Python workshop

Week 4: Files and lists

barbera@van-schaik.org

Overview of this workshop series

- Week 1: Writing your first program
- Week 2: Make choices and reuse code
- Week 3: Loops and strings
- Week 4: Files and lists
- Week 5: Dictionaries and tuples

Acknowledgments: Structure of the workshop follows the book
“Python for informatics” by Charles Severance.
Several examples are from this book or the accompanying slides.


WHEN YOU'VE BEEN WAITING FOR A RECAP
FOR OVER 2 WEEKS
daebakdrivel.tumblr.com

Recap

- Making choices
  - *if, elif, else* (test if something is True or False)
  - *try, except* (test if python fails on something, if so, do something else)
- Loops
  - *while* (go on and on while condition is True)
  - *for* (go through a list, a range, a file)
  - *continue, break* (go to start of loop, break the loop)
- Functions
  - *def*, arguments, *return* values

How to continue after this course?

- http://coursera.org
  - Programming for everybody (repeat what you learned already with more examples)
  - Python data structures (idem)
  - Using Python to access web data
  - Using databases with Python
  - Interactive programming (e.g. games)
  - Raspberry Pi and Python (IoT)
  - … and more

- http://edx.org
  - Introduction to Computer Science and programming using Python
  - Computational thinking and data science
  - … and much more
- https://www.codecademy.com/

How to continue? Other languages

- Web: HTML5, Javascript, PHP
- Apps: Java, C, C++, C#
- Statistics/math: R, matlab
- Electronics: C, Arduino
- Heavy calculations: C

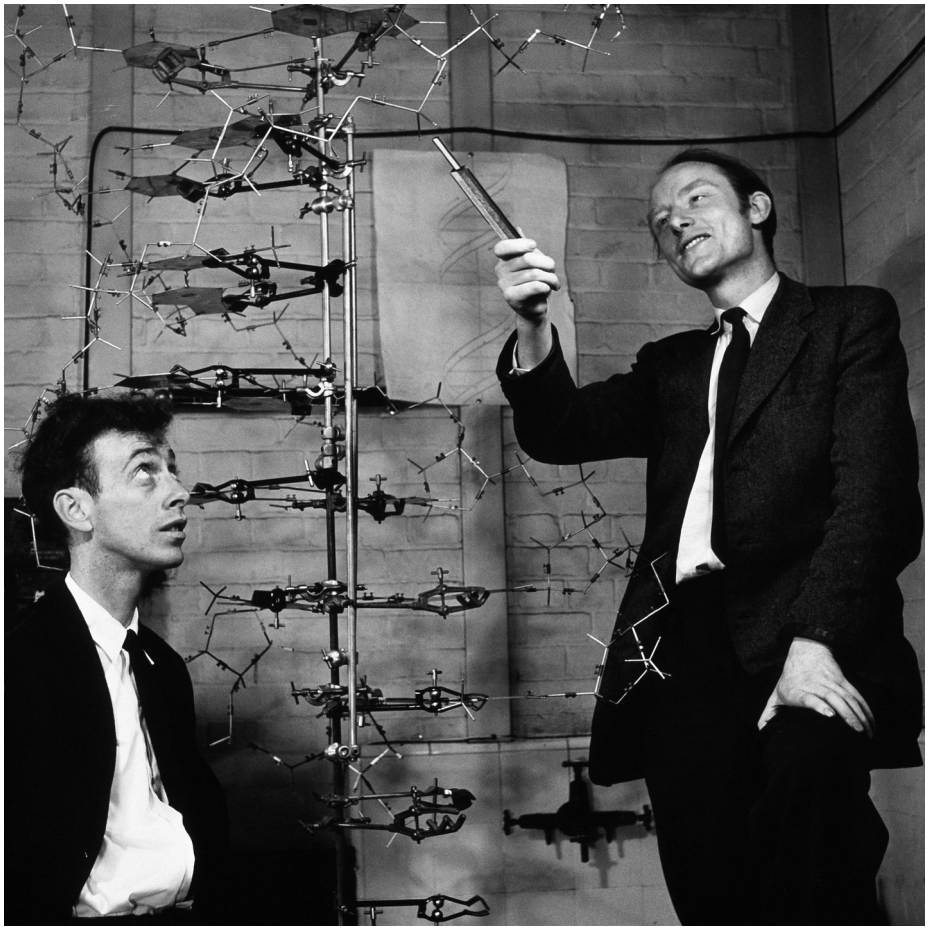
Disclaimer: several languages can be used to do the same, but these are often used for these purposes and this is definitely not a complete list

