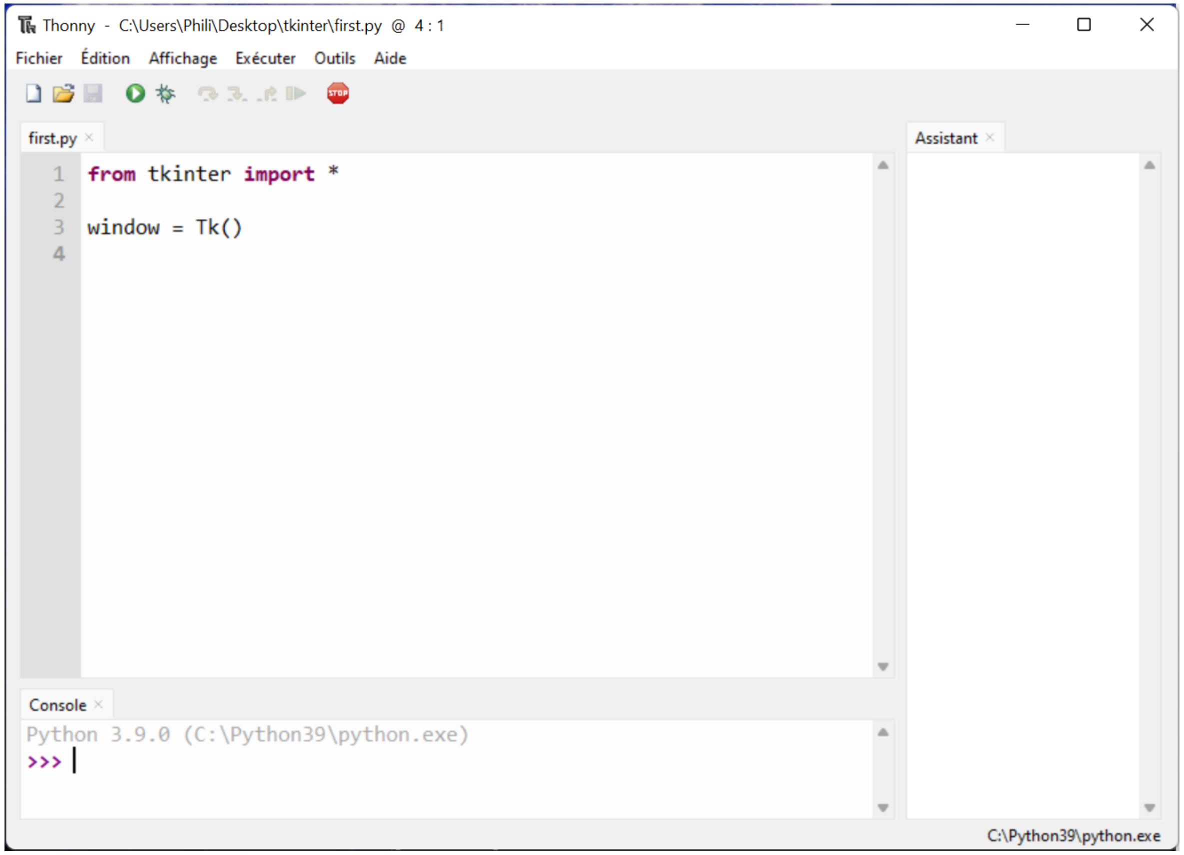
- **tkinter** gives you the ability to create Windows with widgets in them

- *Definition*: a widget is a graphical component on the screen (such as button, text label, drop-down menu, scroll bar, picture, etc…)

- GUIs are built by arranging and combining different widgets on the screen.

Thonny: IDE for dummies

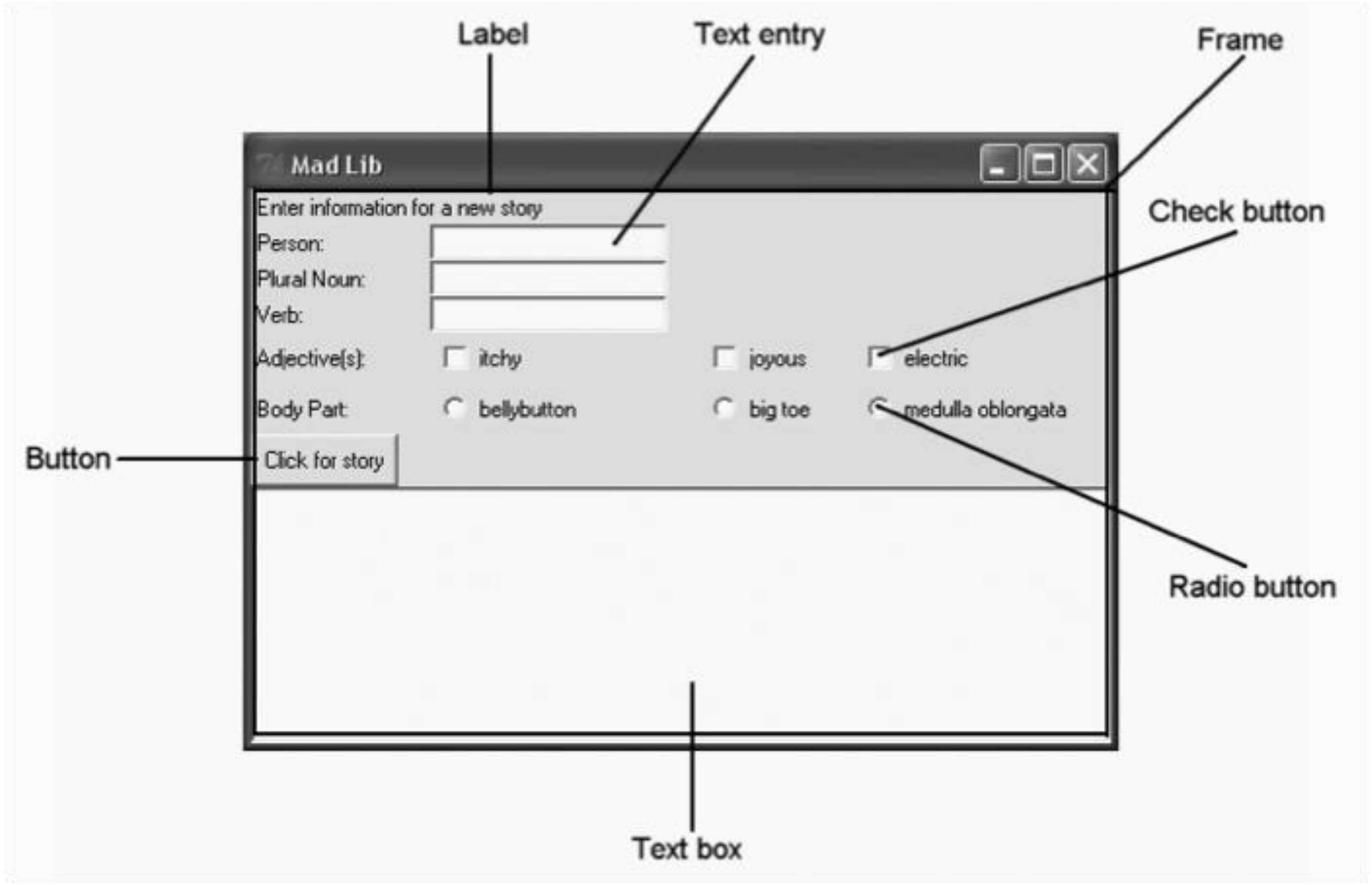
- First, download and install Thonny (https://thonny.org)


Thonny - C:\Users\Phili\Desktop\tkinter\first.py @ 4 : 1
Fichier Édition Affichage Exécuter Outils Aide
first.py
Assistant
Console
C:\Python39\python.exe

```
from tkinter import *

window = Tk()

