EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Science
Department of Computer Science

Vorkurs Programming - Informatik 4 Lifescientists

Introduction into Programming & Scripting in Python

Part 2

Leon Bichmann & Theresa Harbig

You make good progress.

You were faced to a lot of typing and probably self-made typos (syntax errors).

From now on we assume, you are more experienced in writing code and enable the copy paste option to proceed faster.

Thanks for understanding!

☺

CHAPTERS

1) Basic terms
2) First Steps in Python
3) Data Types and more (e.g. variables, data types, functions)
4) Conditional Programming and loops (e.g. if statements, for loops)
5) Import packages and work on files (open, write, close files)
6) Itertools and groupby and other related packages
7) Flow control
8) Introduction to matplotlib and numpy
9) How to be a good programmer?
10) Biopython

Import packages and work on files

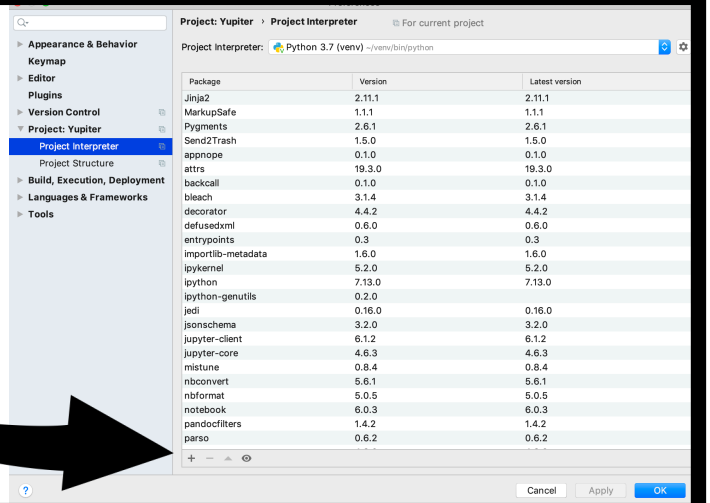
What is a package (module)?

Installing packages ...

In PyCharm:

To do so, go to Settings, Preferences or Configurations of your Project.

There you can check or add / install new packages like os or biopython.



Some packages are built-in packaged and come with the installation automatically, such as the package **os** (Miscellaneous **operating system** interfaces) .

How to load the package "os"?

```
import os
print(os.listdir()) # a function of the package os
```

Look up all functions of the package os!

How to load a function of the package "os"?

```
from os import listdir
print(listdir())

from os import listdir as ld  # create an abbreviation of a function name
print(ld())
```

What is the output of the listdir() function and its data type?

Count the files in the current directory!

```
print(len(listdir())) # default argument is ".", current directory

print(len(listdir(".")))  # check current directory "."

print(len(listdir("..")))  # check upper directory ".."

print(len(listdir("./..")))  # take a relative or absolute path as argument
```

Remember the linux commands / short cuts for traversing the file system.

Iterate over the files in the current directory and report on their names per line using print() and, finally, report on the total file number applying len().

```
for file in listdir():
    # please fill here
```

Change your script so that it consists of three functions:

```
def get_list_of_files(path):
    # fill here

def report_on_list_len(path_list):
    # fill here

def main():
    file_path = #fill here
    file_list = get_list_of_files(file_path)
    report_on_list_len(file_list)

# execute the main function only
if __name__ == "__main__":
    main()
```

Is there a way to reduce the lines (and variables) of the main() function?

Look up the convention of using a main function.

How to read and write files?

```
# open file for writing
f = open("new_file.txt", "w")

# only strings can be written to the file:
for i in range(10):
   f.write(str(i) + "\n")

# don't forget to close the file:
f.close()
```

For writing you have to specify a filename and "w" in the open() function!
Pay attention to the newline operator \n!
What happens if you remove it.

```
# open file for reading
f = open("new_file.txt", "r")

# get text from file:
a = f.read()
#a=f.readline()
#a=f.readlines()
print(a)

# don't forget to close the file:
f.close()
```

For reading you have to specify a filename and "r" in open()!

Comment and uncomment the 3 lines respectively and rerun the code. How does the output differ for all three functions?

Instead of writing to a file, what will overwrite an old content, you may wish to append intformation by using "a" in open()!

