EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Faculty of Science
Department of Computer Science

Vorkurs Programming - Informatik 4 Lifescientists

Introduction into Programming & Scripting in Python

Part 2 - https://github.com/BioInfPrep/python_basics

13/10/2021, Mathias Witte Paz, Dominic Boceck
With slides from Marco Schäfer, Nicolas Brich and Sascha Patz

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Tutors for today and tomorrow

Mathias Witte Paz

- PhD Student at Integrative Transcriptomics (PI: Prof. K. Nieselt)
- Office: C304, Sand 14
- Interests:
  - (Ancient) genomics
  - Transcriptomics
  - Biol. Visualization

Dominic Boceck

- PhD Student at Institute of Medical Genetics and Applied Genomics (PI: Prof. S. Ossowski)
- Office: 1.101, Calwerstraße 7/8
- Interests:
  - Machine Learning
  - Cancer Genomics
  - Rare Diseases

Congratulations! You have learned a little bit of python, so that you could program your own python scripts. Now we dig a little deeper

You can get the code snippets from the github (https://github.com/BioInfPrep/python_basics). Since you have seen the basics of python already, you could follow using the code too. But only look at the solutions once you're done ;)

Feel always **free to stop us and ask questions** (or to complain if the pace is too fast)!

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 with files (open, write, close files)
6) Itertools and groupby and other related packages
7) Introduction to matplotlib and numpy
8) Flow Control
9) Best Practices

…

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) **Flow Control**
6) Import packages and work with files (open, write, close files)
7) Itertools and groupby and other related packages
8) Introduction to matplotlib and numpy
9) Best Practices

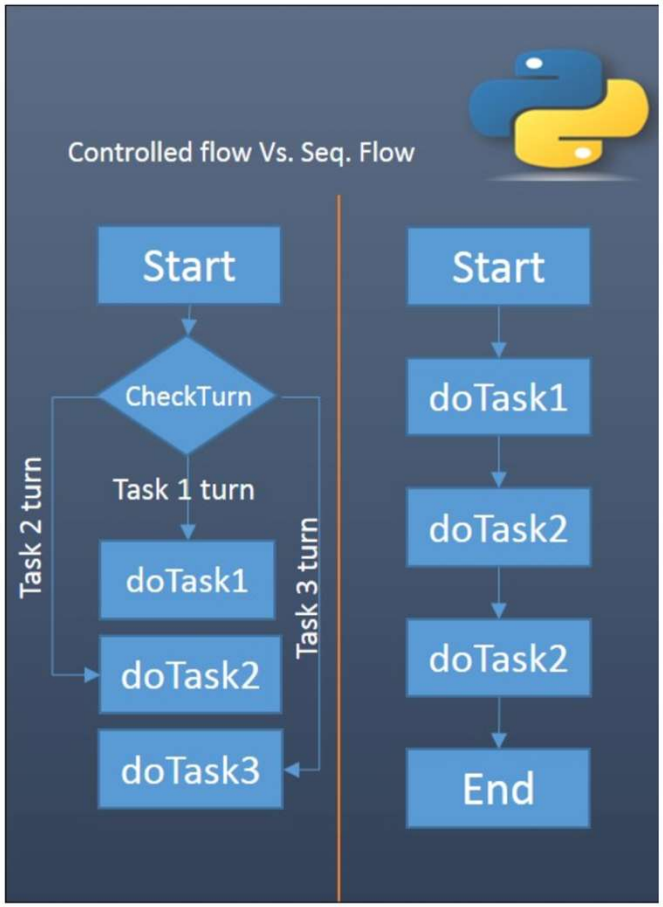
…

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.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Import Packages and Work on Files

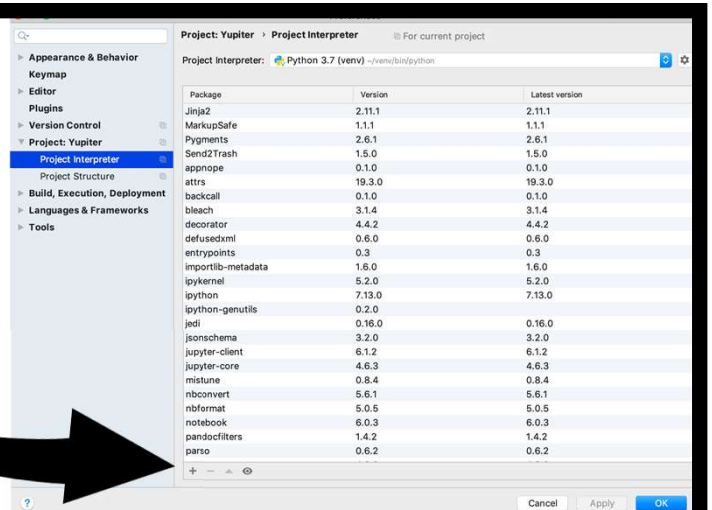
What is a package (module)?