```

```
Python 3.9.0 (C:\Python39\python.exe)
>>> 
```

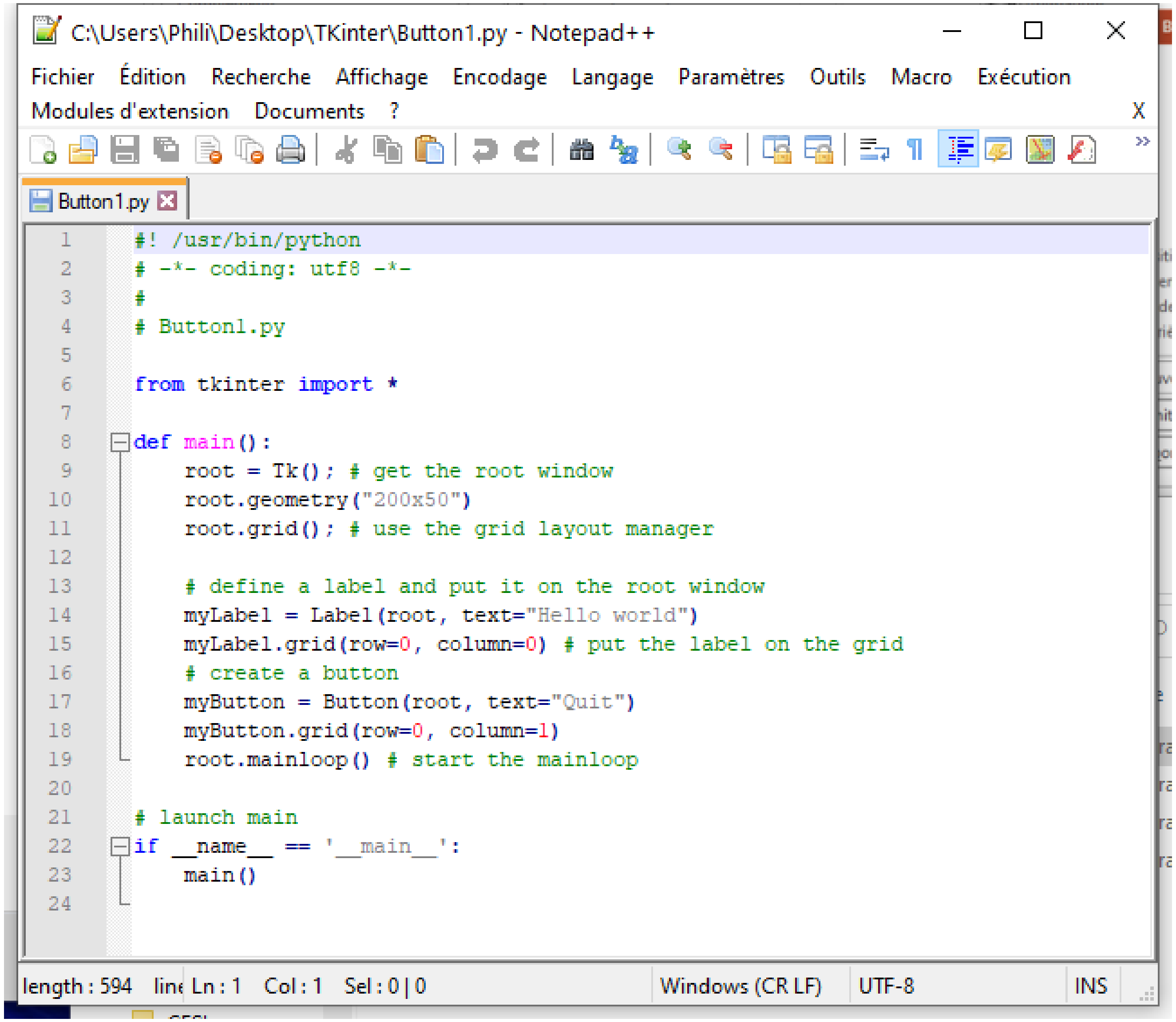
Widgets (GUI Components)



Building a GUI

- **Create widgets**
  - Build the widgets that are going to appear on the screen
- **Layout widgets**
  - Determine where the widgets go on the screen (and how they move when the window is resized, scrolled, etc…)
- **Process Events**
  - Write functions that process events like button presses, mouse clicks, etc…

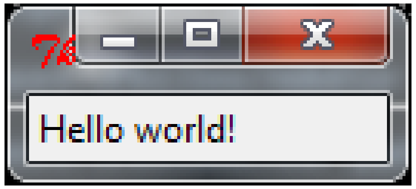
First tkinter Window



C:\Users\Phili\Desktop\TKinter\Button1.py - Notepad++
Fichier Édition Recherche Affichage Encodage Langage Paramètres Outils Macro Exécution
Modules d'extension Documents ?
Button1.py

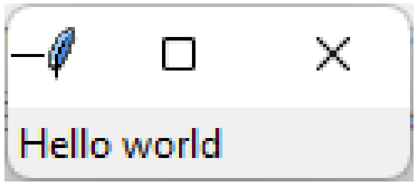
```python
#! /usr/bin/python
# -*- coding: utf8 -*-
#
# Button1.py

from tkinter import *

def main():
    root = Tk(); # get the root window
    root.geometry("200x50")
    root.grid(); # use the grid layout manager

    # define a label and put it on the root window
    myLabel = Label(root, text="Hello world")
    myLabel.grid(row=0, column=0) # put the label on the grid
    # create a button
    myButton = Button(root, text="Quit")
    myButton.grid(row=0, column=1)
    root.mainloop() # start the mainloop

# launch main
if __name__ == '__main__':
    main()

```

length : 594 line Ln : 1 Col : 1 Sel : 0 | 0 Windows (CR LF) UTF-8 INS



Hello world!

Explain the code


Hello world

# 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!

myLabel = 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

myLabel.grid()

Tell the label to place itself into the root window at row=0, col=0. Without calling grid 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!

Entry widget

It’s how to use them to get input from a user. There are three main operations that you can perform with Entry widgets:

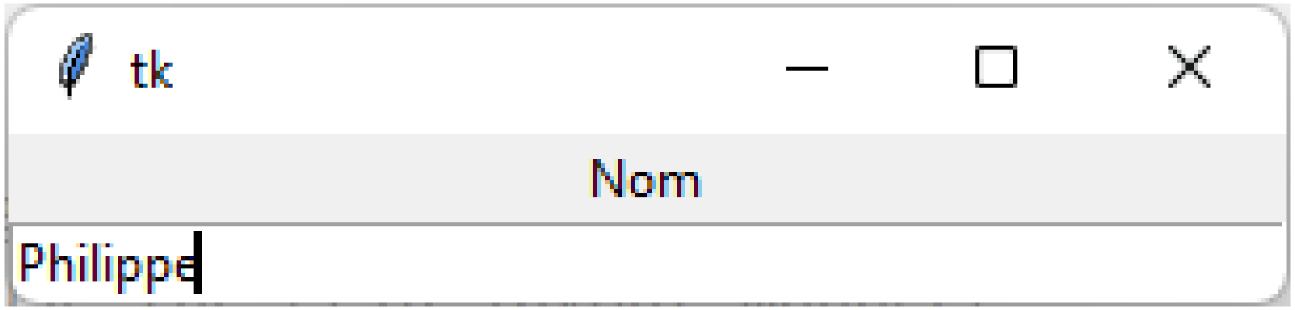
- **Retrieving text** with .get()
- **Deleting text** with .delete()
- **Inserting text** with .insert()

```
entry = Entry(window, width=50);
```

The best way to get an understanding of Entry widgets is to create one and interact with it.

Try it!

- Create your first application composed of a Label and an Entry widgets (you can use colors if you want)



Nom

Philippe

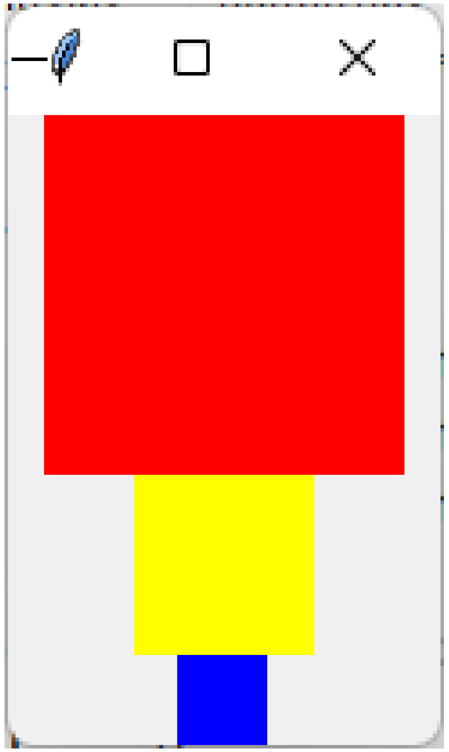
Frame widget

```
f1 = Frame(master=window,
      width=100, height=100, bg='red’)
f1.pack()
```

.pack() places each Frame in the order assigned to the window

Try it!

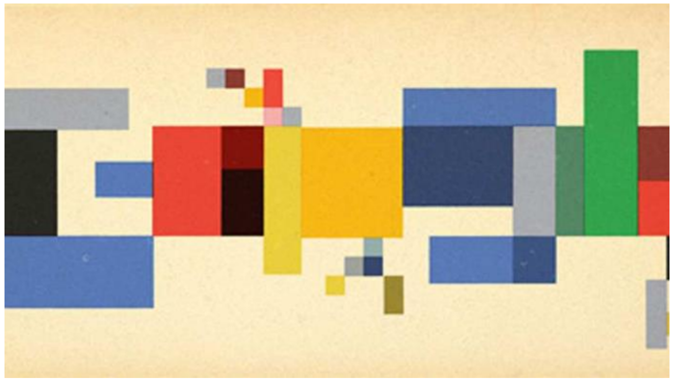
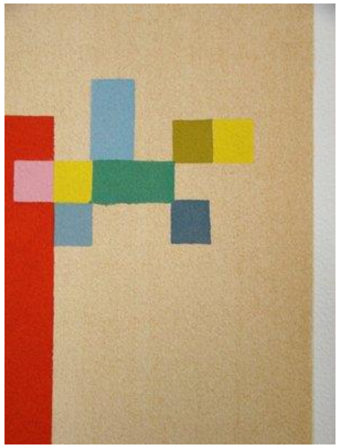
- Create this →



Try it (again)!