Itertools and groupby and other related packages

Another package is "itertools" ...

```
# use groupby
from itertools import groupby

def keyfunc(some_element_of_list):
    if some_element_of_list[0].isupper():
        return 'Upper'
    return 'Lower'

my_list = ['B', 'i', 'o', 'i', 'n', 'f', 'o', 'r', 'm', 'a', 't', 'i', 'c', 's']
print(sorted(my_list))

my_list = sorted(my_list, key=keyfunc)
print(my_list) # Do you notice any difference?

for k, g in groupby(my_list, keyfunc):
    print('starts with', k, '->', ' '.join(list(g)))
```

What does groupby return? What happens when you remove sorting?
Look up all itertools functions creating iterators for efficient looping!

Combinations and products ...

```
from itertools import combinations, product

my_list_1 = ['a', 'b', 'c']
my_list_2 = [1, 2, 3]

# all combinations
print(list(product(my_list_1,my_list_2)))
```

```
[('a', 1), ('a', 2), ('a', 3), ('b', 1), ('b', 2), ('b', 3), ('c', 1), ('c', 2), ('c', 3)]
```

	a	b	c
1	a1	b1	a1
2	a2	b2	b2
3	a3	b3	c3

```
# lower triangle matrix
print(list(combinations(my_list_1, 2)))
```

```
[('a', 'b'), ('a', 'c'), ('b', 'c')]
```

	a	b	c
a	aa	ba	ca
b	ab	bb	cb
c	ac	bc	cc

WORK ON YOUR OWN !

Write a function to list files (not catalogues) present in the given directory: *get_file_info(path)*

The function should take a path to the directory as a parameter/argument, default path is the current directory.

Print out should be formatted in the following way:

overall 27 files in direcory: [path]
png: 10 files
md: 3 files
fasta: 12
files without type: 2 files

Function should return the Dictionary of file_types: list of files.
Do not forget to use the main() function!

WORK ON YOUR OWN !

1) Write the output of the previous written *get_file_info(path)* function into a new file, called “my_dir_stats.txt”

2) Read the *new_file.txt* and display only the last 5 lines.

Another package is "collections" ...

```
# Another very useful package, and example of how to import subclasses (functions)
from collections import Counter

my_list = ['B', 'i', 'o', 'i', 'n', 'f', 'o', 'r', 'm', 'a', 't', 'i', 'c', 's']
print(Counter(my_list))
```

What does Counter return? Write a function to print any result of the Counter in nicely formatted and sorted, use list comperhension.

WORK
ON
YOUR
OWN !

Redo: count **a**,**t**,**g**,**c** in a given string using a dictionary, using collections.Counter .

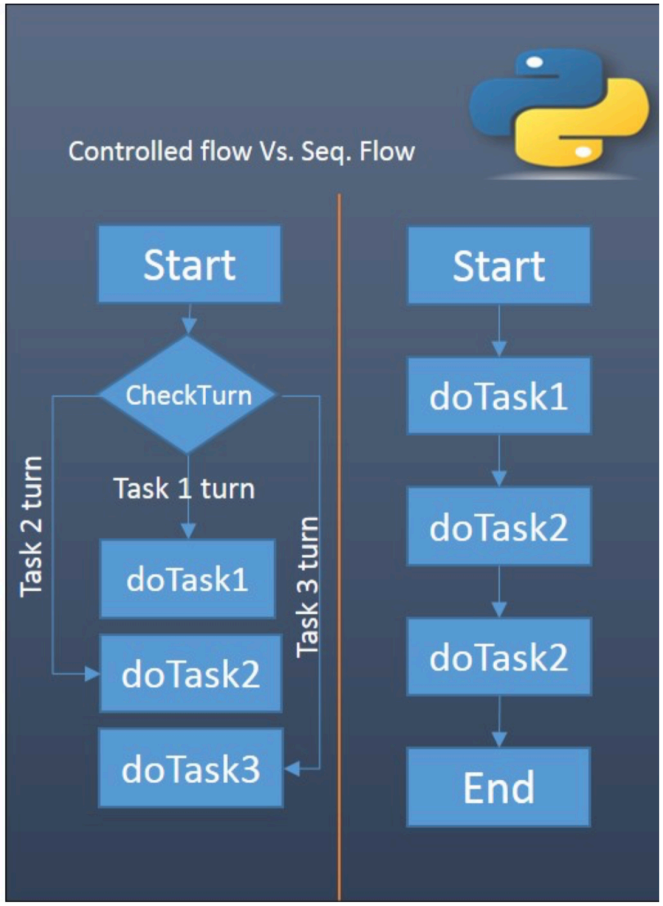
```
sequence = “atgaagattc”
```

Flow Control

Flow control

`if/elif/else`
`for`
`while`
functions

`pass`
`break`
`continue`



break

```
for letter in 'Python':
   if letter == 'h':
      break
   print('Current Letter :', letter)
```

continue

```
for letter in 'Python':
   if letter == 'h':
      continue
   print('Current Letter :', letter)
```

Check the differences!

pass

```
for letter in 'Python':
   if letter == 'h':
      pass
   print('Current Letter :', letter)
```

Pass you will use quite often, to avoid adding content to loops or functions while solving a complex task and using/planning its sub-functions.