How to continue: application areas

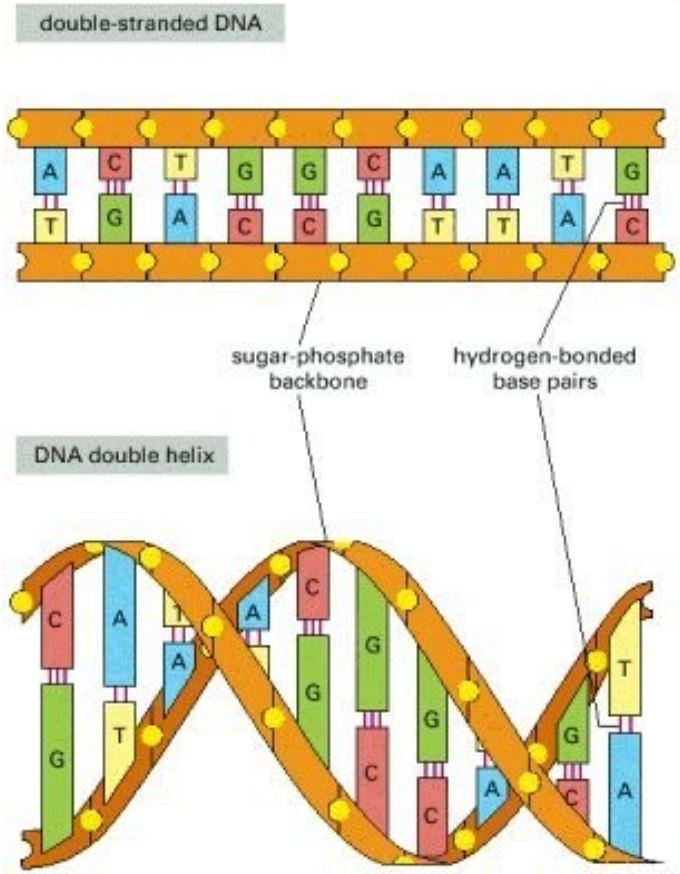
- Games
- Web applications
- Mobile applications
- Science, big data and/or math (e.g. life sciences, physics, finances)
- Cloud or high performance computing
- Computer graphics
- etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc, etc,

A bioinformatics example

DNA structure



DNA



- A opposite T
- G opposite C

Get complement reverse of DNA

You get this sequence/string:

ACTGCCCCAAAATTTGGG

The complement (A-T, C-G) is this:

TGACGGGGTTTTAAACCC

Then reverse the string:

CCCAAATTTTGGGGCAGT

How to solve the DNA puzzle?