.place(x=…, y=…) allows to place widgets where you want on the window

Create a composition with Frames like ones inspired by Sophie Taeuber-Arp

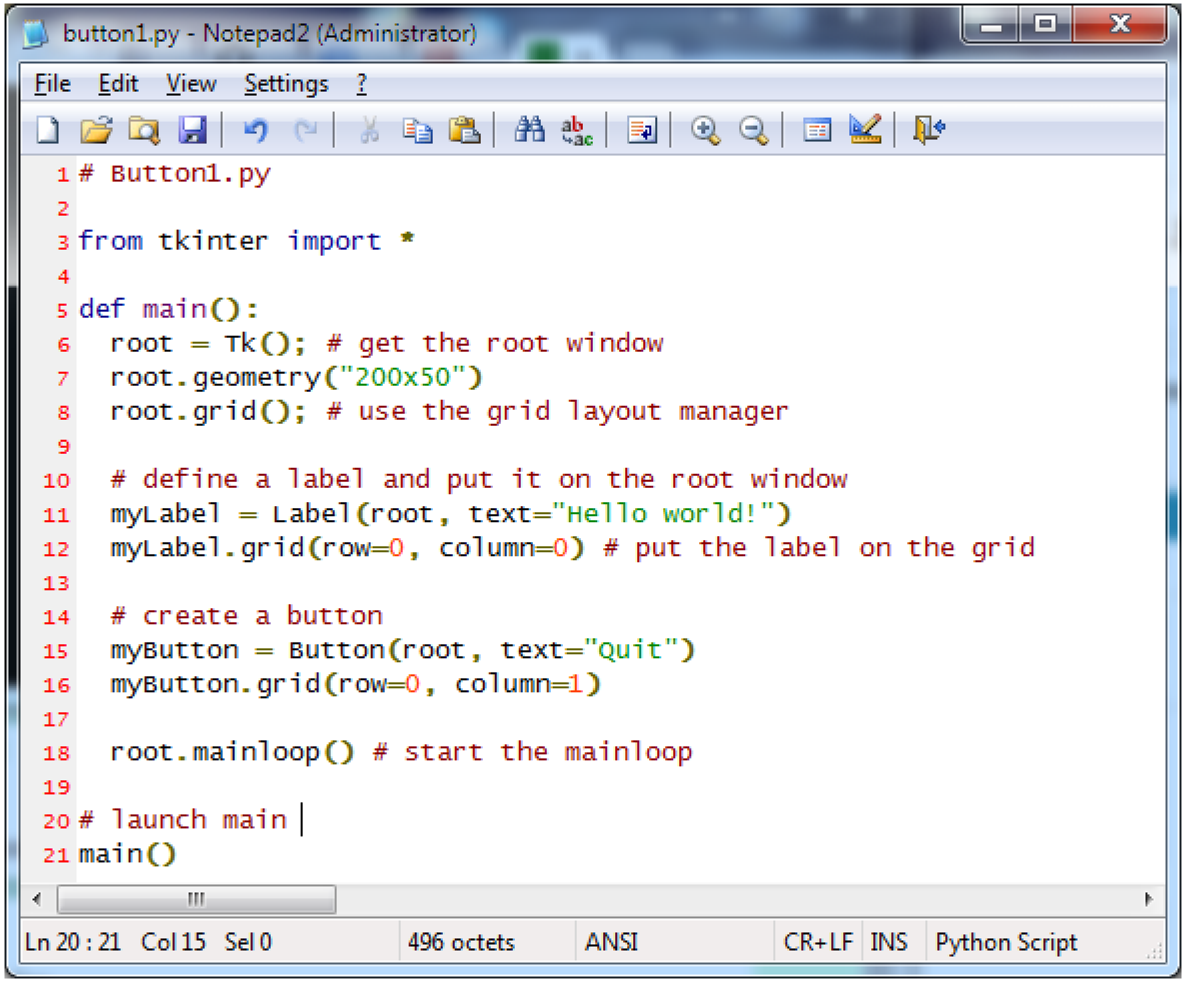



Widgets are objects

- Actually, Widgets are objects.

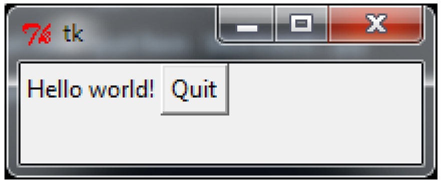
- Classes:
  - Button, Canvas, Checkbutton, Entry, Frame, Label, Listbox,Menu,Menubutton, Message, Radiobutton, Scale, ScrollBar, Text, TopLevel, and many more…

More objects we can build


button1.py - Notepad2 (Administrator)
File Edit View Settings ?

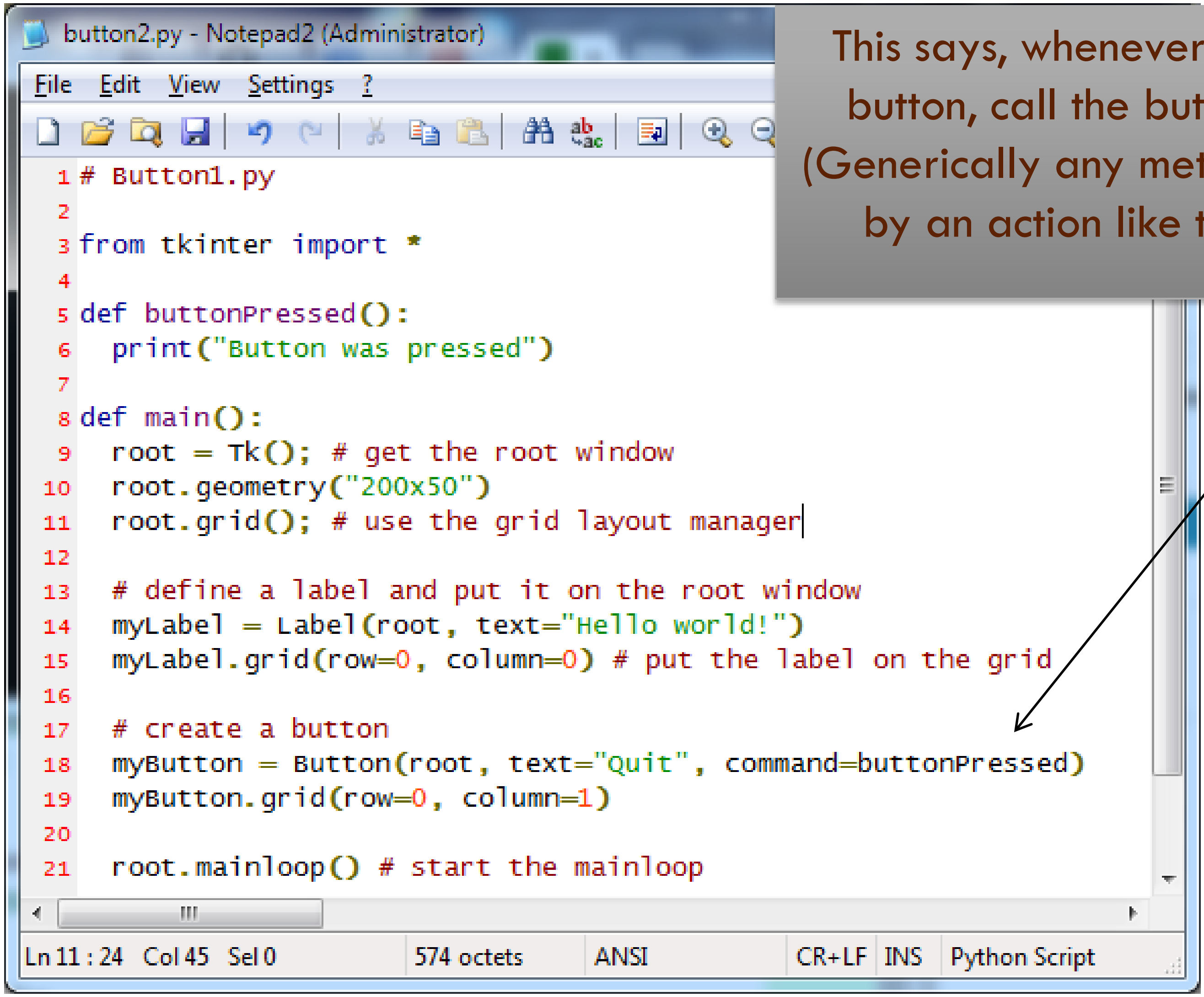
```python
# Button1.py

from tkinter import *

def main():
  root = Tk(); # get the root window
  root.geometry("200x50")
  root.grid(); # use the grid layout manager

  # define a label and put it on the root window
  myLabel = Label(root, text="Hello world!")
  myLabel.grid(row=0, column=0) # put the label on the grid

  # create a button
  myButton = Button(root, text="Quit")
  myButton.grid(row=0, column=1)

  root.mainloop() # start the mainloop

# launch main
main()
```