Introduction to matplotlib and numpy

The Matplotlib package ...

import matplotlib

Type and run!

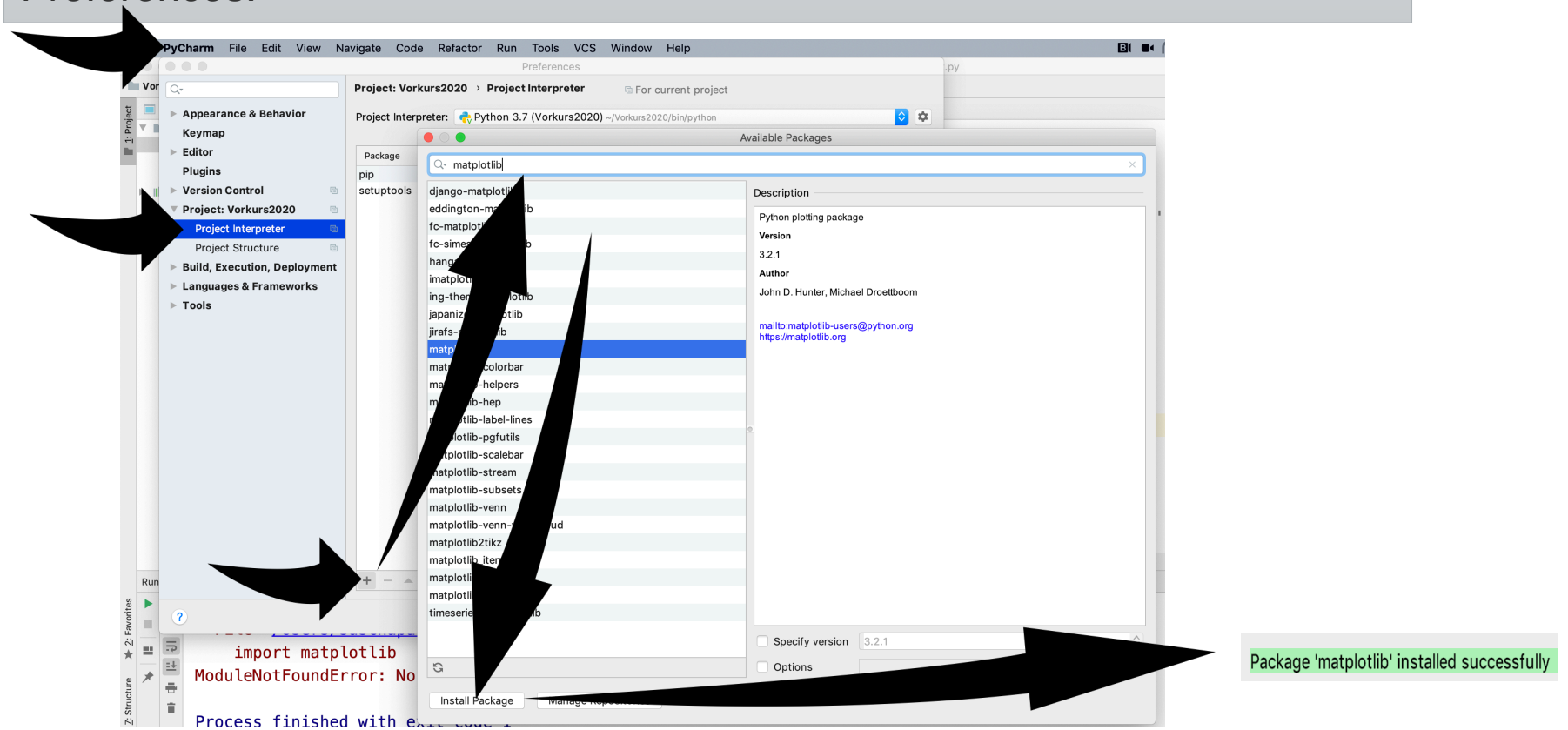
```
Traceback (most recent call last):
  File "/Users/saschapatz/PycharmProjects/Vorkurs2020/python_script.py", line 353, in <module>
    import matplotlib
ModuleNotFoundError: No module named 'matplotlib'

Process finished with exit code 1
```

As written before, you have to install / add the package via Settings or Preferences.