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). Try installing matplotlib (we'll use this package later on)

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/subfolders 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
```

Explore the default argument and alternative arguments of listdir() function?

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

Hands on!

Iterate over the files/subdirs 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
```

Can you iterate only on the files in the current directory?
Hint: os.path.isfile() might help

How to Read and Write files?

```
# open file for writing
with open("new_file.txt", "w") as f:
  # only strings can be written to the file:
  for i in range(10):
    f.write(str(i) + "\n")
```

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
with open("new_file.txt", "r") as f:
  # get text from file:
  a = f.read()
  #a=f.readline()
  #a=f.readlines()
  print(a)
```

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()!

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

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', 'l', '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 key, group in groupby(my_list, keyfunc):
    print('starts with', key, '->', ' '.join(list(group)))
```

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

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', 'l', '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 comprehension.

WORK ON YOUR OWN !

Write a function to list all files (ignoring sub-folders) present in the given directory

The function should take a path to the directory as an argument – the default arg. is the current directory.

Print out should be formatted in the following way:

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

Your get_file_types function should return a dictionary with the structure type_of_file: list of corresponding files.

Do not forget to use the main() function!

Generate a "code skeleton" with the following:

click source code

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

def get_file_types (path_list):
    # fill here

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

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

Look up the convention of using a main function.

WORK ON YOUR OWN !

Write the output of the previous *"get_file_types"* function into a new file, called "my_dir_stats.txt"

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Introduction to matplotlib and numpy

The Matplotlib package …

import matplotlib

Type and run!

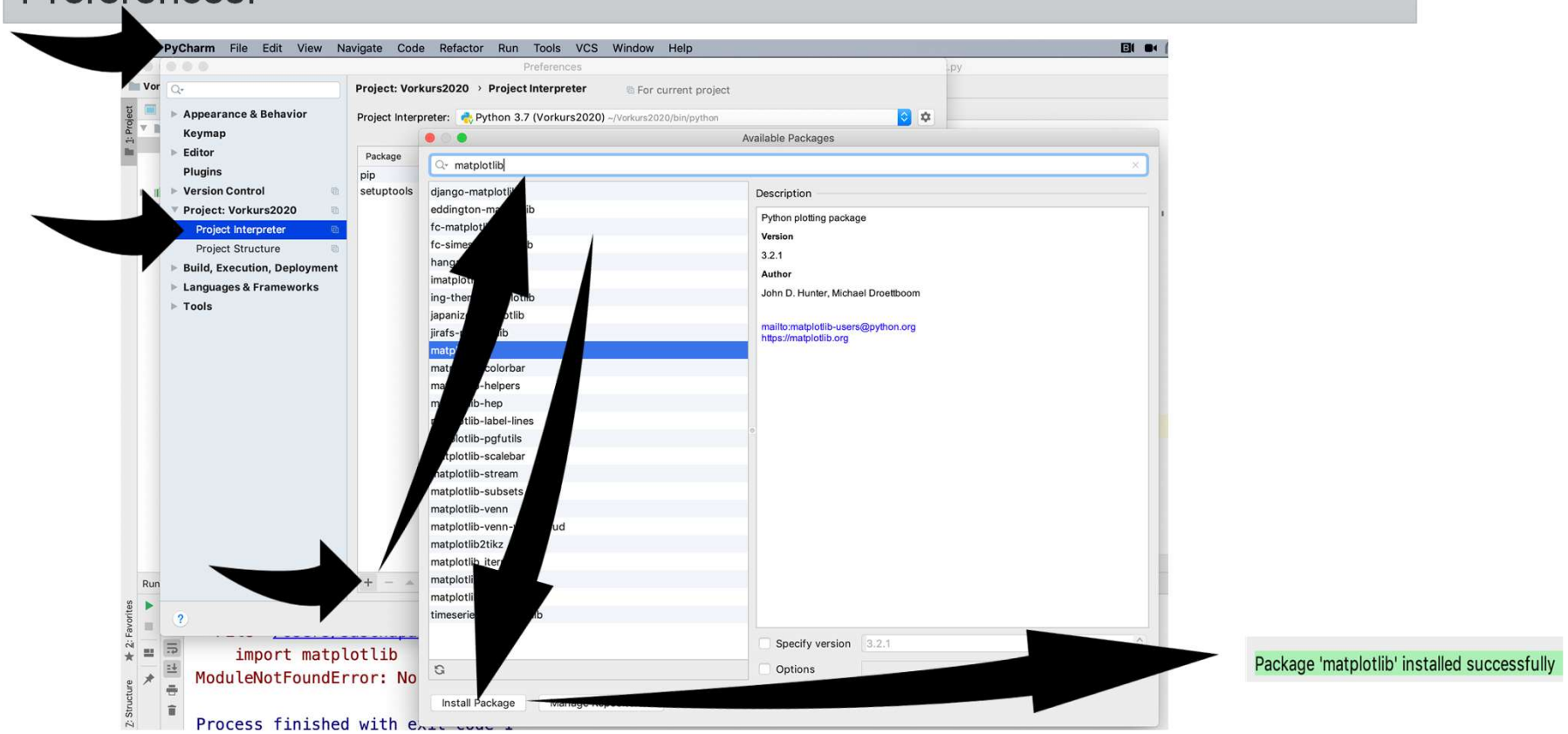
```
Traceback (most recent call last):
  File "/Users/saschapatz/PycharmProjects/Vorkurs2020/python_script.py", line 353, in <module>
    import matplotlib
ModuleNotFoundError: No module named 'matplotlib'

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.



Working on comma separated files (csv) …

Copy the iris.csv file (it is in the github link) to yoyr the current working directory.