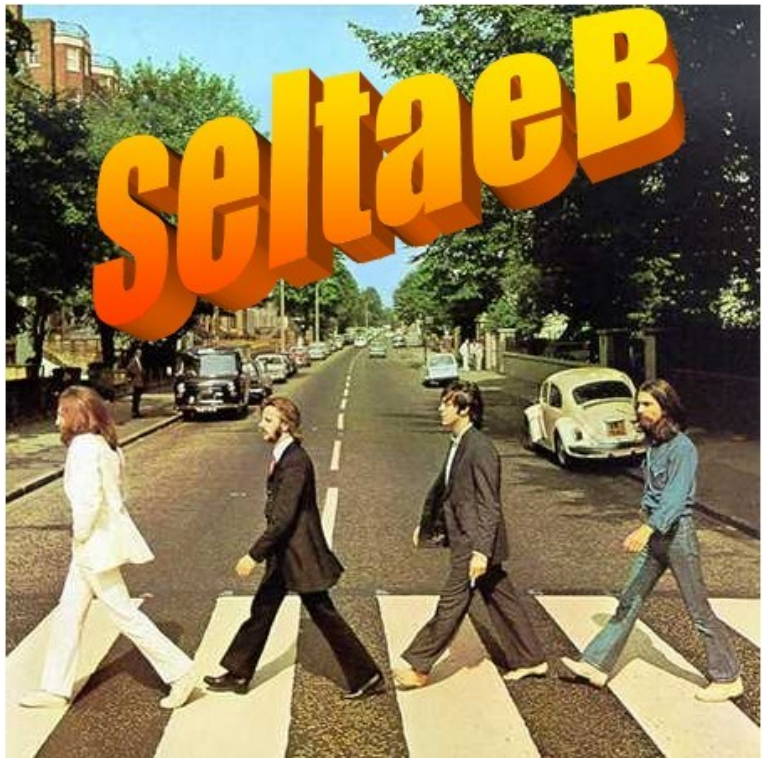
If there is an A, transform it to a T
If there is a T, make it into an A
If there is a C, make it a G
If it is a G, make it a C

Then reverse the entire string

dna.py

Reverse a string

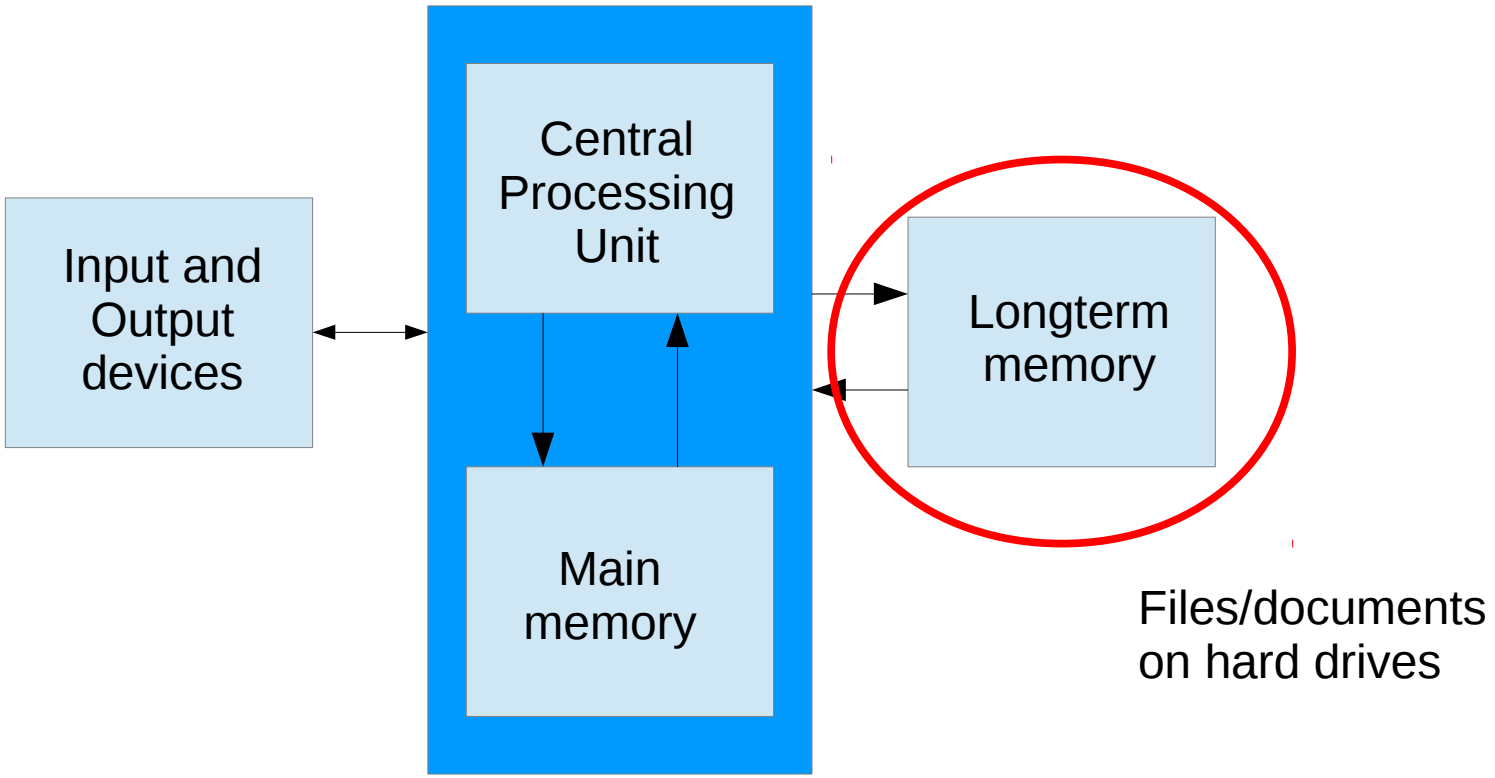
```
>>> s = “Strawberry fields forever”
>>> s[::-1]
'reverof sdleif yrrebwartS'
```