The Matplotlib package ...

As written before, you have to install / add the package via Settings or Preferences.



The Matplotlib package ...

```
import matplotlib
```

Re-run!

The Matplotlib package ...

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('ggplot')
```

Import packaged and functions.

Define style.

The Matplotlib package ...

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('ggplot')
```

Import packaged and functions.

Define style.

```
# example of a linechart
x = np.linspace(0, 3*np.pi, 500)
plt.plot(x, np.sin(x**2))
plt.title('A simple chirp')
plt.xlabel('X label')
plt.ylabel('Y label')
plt.show()
```

Your first plot of simulated data.

Try to understand the code and functions! Perhaps use print().

What would happen if you remove the style declaration? What other styles are there in matplotlib package? Look up all possible numpy.linspace() function input arguments.

The Matplotlib package ...

```
import matplotlib
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('ggplot')
'''
# example of a linechart
x = np.linspace(0, 3*np.pi, 500)
plt.plot(x, np.sin(x**2))
plt.title('A simple chirp')
plt.xlabel('X label')
plt.ylabel('Y label')
plt.show()
'''
```

After running the code, use block comment to inactivate code. What happens if you don't do so?

Barchart

```
# example of a barchart
D1 = {'Label0':26, 'Label1': 17, 'Label2':30}
D2 = {'Label0':16, 'Label1': 10, 'Label2':10}
xx = np.arange(len(D1))
plt.bar(xx, D1.values(), align='center', color='blue', width=0.4)
plt.bar(xx + 0.4, D2.values(),align='center', color='red', width=0.4)
plt.xticks(xx + 0.2, D1.keys())
plt.show()
```

AGAIN:
After running the code, use block comment to inactivate code.

What is the output of the function numpy.arange() and what are its input arguments. How does the output for np.arange(3,7,2) looks like?

Boxplot

```
import random
# make a random subset
local_list = []
for i in range(3):
    local_list.append(random.sample(range(300*i, 1000 + 300*i), 100))
plt.boxplot(local_list, labels=['A', 'B', 'C'])
plt.savefig('exemplary_boxplot.png') # write figure into a file #
plt.show()
```

AGAIN:
After running the code, use block comment to inactivate code.

WORK
ON
YOUR
OWN !

Check out the matplotlib Pyplot tutorial:

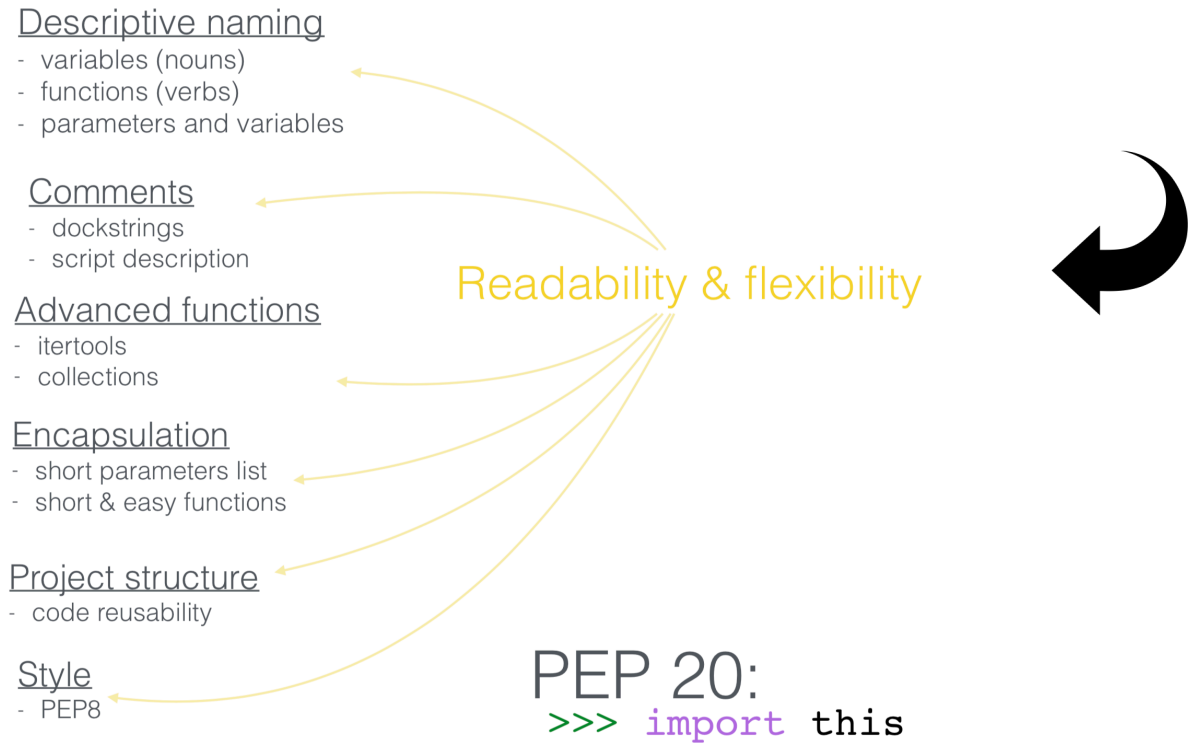
https://matplotlib.org/tutorials/introductory/pyplot.html