```
# file handling
with open('iris.csv', 'r') as file_handle: # open file for reading ‘r’
    entire_file_text = file_handle.read()
    print(entire_file_text[:300])  # print only part of the file cave: Number indicates CHARS!!!

with open('iris.csv', 'r') as file_handle:  # open file for reading ‘r’
    file_lines = file_handle.readlines()  # returns a list
    print(' '.join(file_lines[:2]))  # print first 2 lines

    column_names = file_lines[0].split(';')  # get column names
    print(column_names)
```

What does string '\n' denote? How is it printed? And what is '\t'?

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')


# 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()
```

Import packaged and functions.

Define style.

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()
```

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()
```

Histogram

```
sepal_length = []
with open('iris.csv', 'r') as file_handle:  # open file for reading ‘r’
    file_lines = file_handle.readlines()
    for line in file_lines[1:]:  # iterate through lines in the file,
ommit headers
        split_line = line.split(',')  # split by colon
        if split_line[1]:
            # print(split_line[0])
            sepal_length_loop = float(split_line[0])
            sepal_length.append(sepal_length_loop)
plt.hist(sepal_length, bins=30)
plt.title("Histogram of sepal length")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Count")
plt.show()
```

WORK ON YOUR OWN !

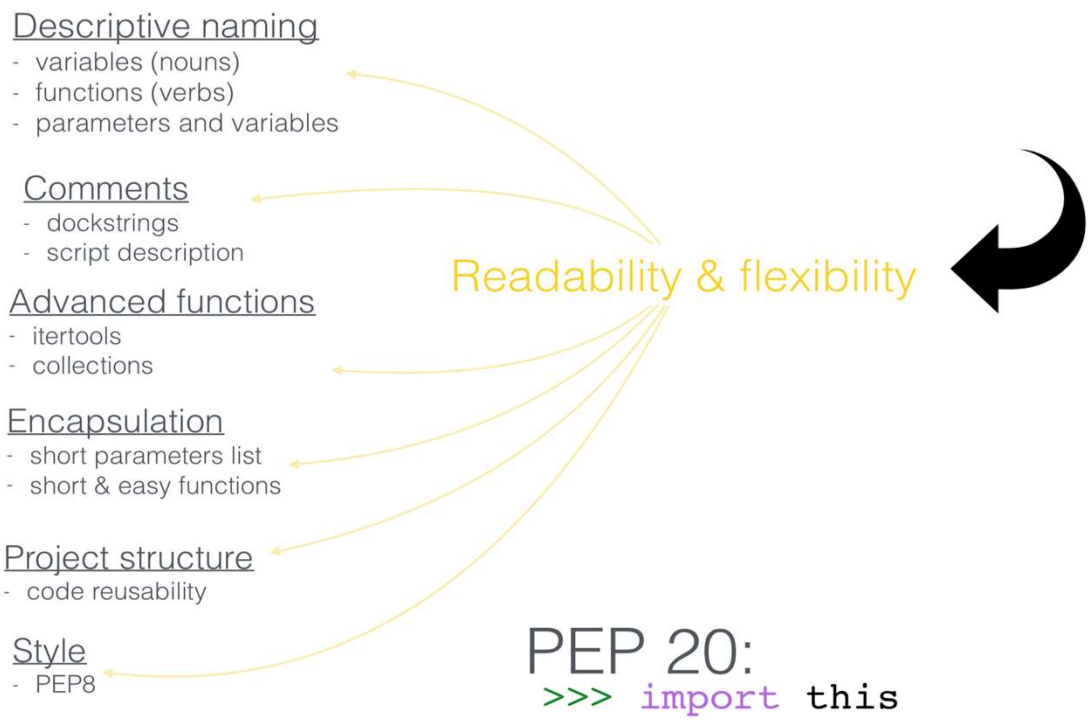
Make a boxplot of the columns sepal.length, sepal.width, petal.length, petal.width.