https://youtu.be/09SdN_a1JO8

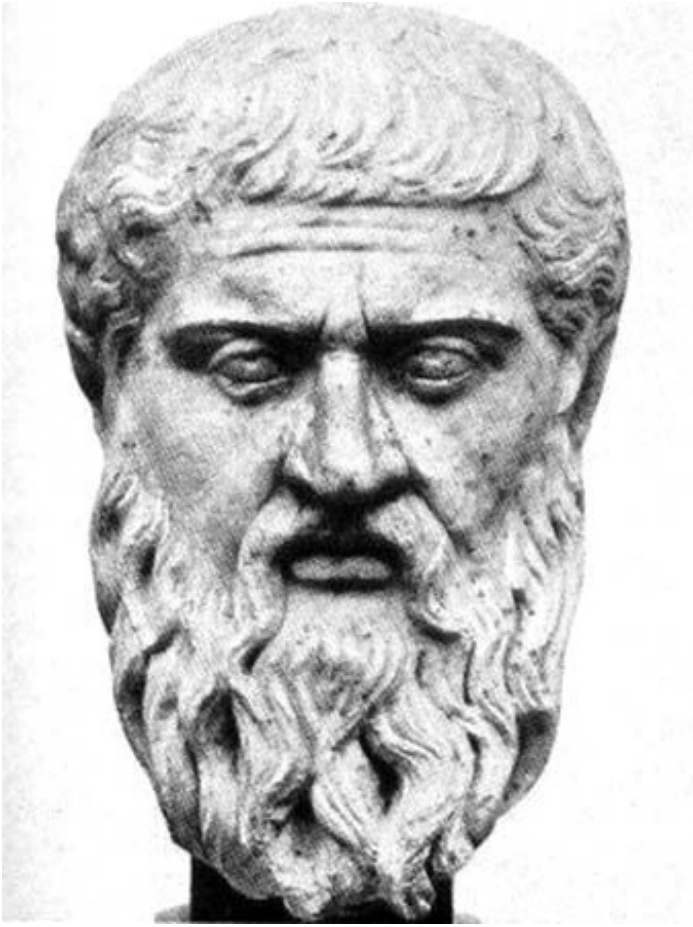
Files

Files


Input and Output devices
Central Processing Unit
Main memory
Longterm memory
Files/documents on hard drives

Plato - Gorgias

- Illusion of logos (I just picked a random text from Plato)
- Persons of the dialogue: Callicles, Socrates, Chaerephon, Gorgias, Polus
- Abbreviated with: Cal., Soc., Chaer., Gor., Pol.



plato.txt

Part of the text

Soc. How fortunate! will you ask him, Chaerephon-?