Ln 20 : 21   Col 15   Sel 0   496 octets   ANSI   CR+LF   INS   Python Script



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

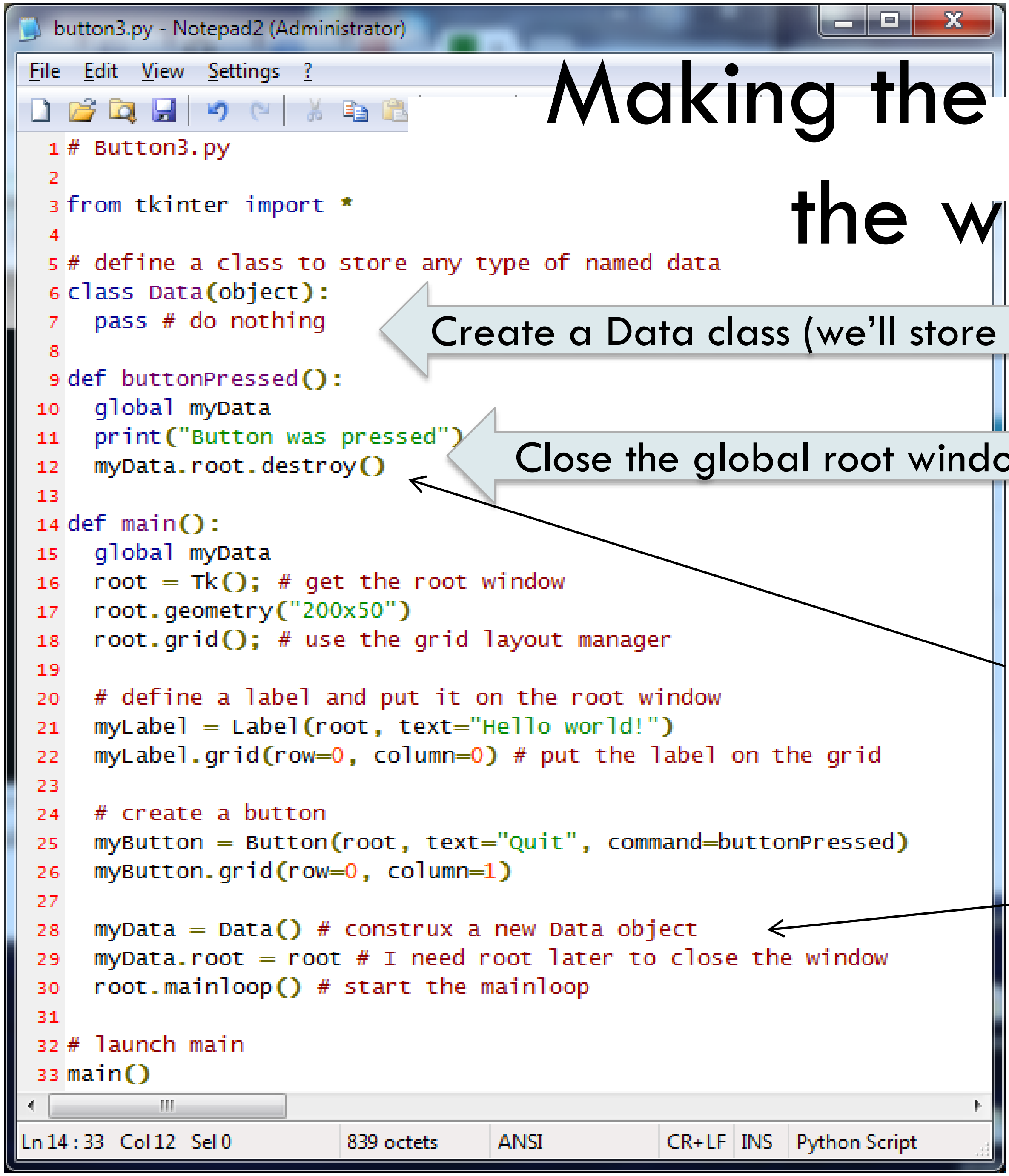
Making the button do something


button2.py - Notepad2 (Administrator)
File Edit View Settings ?
Ln 11 : 24 Col 45 Sel 0 574 octets ANSI CR+LF INS Python Script

```
# Button1.py

from tkinter import *

def buttonPressed():
  print("Button was pressed")

def main():
  root = Tk(); # get the root window
  root.geometry("200x50")
  root.grid(); # use the grid layout manager

  # define a label and put it on the root window
  myLabel = Label(root, text="Hello world!")
  myLabel.grid(row=0, column=0) # put the label on the grid

  # create a button
  myButton = Button(root, text="Quit", command=buttonPressed)
  myButton.grid(row=0, column=1)

  root.mainloop() # start the mainloop
```

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

Making the button close the window



button3.py - Notepad2 (Administrator)
File Edit View Settings ?

```
# Button3.py

from tkinter import *

# define a class to store any type of named data
class Data(object):
  pass # do nothing

def buttonPressed():
  global myData
  print("Button was pressed")
  myData.root.destroy()

def main():
  global myData
  root = Tk(); # get the root window
  root.geometry("200x50")
  root.grid(); # use the grid layout manager

  # define a label and put it on the root window
  myLabel = Label(root, text="Hello world!")
  myLabel.grid(row=0, column=0) # put the label on the grid

  # create a button
  myButton = Button(root, text="Quit", command=buttonPressed)
  myButton.grid(row=0, column=1)

  myData = Data() # construx a new Data object
  myData.root = root # I need root later to close the window
  root.mainloop() # start the mainloop

# launch main
main()
```

Create a Data class (we'll store all our global data in here)

Close the global root window

Calling this also ends the mainloop() function (and thus ends your program)

Create a Data object (make it global). Add attributes to it for later use.

UNIVERSITÉ TOULOUSE III PAUL SABATIER

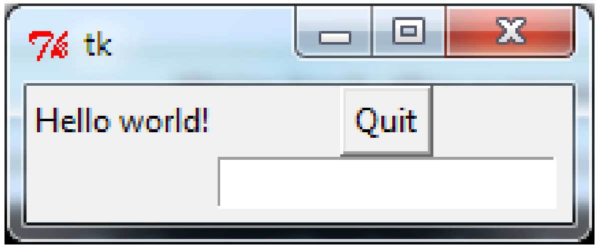
Creating text entry box

General form for all widgets:

1. # Create the widget
   widget = <widgetname>(parent, attributes…)

2. # place the widget to make it show up
   widget.grid(row=someNum, column=somNum)

```
def createTextBox(parent):
  myBox = Entry(parent)
  myBox.grid(row=2, column=1)

 # create an Entry
 createTextBox(root)
```



Using a text entry box

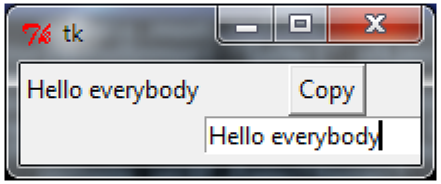
To use a text entry box you must be able to get information from it when you need it. (Generally in response to an event)

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

Using a text entry box

- Create it as an attribute
- Use "get" method when needed to get the text inside the box

```
# define a label and put it on the root window
v= StringVar()
myLabel = Label(root, textvariable=v)
myLabel.grid(row=0, column=0) # put the label on the grid
v.set("Hello World")
```



Hello everybody

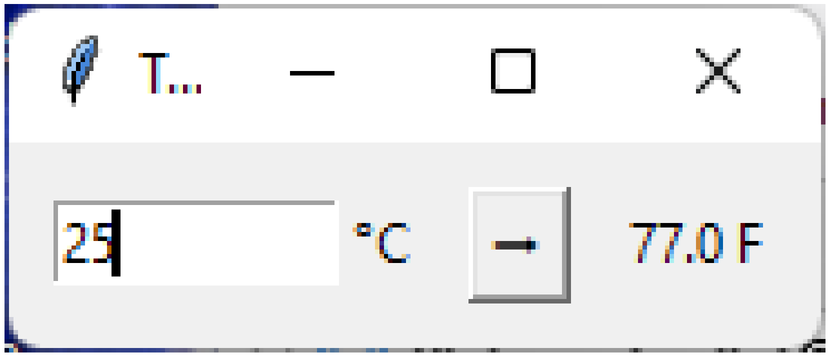
Hello everybody

```
myData = Data() # construct a new Data object
myData.root = root # I need root later to close the window
myData.root.myBox = myBox
myData.root.v = v
```

```
def buttonPressed():
  global myData
  print("Button was pressed")
  s = myData.root.myBox.get()
  myData.root.v.set(s)
```

Try it!

- Create a Temperature converter application (°C to °F). You will need:
  - An Entry widget
  - A Label widget
  - A Button widget
  - And a formula ☺
    - Fahrenheit = (Celcius * 9/5) + 32



Text widgets

- Use Text widgets to have a user entry multiple lines of text
- aWidget = Text(root, width=40, height=5, borderwidth=3)
  Create a text entry box
  40 chars wide, 5 lines tall, with a border that is 3 pixels wide

- aWidget.config(relief = RAISED) # How does the border look
  SUNKEN, RAISED, GROOVE, RIDGE, or FLAT

- aWidget.delete(1.0, END) # Clear the box
- aWidget.insert(END, someString) # Insert string into the box

Add scrollbar to text widget

Scrollbars appear if the text in the box needs more space.

1. Create the Text box
   ```
   tBox = Text(root, width=40, height=5)
   ```

2. Put it on the grid
   ```
   tBox.grid(row=0, column=0, columnspan=5)
   ```

3. Create the scroll bar
   ```
   scrollBarY = Scrollbar ( root, orient=VERTICAL, command=tBox.yview
   )
   ```

4. Put it on the grid, stick it to top and bottom
   ```
   scrollBarY.grid ( row=0, column=6,sticky=N+S )
   ```

5. Link the text box's yscrollcommand to the scrollbar's set function:
   ```
   tBox.config(yscrollcommand =  scrollBarY.set)
   ```

UNIVERSITÉ TOULOUSE III PAUL SABATIER

Try It

- Create a GUI with a button and scrollable text area.
- Type in some text until the scrollbars appear
- Make the button clear the text area when clicked.

General Methods (for many widgets)

- w.config(bg="red") # Modify a config (or optional) value in the widget. w is the widget.

- w.config(state=DISABLED) # "gray out" a widget
- w.config(state=NORMAL) # Go back to normal

Layout management

- You may have noticed we always call “grid”. If not, the widgets will not show up!
- Grid is a layout or geometry manager. It is responsible for determining where widgets go in a window, and what happens as the window changes size, widgets get added, removed, etc…
- Most windowing toolkits have layout management systems to help you arrange widgets.