See how far you get!

How to be a good programmer?

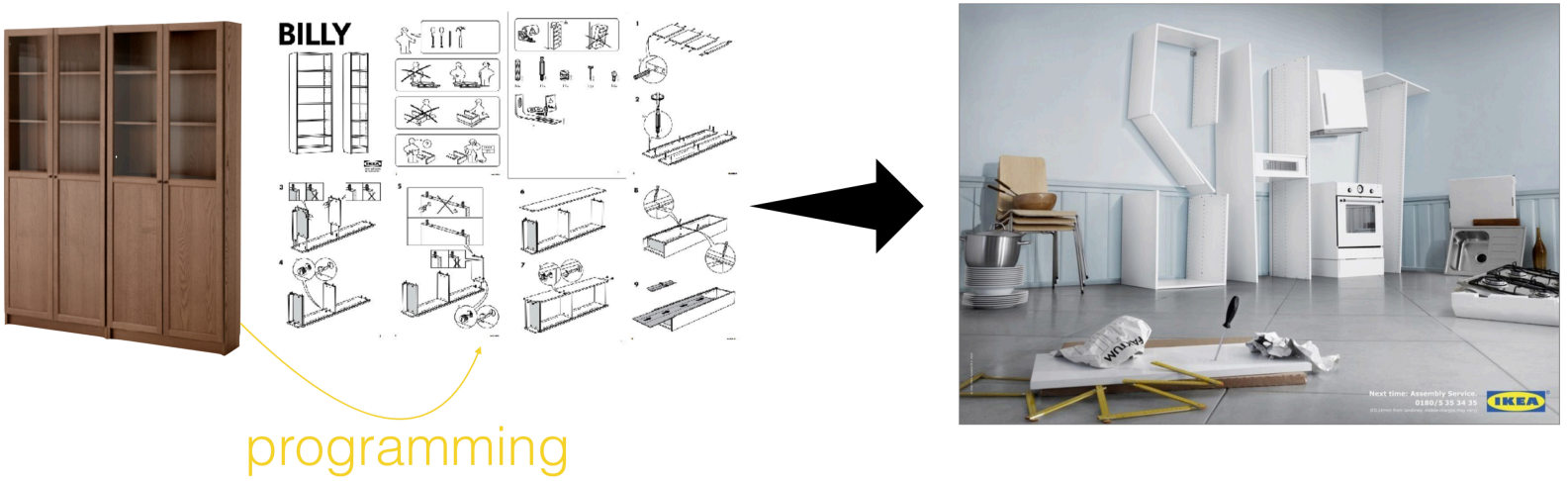
A well written program?

. . . everybody can read and understand it.



Ability to program?

. . . you can divide a task into its sub-tasks?



How to be a good programmer?

Setup: 3 elements

Python version(s)

IDE

Python packages

```
pip3 list
pip3 show <package name>
sudo pip3 install <package name>
```

```
In [8]: import numpy

In [9]: numpy.__file__
Out[9]: '/Library/Frameworks/Python.framew
ork/Versions/3.5/lib/python3.5/site-packag
es/numpy/__init__.py'
```

Stay updated, check versions, IDE's and packages. Use pip3 package installer, …

Keep up: books, docks, blogs, podcasts

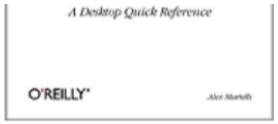
more:

» PythonBooks

» Ten things ... bout Python » PythonBooks

ROSALIND

http://rosalind.info/problems/locations/



python SOFTWARE FOUNDATION

- » ScientificProgrammingBooks
- » SystemAdministrationBooks
- » WebProgrammingBooks
- » WindowsBooks
- » XmlBooks
- » ZopeBooks

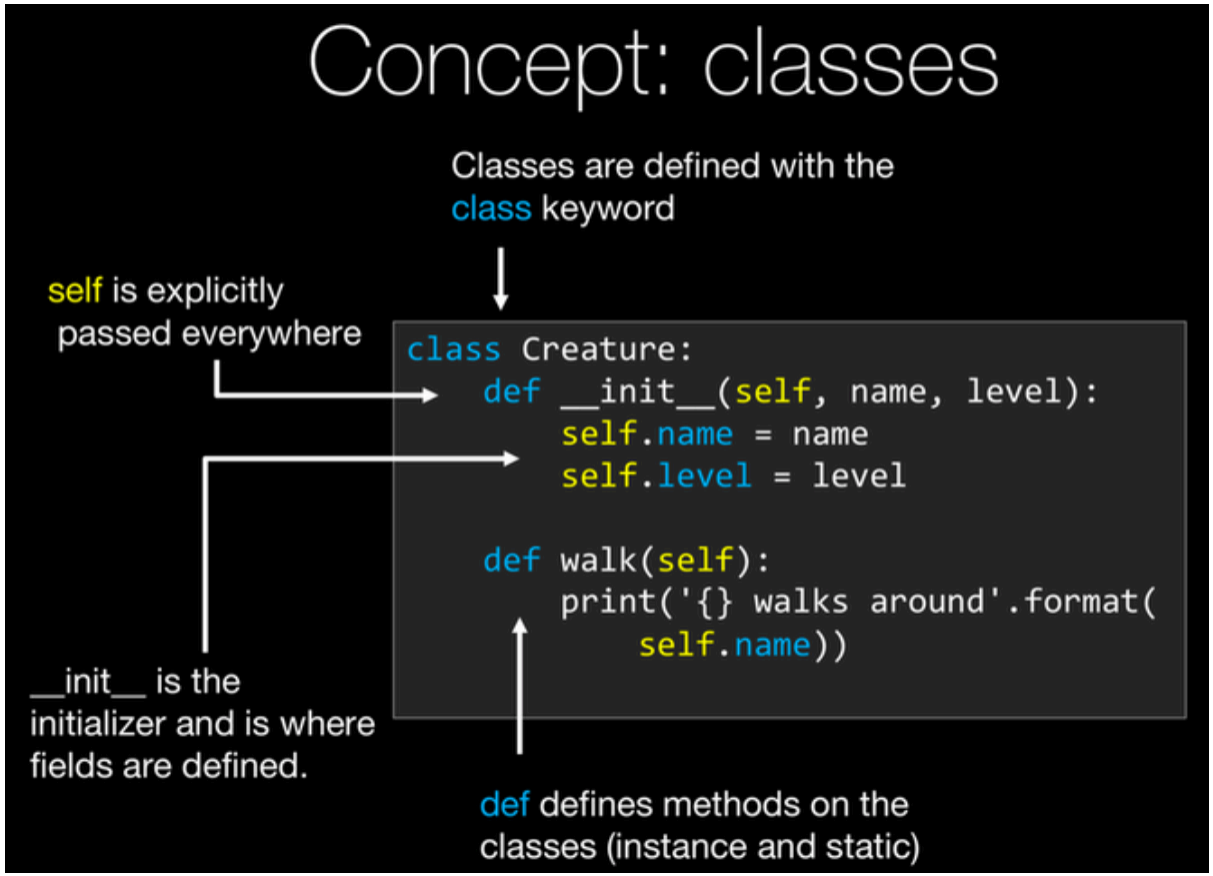
https://wiki.python.org/moin/PythonBooks

https://docs.python.org/3/tutorial/

Train yourself with online tutorials?

http://rosalind.info/problems/locations/

Advanced: Object oriented programming ...



Look up the concept of object-oriented programming by using classes.

You will learn more about it in Java!

Congratulation, you are done with Python.

Of course it was just a tiny and quick introduction, so from now on you have to improve you skills in programming step by step.

It may be that Python is not the programming language that fit best to you or to your future aims!

Homework: Biopython

Checkout Biopython!

https://biopython.org/wiki/Getting_Started

WORK
ON
YOUR
OWN !

Redo: reverse complement using Biopython

```
sequence = “atgaagattc”
```