Chaer. What shall I ask him?

Soc. Ask him who he is.

Chaer. What do you mean?

Read files

```
fh = open("plato.txt")


for line in fh:
    print line
```

open-file.py

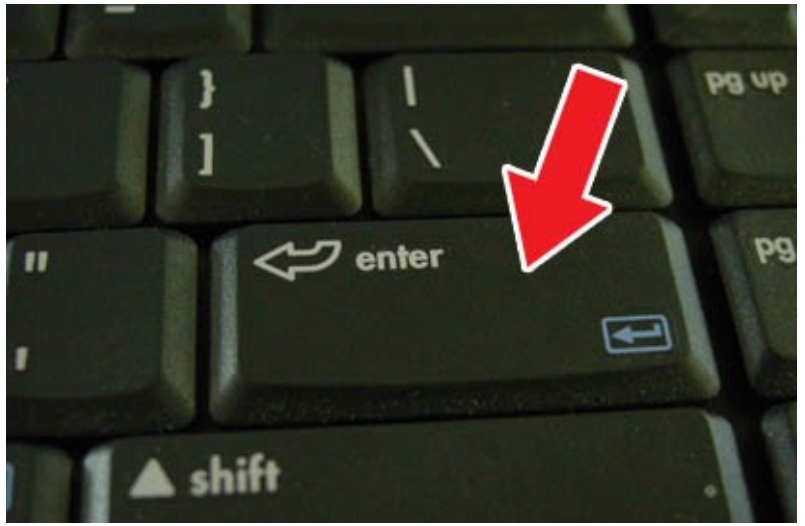
Why do you get extra empty lines between the lines?

Answer: each line is read **including** the return/newline at the end!

You can print a return yourself:
`print “Blah\n”`

How to remove the newline?

```
fh = open("plato.txt")


for line in fh:
    line = line.rstrip()
    print line
```

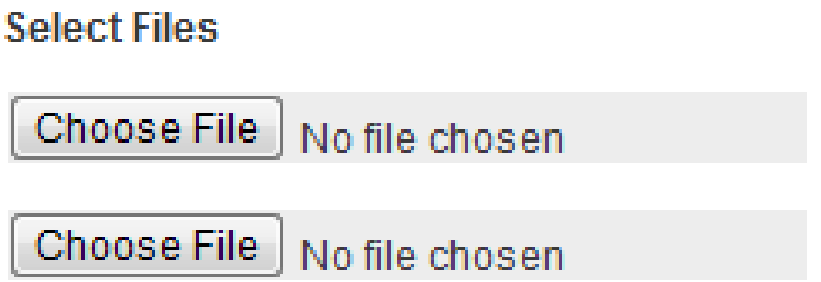


Let the user choose a file

- And check if everything goes right with try/except

```
import sys
myfile = raw_input("Enter filename: ")
try:
    fh = open(myfile)
except:
    sys.exit("cannot open file")
for line in fh:
    line = line.rstrip()
    print line
```