Make a scatterplot of the sepal.length and the petal length. Each dot should be colored differently depending on their variety.

How to be a good programmer?

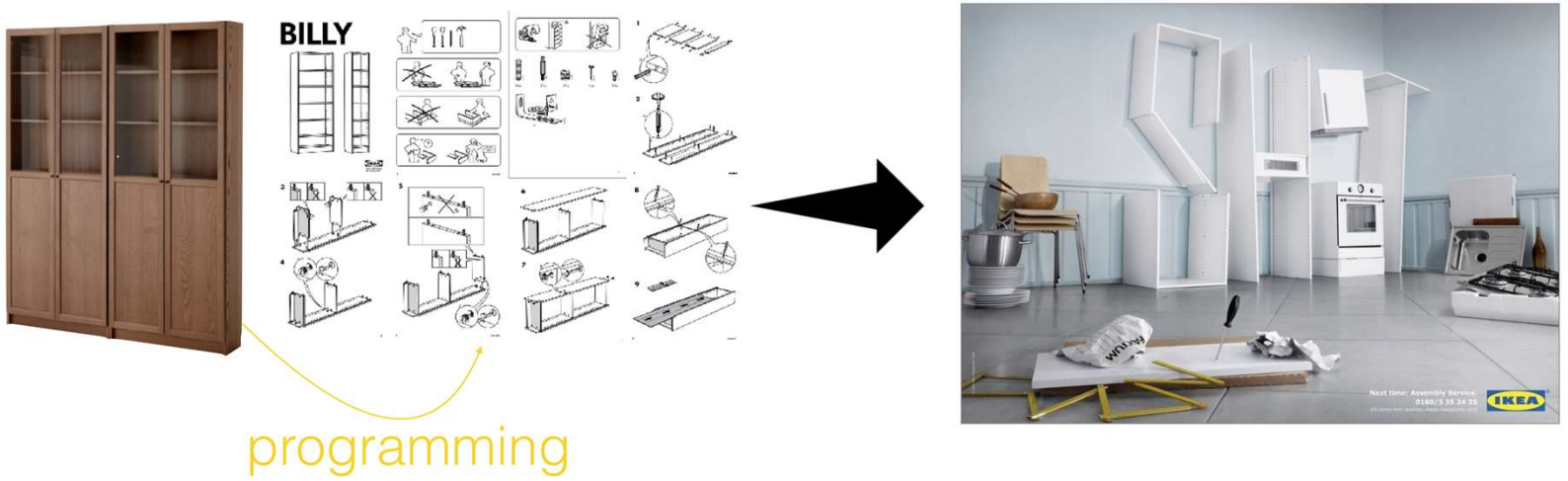
A well written program?

... everybody can read and understand it.



Ability to program?

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



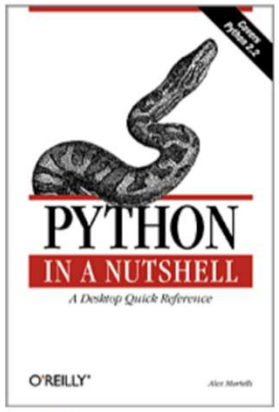
Best Practices

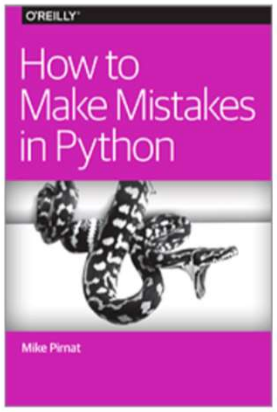
- Keep a well-documented code!
- Keep up with packages updates → Plan deprecations/ breaking changes
  - Use the appropriate update channels i.e. Conda OR PIP
  - “conda install …” and “pip install …”
- Keep up with core language updates

- Write reusable and readable code!