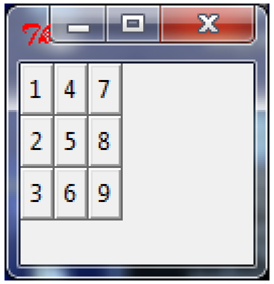
Grid parameters

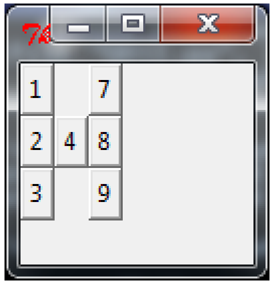
- **row, column –** specify the row and column location of each widget.
  - 0,0 is the upper left corner
  - Empty rows are discarded (they do NOT make blank space)
- **rowspan, columnspan –** specify how many rows or columns a single widget should take
- **padx, pady –** specify how much blank space should be put around the widget

Grid parameters

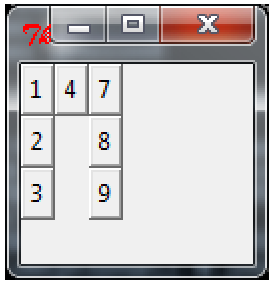
- **sticky** - Defines how to expand the widget if the resulting cell is larger than the widget itself. This can be any combination of the constants S, N, E, and W, or NW, NE, SW, and SE.
- For example, W (west) means that the widget should be aligned to the left cell border. W+E means that the widget should be stretched horizontally to fill the whole cell. W+E+N+S means that the widget should be expanded in both directions.
- Default is to center the widget in the cell.

Examples

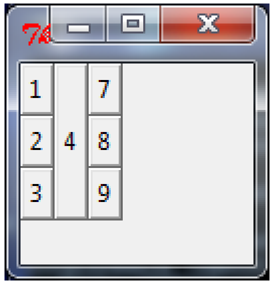




```
b4 = Button(root, text="4")
b4.grid(row=0, column=1, rowspan=3)
```

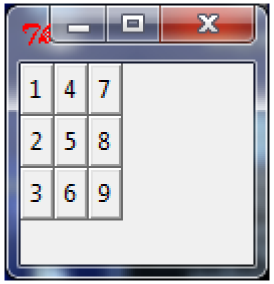


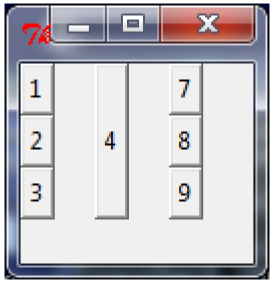
```
b4 = Button(root, text="4")
b4.grid(row=0, column=1, rowspan=3, sticky=N)
```



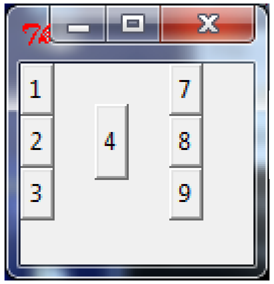
```
b4 = Button(root, text="4")
b4.grid(row=0, column=1, rowspan=3, sticky=N+S)
```

Examples





```
b4 = Button(root, text="4")
b4.grid(row=0, column=1, rowspan=3, sticky=N+S, padx=20)
```



```
b4 = Button(root, text="4")
b4.grid(row=0, column=1, rowspan=3, sticky=N+S, padx=20, pady=20)
```

Other geometry managers

Python has other geometry managers (instead of grid)

to create any GUI layout you want

- **pack –** lets you put items next to each other in different ways, allowing for expansion

- **grid –** lets you specify a row,column grid location and how many rows and columns each widget should span

- **place –** specify an exact pixel location of each widget

WARNING: Never use multiple geometry managers in one window! They are not compatible with each other and may cause infinite loops in your program!!

Showing Images

An image is just another widget.

photo = PhotoImage(file=‘somefile.gif’)

Note: tkinter only supports GIF, PGM, PBM, to read JPGs you need to use the Python Imaging Library and ImageTk (Python 2,x only!)

```
im = PhotoImage(file=‘beer.gif') # Create the PhotoImage widget

# Add the photo to a label:
w = Label(root, image=im) # Create a label with image
w.image = im # Always keep a reference to avoid garbage collection (memory cleanup)
w.grid() # Put the label into the window
```

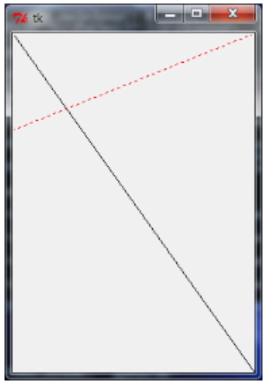
When your image disappears randomly… read this!

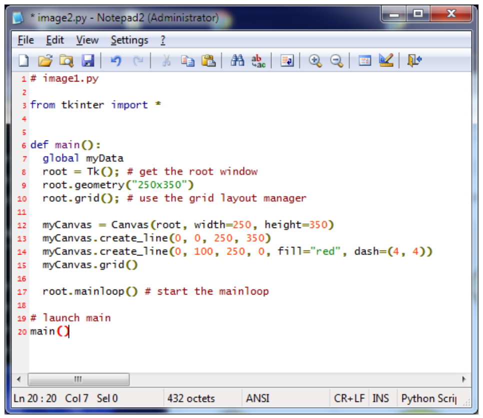
Guess how you put an image in a Button?

Showing Images

A Canvas is a container that allows you to show images and draw on the container. Draw graphs, charts, implement custom widgets (by drawing on them and then handling mouse-clicks).

```
myCanvas = Canvas(root, width=400, height=200)
myCanvas.create_line(0, 0, 200, 100)
myCanvas.create_line(0, 100, 200, 0, fill="red", dash=(4, 4))
myCanvas.create_image(0, 0, anchor=NW, image=myPhotoImage)
```




* image2.py - Notepad2 (Administrator)
File Edit View Settings ?

```
# image1.py

from tkinter import *


def main():
  global myData
  root = Tk(); # get the root window
  root.geometry("250x350")
  root.grid(); # use the grid layout manager

  myCanvas = Canvas(root, width=250, height=350)
  myCanvas.create_line(0, 0, 250, 350)
  myCanvas.create_line(0, 100, 250, 0, fill="red", dash=(4, 4))
  myCanvas.grid()

  root.mainloop() # start the mainloop

# launch main
main()
```

Ln 20 : 20 Col 7 Sel 0 | 432 octets | ANSI | CR+LF | INS | Python Scri

How can we change the background color of a canvas?

Adding a title to your window

- This is actually very simple. You simply call the title method of the root window:

```
root.title("This is my window title")
```

Message Dialog Boxes

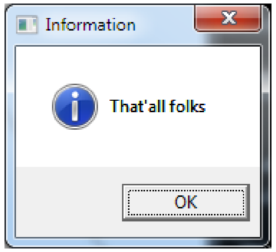
- A dialog box is a small modal window that appears on top of the main window
  - used to ask a question, show a message or do many other things
  - File->Open usually opens a dialog box
  - **Definition:** A modal window is one that temporarily stops all other GUI processing (events)

- You may notice that in many programs the dialog box to open a file is very similar, or the dialog box to select a file or choose a color. These are very standard things, and most GUI toolkits (including Tk) provide support to make these tasks easy.

Message Dialog Boxes

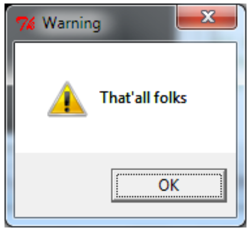
- Using tkinter to create a dialog box you do this code:

```
from tkinter.messagebox import *   # Another way you can import

showinfo(title=“Information”,
    message=“That’s all Folks”)
```

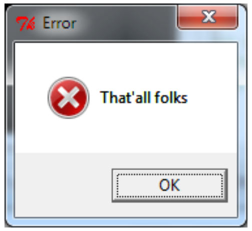


That'all folks

- You can also call showwarning, showerror the only difference will be the icon shown in the window.



That'all folks



That'all folks

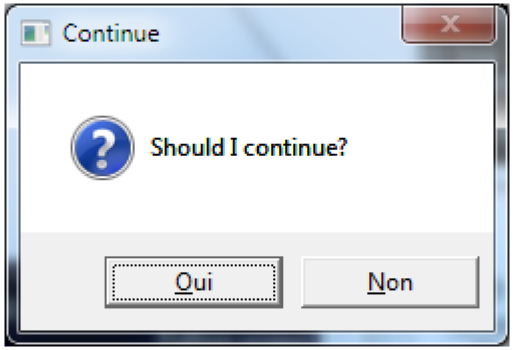
Question Dialog Boxes

Question dialogs are also available

from tkinter.messagebox import *

ans = askyesno("Continue", "Should I continue?")

ans will be True (for Yes) or False (for No).
What do you do with answer then?



Other questions available are: askokcancel, askretrycancel, askquestion

Warning: askquestion by itself will return “yes” or “no” as strings, NOT True and False!