ask-user-for-file.py



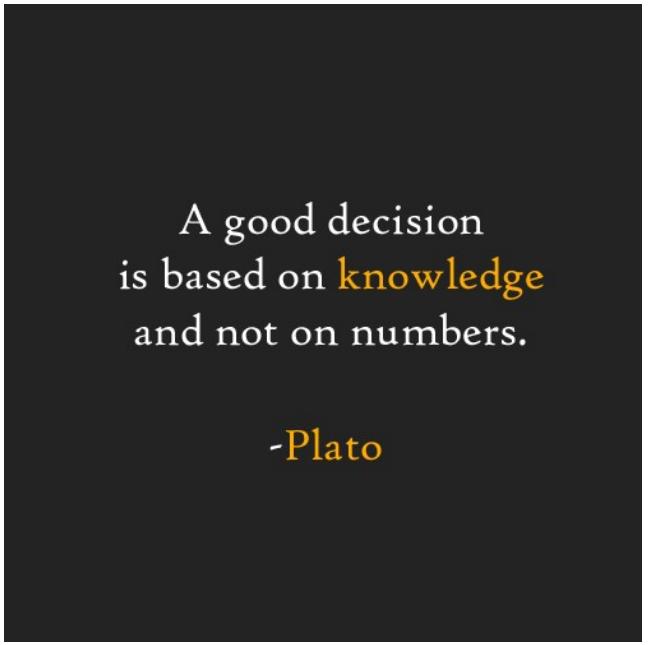
Providing a file via the commandline

```
import sys
if len(sys.argv) < 2:
    sys.exit("Usage: thisscript.py somefile.txt")
myfile = sys.argv[1]
try:
    fh = open(myfile)
except:
    sys.exit("cannot open file")
for line in fh:
    line = line.rstrip()
    print line
```

Run it like this from the commandline:
>python ask-user-for-file2.py plato.txt

Count lines in a file

```
import sys

myfile = "plato.txt"

fh = open(myfile, "r")


count_lines = 0

for line in fh:

    count_lines = count_lines + 1


print "File contains", count_lines, "lines."
```



count-lines.py

Search for stuff in a file

```
count_socrates = 0
count_callicles = 0
for line in fh:
    line = line.strip()
    if line.startswith("Soc."):
        count_socrates = count_socrates + 1
    elif line.startswith("Cal."):
        count_callicles = count_callicles + 1


print "Socrates spoke", count_socrates, "times"
print "Callicles spoke", count_callicles, "times"
```

Writing files

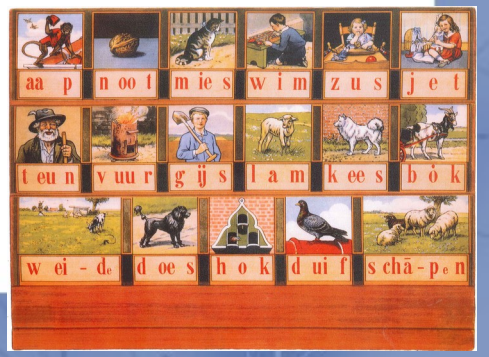
```
fh = open("snoepjes.txt", "w")


for i in range(10):
    print >> fh, i, "Ik mag niet met snoepjes gooien"

fh.close()
```

snoepjes2.py

Lists

Lists and indices

```
>>> cijfers = [10,20,30,40,50,60]
>>> woorden = [“aap”,”noot”,”mies”]
>>> leeg = []


>>> print woorden, cijfers, leeg
>>> print woorden[2]
>>> print woorden[10] # error
```



Populate lists

```
>>> mylist = range(0,11,2)
>>> mylist
```

```
>>> zin = “Dit is een zin”
>>> woorden = zin.split()
>>> woorden
```

Default: it splits on a space or a tab
For comma-separated files: zin.split(“,”)

Slices

```
>>> line = “scaramouch scaramouch will you do the fandango”
>>> words = line.split()
>>> words
>>> words[2:5]
>>> words[0]
>>> words[0][1]
```

List operations

```
>>> a = [1,6,9]
>>> b = [2,4,6]
>>> c = a+b
>>> c
>>> 3 * a
>>> d = [1,2,b]    # list in a list
>>> d
>>> len(d) # is this what you expect?
```

Mutability

- Strings are NOT mutable
- Lists are

```
>>> c
>>> c[2] = 108
>>> c
```

Loops

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
>>> for cheese in cheeses:
>>>     print cheese
```

Check if element is present in list

```
>>> cheeses = ['Cheddar', 'Edam', 'Gouda']
>>> 'Edam' in cheeses
>>> 'Brie' in cheeses
```

List functions