Keep up: books, docks, blogs, podcasts

used here:





more:

» PythonBooks

» Ten things ...bout Python » PythonBooks

There are a variety of books about Python. Here's a guide to them:

- » IntroductoryBooks (gentle overviews of the language)
- » AdvancedBooks (for when you don't want gentle)
- » ReferenceBooks (much information in a small space)
- » Specific applications:
  - » GameProgrammingBooks
  - » NetworkProgrammingBooks
  - » GuiBooks
  - » JythonBooks
  - » ScientificProgrammingBooks
  - » SystemAdministrationBooks
  - » WebProgrammingBooks
  - » WindowsBooks
  - » XmlBooks
  - » ZopeBooks

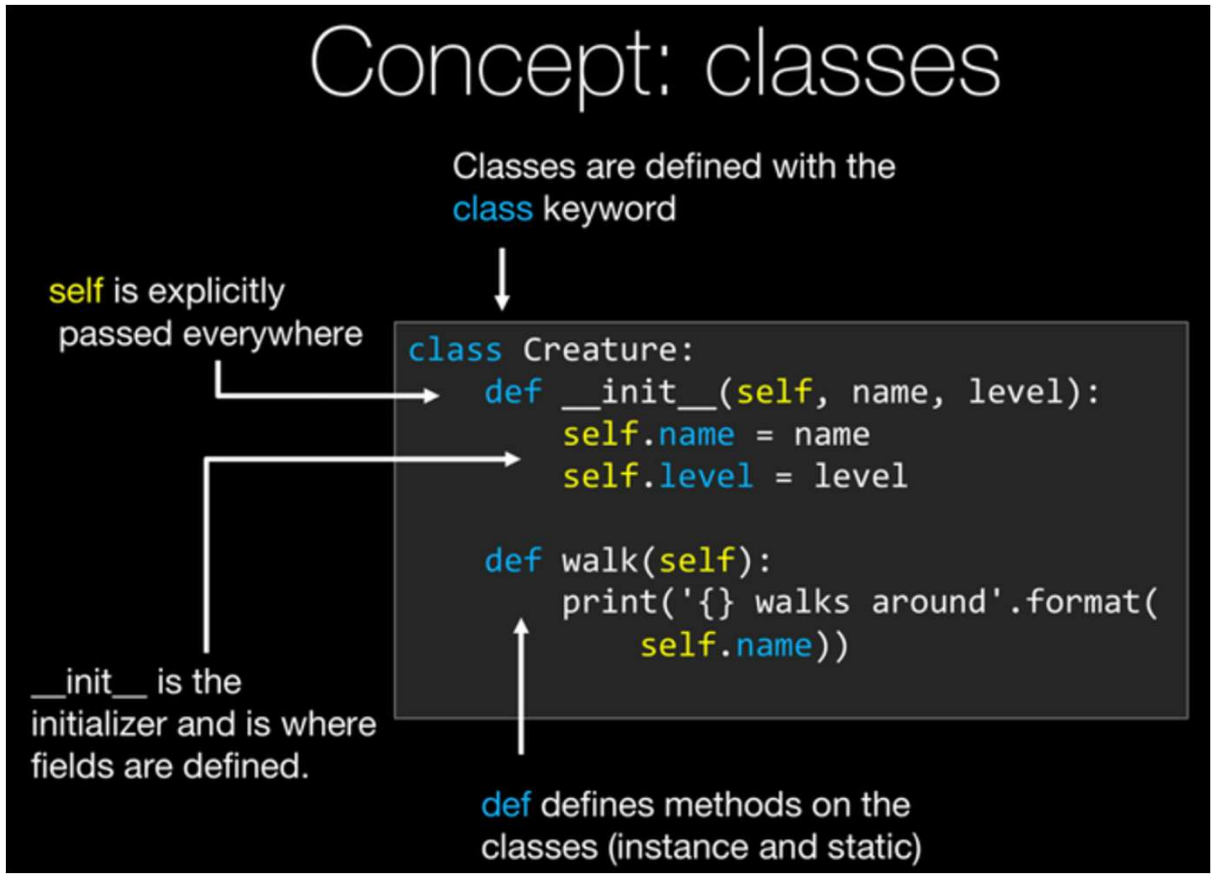
https://wiki.python.org/moin/PythonBooks

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 tomorrow!

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!