File Dialog Boxes

- See the link for some examples of standard dialogs to
  - open a file
  - select a directory
  - selecting a file to save

- from tkinter.filedialog import *
  - methods: askdirectory, askopenfile, askopenfilename, askopenfilenames, askopenfiles, asksaveasfile, asksaveasfilename

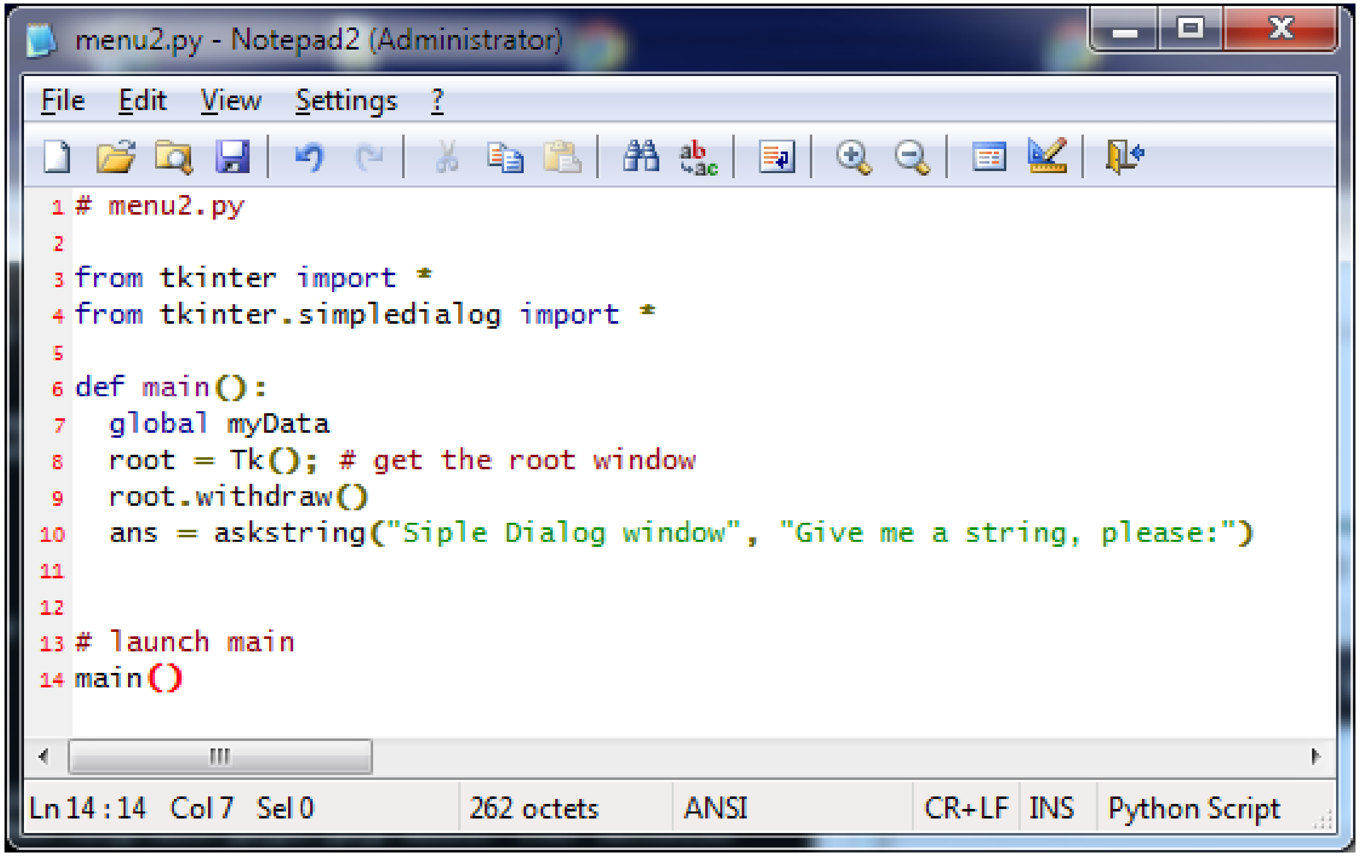
Data Input Dialogs

- You can also use tkinter.simpledialog to ask for a number or string using a dialog box:

  askstring(title, prompt), askinteger…, askfloat...

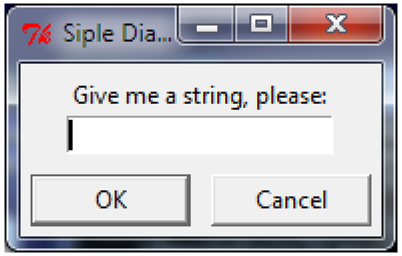
```
from tkinter.simpledialog import *
ans = askstring("Title", "Give me your name")
print ans
ans = askinteger(”Dialog Title", "Give me an integer")
print ans
ans = askinteger(”Num", "Give me an integer between 0 and 100",
    minvalue=0, maxvalue=100)
print ans
```

Data Input Dialog Example


menu2.py - Notepad2 (Administrator)
File Edit View Settings ?

```
# menu2.py

from tkinter import *
from tkinter.simpledialog import *

def main():
  global myData
  root = Tk(); # get the root window
  root.withdraw()
  ans = askstring("Siple Dialog window", "Give me a string, please:")


# launch main
main()
```

Ln 14 : 14   Col 7   Sel 0   262 octets   ANSI   CR+LF   INS   Python Script



Try it!

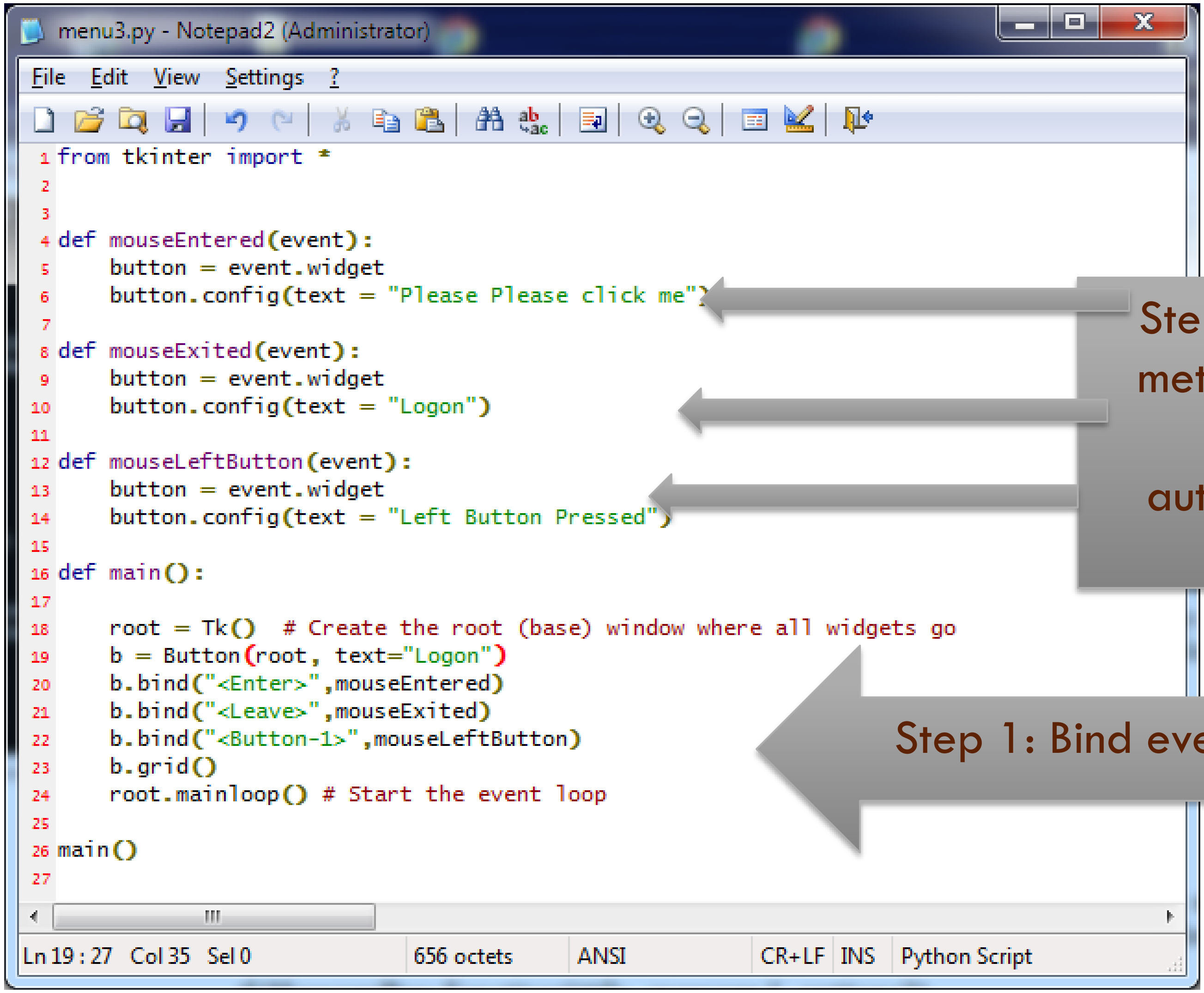
- Create a number guessing game using dialogs to ask for the a number, and then say “higher”, “lower” and ask for another number. Show a message dialog when they finally get the correct answer.

Capturing mouse-clicks

- To capture mouse events you can “bind” events to a widget.
  - widget.bind(event, handler)
    - Event is what the user did
    - Handler is the function to call when the user does the event
  - events can be:
    - <Button-1>
      - (1 is left mouse button, 2=right, 3=middle)
    - <Double-Button-1> - double clicked button 1
    - <Enter> - mouse entered the widget
    - <Leave> - mouse left the widget
    - <Return> - user pressed enter key
    - <key> (<a> for example) – user pressed “a”

Capturing mouse-clicks

For example, to make a button beg to be clicked:


menu3.py - Notepad2 (Administrator)
File Edit View Settings ?

```
from tkinter import *


def mouseEntered(event):
    button = event.widget
    button.config(text = "Please Please click me")

def mouseExited(event):
    button = event.widget
    button.config(text = "Logon")

def mouseLeftButton(event):
    button = event.widget
    button.config(text = "Left Button Pressed")

def main():

    root = Tk()  # Create the root (base) window where all widgets go
    b = Button(root, text="Logon")
    b.bind("<Enter>",mouseEntered)
    b.bind("<Leave>",mouseExited)
    b.bind("<Button-1>",mouseLeftButton)
    b.grid()
    root.mainloop() # Start the event loop

main()
```

Step 2: Write functions (or methods) to handle events. Notice: event object automatically passed into event handler!