```
>>> a = [“z”,”o”,”b”]
>>> b = [“e”,”d”,”c”]
```

- Append to a list

```
>>> a.append(“x”)
```

- Extend list with another list

```
>>> b.extend(a)
```

- Sort list

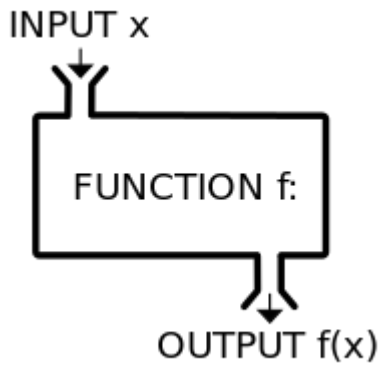
```
>>> b.sort()
```

- Pop, remove element from list and return it

```
>>> b.pop(2)
>>> b.pop()
```

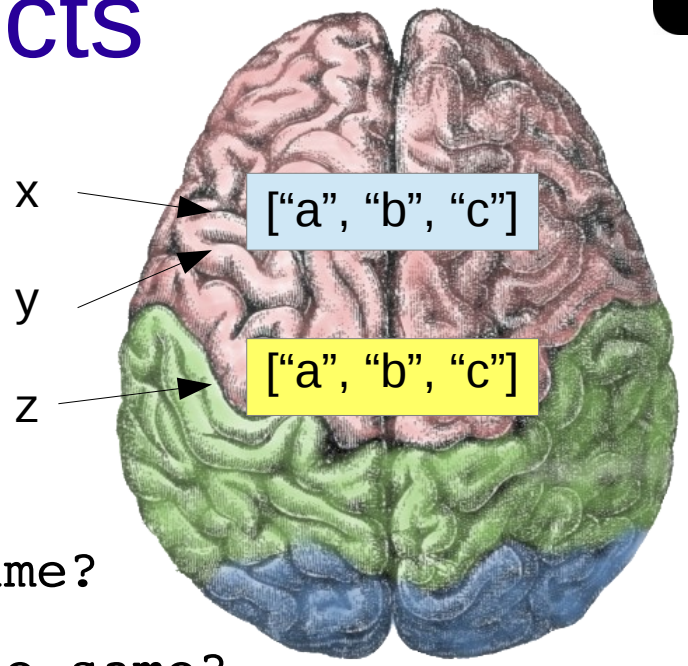
Apply functions to lists

```
>>> nums = [3,41,12,9,74,15]
>>> len(nums)
>>> max(nums)
>>> sum(nums)
>>> sum(nums)/len(nums)
```



List objects

```
>>> x = [“a”, “b”, “c”]
>>> y = x
>>> z = [“a”, “b”, “c”]
```




```
>>> x is y # is the object the same?
>>> x is z # are these objects the same?
>>> x == z # are the values the same?
```

```
>>> y[0] = “bla”
>>> y
>>> x
```

If you change y, x is also changed
x and y point to same “thing” in memory

Two ways to make a copy that you can change without changing the original list

```
>>> x = [“a”,”b”,”c”]
>>> y = x[:]
>>> y = list(x)
```

Summary

- Files
  - Open, read, write
  - Parse elements from file
- Lists
  - A collection of words, letters, numbers, and even lists
  - List methods (append, pop, etc) and functions on lists (sum, len, etc)
- Files and lists
  - Get words from a file or specific columns

Assignment 1

- How many lines with the word “true” and how many with “false” in plato.txt?
- Hints
  - Open file
  - Make two variables to count “true” and “false”
  - Use the string method: find

Assignment 2

- Open the file “hobbies.txt” and print the names of the persons
- Hints:
  - Open file
  - “split” the lines
  - Get the right column and print it

Bonus exercise – guessing game

- Let the user think of a number between 1 and 1000
- The computer makes guesses
- The user gives hints: higher/lower (or h/l)

One solution: let computer guess all the numbers between 1 and 1000 (not so efficient)

How would you solve this?

With code or as a concept

Next week

- Next: Dictionaries and tuples
- More programming exercise?
  - Chapter 7 and 8 of the book