Step 1: Bind events to functions

General Design Strategy

- Design the GUI – Layout what widgets you want, and where they should go

- Code the GUI

- Tell the system what events you want to know about
  - associate events with the appropriate event handlers (typically called **binding** or **registering** an event listener)

- Tell the system to begin accepting events
  - root.mainloop()

Capturing mouse-clicks

```
def mouseEntered(event):
    button = event.widget
    button.config(text = "Please Please click me")
```

Notice how I say “event.widget”… that is because all events store as data the widget that caused the event. In this case it is a button. (This again is because event is an object of class Event. That object stores data items – one of which is named “widget”.

Note: if you need to bind left-button mouse events to the canvas and then look at the x,y location of the click. Is x,y stored in the event? Check the link below to see the names to everything you can get from an event object just by saying:

```
myVariable = event.attribute
```

Common problem!

```
def main():
    global root
    root = Tk()  # Create the root (base) window where all widgets go
    b = Button(root, text="Logon")
    b.bind("<Enter>",mouseEntered)
    b.bind("<Leave>",mouseExited)
    b.pack()
    root.mainloop() # Start the event loop

main()
```

WARNING: When you specify a callback, you must NOT use parenthesis… using parenthesis CALLS the function once.. you want to pass the function as a parameter!

```
b.bind("<Enter>", mouseEntered)    # GOOD

b.bind("<Enter>", mouseEntered()) # BAD!
```

Try it!

- Create a window
- Add a canvas to it (set the background color to green so you can see it)
- Bind left click <Button-1> to the canvas..
- Print out the location of the click when clicked:
  - print(event.x, event.y)
- We'll use this later.

How mouse-clicks work: the event loop

- In this GUI we are using event based programming."**root.mainloop()**" starts an event loop in Python that looks like this:

  -
    ```
    while (True): # Loop forever
      wait for an event
      handle the event (usually call an event
        handler with the event information object)
    ```

- Many events you never see (window resized, iconified, hidden by another window and reshown…) You can capture these events if desired, but tkinter handles them for you and generally does what you want.

Event Driven Programming

Event driven programming – a programming paradigm where the flow of the program is driven by sensor outputs or user actions (aka events)

– Wikipedia

Batch programming – programming paradigm where the flow of events is determined completely by the programmer

– Wikipedia

BATCH
Get answer for question 1
Get answer for question 2
Etc…

EVENT-BASED
User clicked "answer q1 button"
User clicked "answer q3 button"
User clicked "answer q2 button"
Etc…

General Event Model

Event

- An object generated as a result of a specific interaction with the system
- Examples: button press, mouse click, typing text in a field and pressing <ENTER>, etc.

Event Listener

- A mechanism that waits for notification that a particular event has occurred --- mainloop function does this in Python.

Event Handler

- A process that should be executed in response to the event having been generated

Question

- In our example with the Mouse click, what was the event? What is the event handler?

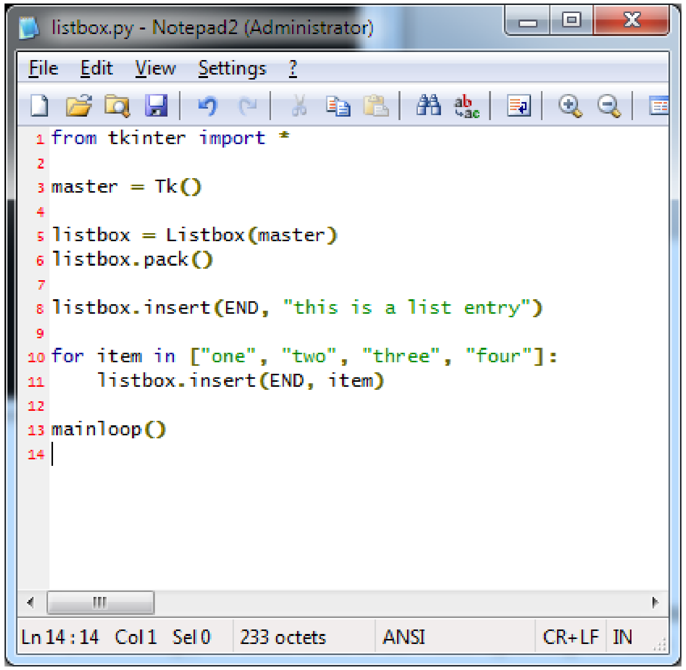
- Try it again!
  - Make a PhotoImage using beer.gif
  - Put that image on the canvas.
  - When mouse is clicked, move the beer to the clicked location

Lets create a drawing program

- **Goal:** Create a drawing program that allows us to draw lines easily

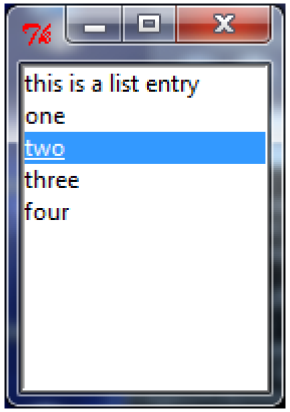
List boxes

- List boxes allow you to select one (or more) items from a list of items



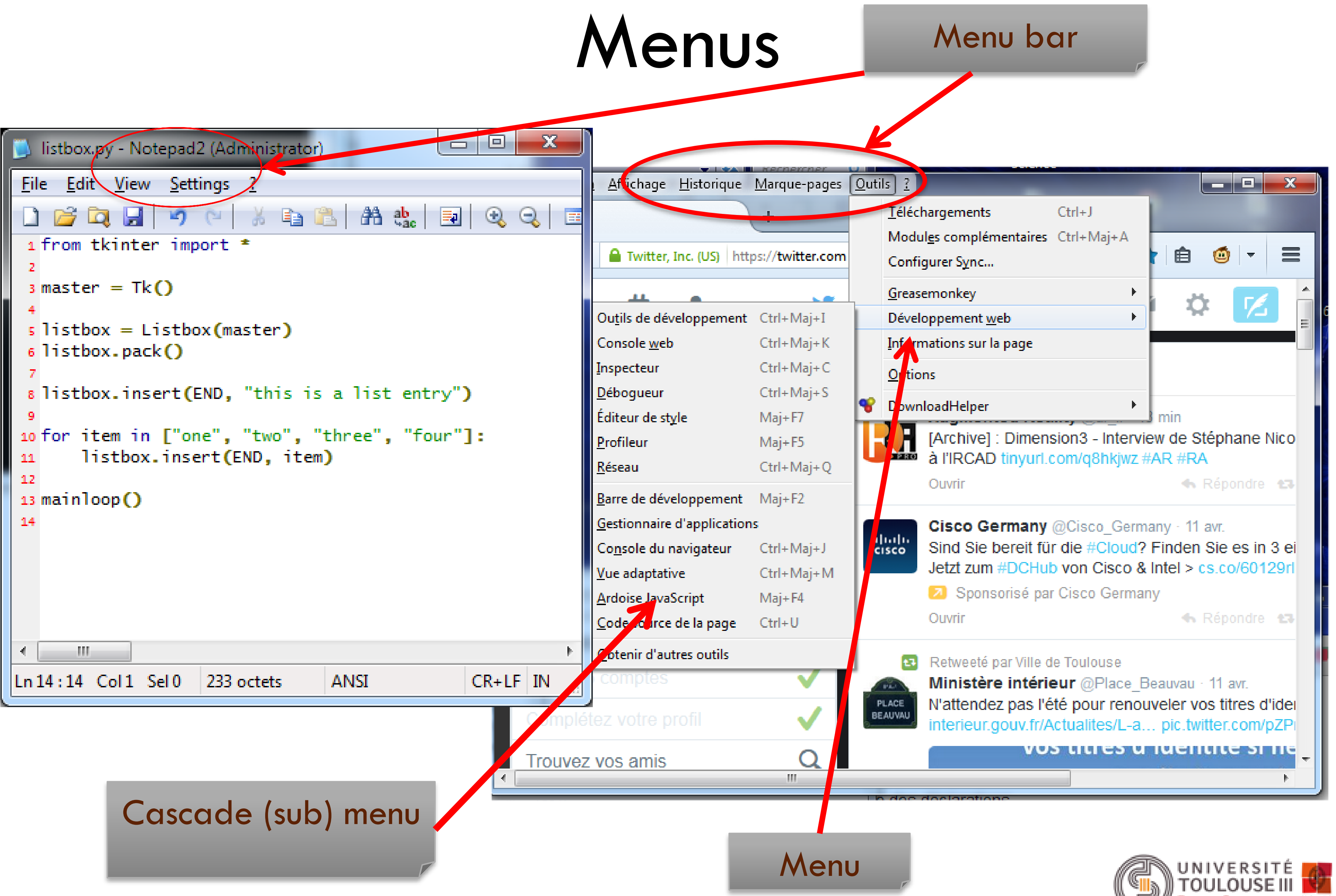
```
from tkinter import *

master = Tk()

listbox = Listbox(master)
listbox.pack()

listbox.insert(END, "this is a list entry")

for item in ["one", "two", "three", "four"]:
    listbox.insert(END, item)

mainloop()
```

listbox.py - Notepad2 (Administrator)
File Edit View Settings ?
Ln 14 : 14 Col 1 Sel 0 233 octets ANSI CR+LF IN



```
this is a list entry
one
two
three
four
```

Menus


Menu bar
Cascade (sub) menu
Menu
listbox.py - Notepad2 (Administrator)
File Edit View Settings ?
Ln 14 : 14 Col 1 Sel 0 233 octets ANSI CR+LF IN
Affichage Historique Marque-pages Outils ?
Téléchargements Ctrl+J
Modules complémentaires Ctrl+Maj+A
Configurer Sync...
Greasemonkey
Développement web
Informations sur la page
Options
DownloadHelper
Outils de développement Ctrl+Maj+I
Console web Ctrl+Maj+K
Inspecteur Ctrl+Maj+C
Débogueur Ctrl+Maj+S
Éditeur de style Maj+F7
Profileur Maj+F5
Réseau Ctrl+Maj+Q
Barre de développement Maj+F2
Gestionnaire d'applications
Console du navigateur Ctrl+Maj+J
Vue adaptative Ctrl+Maj+M
Ardoise JavaScript Maj+F4
Code source de la page Ctrl+U
Obtenir d'autres outils
Twitter, Inc. (US) https://twitter.com
Ouvrir
Répondre
Sponsorisé par Cisco Germany
Retweeté par Ville de Toulouse
comptes
Complétez votre profil
Trouvez vos amis

```python
from tkinter import *

master = Tk()

listbox = Listbox(master)
listbox.pack()

listbox.insert(END, "this is a list entry")

for item in ["one", "two", "three", "four"]:
    listbox.insert(END, item)

mainloop()
```

[Archive] : Dimension3 - Interview de Stéphane Nico à l'IRCAD tinyurl.com/q8hkjwz #AR #RA

Cisco Germany @Cisco_Germany · 11 avr.
Sind Sie bereit für die #Cloud? Finden Sie es in 3 ei Jetzt zum #DCHub von Cisco & Intel > cs.co/60129rl

Ministère intérieur @Place_Beauvau · 11 avr.
N'attendez pas l'été pour renouveler vos titres d'ide interieur.gouv.fr/Actualites/L-a... pic.twitter.com/pZP

Adding Menus

- A menu is simply another type of widget.

```
# create a toplevel menu
menubar = Menu(root)
```

The menubar is a container for Menus

```
# create a pulldown menu, and add it to the menu bar
filemenu = Menu(menubar)
```

Create a single menu

```
filemenu.add_command(label="Open", command=hello)
```

Call the hello function when the Open menu option is chosen

```
filemenu.add_separator()
```

Add a line separator in the menu

```
filemenu.add_command(label="Exit”,command=root.destroy)
```

Call the root.destroy function when the Exit menu option is chosen

```
menubar.add_cascade(label="File", menu=filemenu)
```

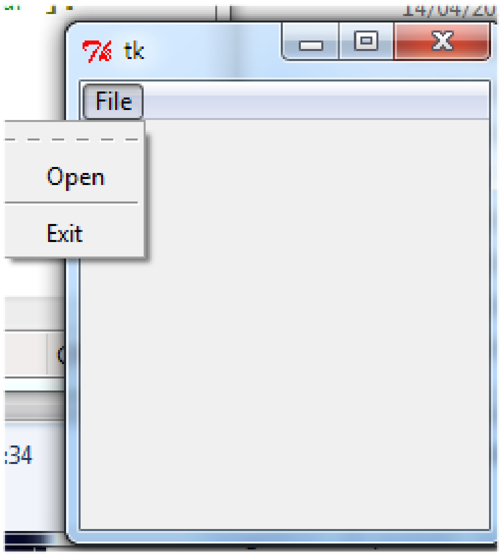
Add the filemenu as a menu item under the menubar

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

Tell the root window to use your menubar instead of default

Adding Menus

```
# create a toplevel menu
menubar = Menu(root)

# create a pulldown menu, and add it to the menu bar
filemenu = Menu(menubar)
filemenu.add_command(label="Open", command=hello)
filemenu.add_separator()
filemenu.add_command(label="Exit”,command=root.destroy)
menubar.add_cascade(label="File", menu=filemenu)
root.config(menu=menubar)
```



Adding Sub-Menus

Adding sub-menus, is done by adding a menu to another menu instead of the menubar.

```
# Create another menu item named Hello
helloMenu = Menu(menubar)
helloMenu.add_command(label="Say hello", command=hello)
menubar.add_cascade(label="Hello", menu=helloMenu)

# Create a submenu under the Hello Menu
subHello = Menu(helloMenu) # My parent is the helloMenu
subHello.add_command(label="English", command=hello) # Menu Item 1
subHello.add_command(label="Spanish", command=hello) # Menu Item 2
subHello.add_command(label="Chinese", command=hello) # Menu Item 3
subHello.add_command(label="French", command=hello)  # Menu Item 4

# Add sub menu into parent with the label International Hello
helloMenu.add_cascade(label="International Hello", menu=subHello)
```

Try it!

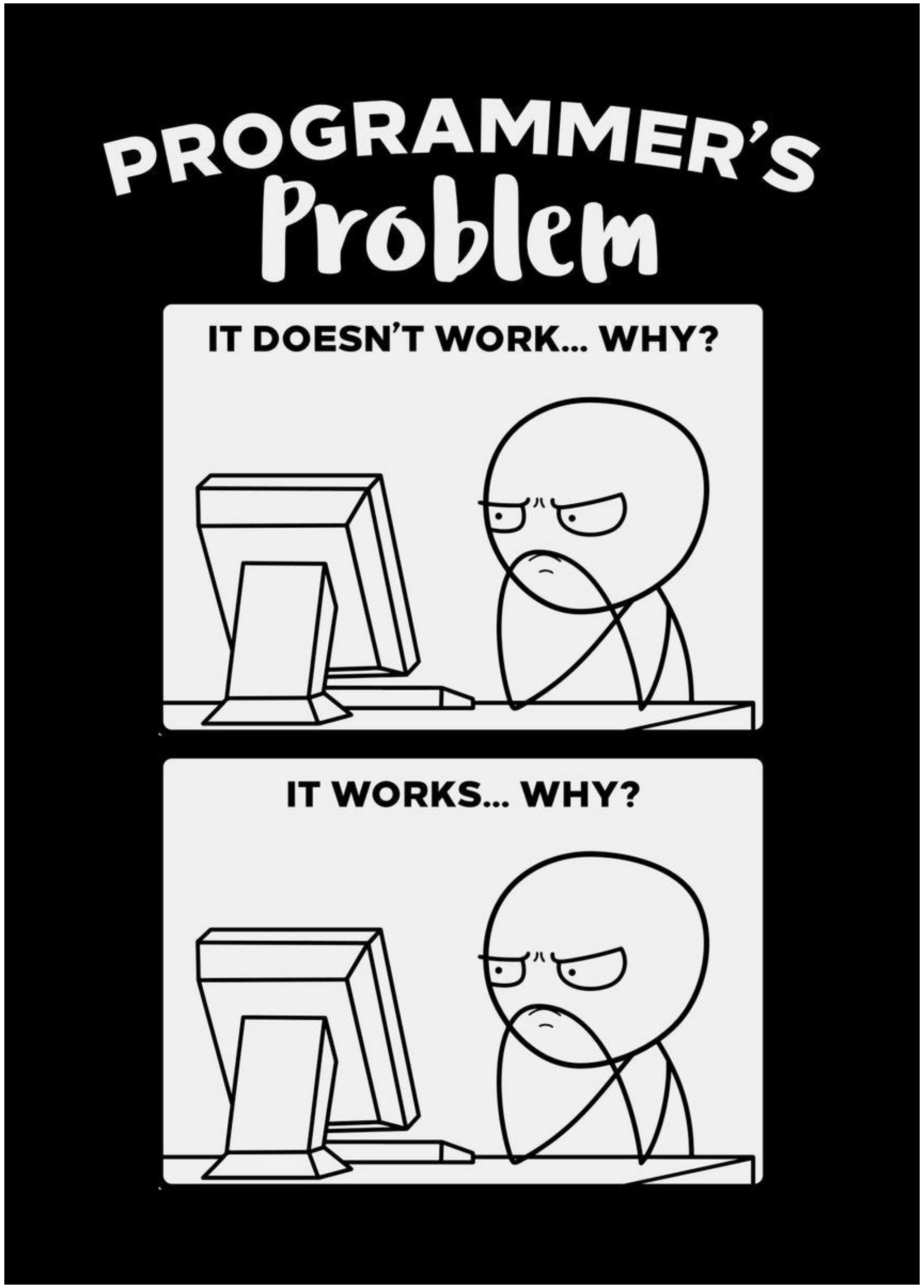
- Create a window that has a changeable label in it.
- Create a File menu with Open a File, Exit Program options. Make the “Open a File” option open a file dialog box using askopenfilename and show the filename in the changeable label.
- Make Exit Program option close the window.

And more!

- **ttk** : https://www.pythontutorial.net/tkinter/tkinter-ttk
- **GUIZero** : https://lawsie.github.io/guizero

Many more widgets

- https://docs.python.org/3/library/tkinter.html
- https://realpython.com/python-gui-tkinter


PROGRAMMER'S Problem
IT DOESN'T WORK... WHY?
IT WORKS... WHY?