



PRINT:
Python foR
dummIes
iN quaranTine

ESO - Chile
April 27th-28th,
2020

R. Thomas, R. de Rosa, T. Berg, J. Hartke & C. Herenz

Program reminder

Today:

09-09:45: *Introduction to Python*

09:45-10:30: *Variable, Operation, built-in functions and errors*

10:30-11:00: *Working with strings [Robert]*

11:00-11:30: *Break*

11:30-12:00: *Loops and conditional [Johanna]*

12:00-12:45: *Data structures: list, tuples and dictionaries [Trystyn]*



light roast comics

Program reminder

Tomorrow:

09-09:45: Working with files [Robert]

09:45-10:30: Functions in Python [Christian]

10:30-11:00: Plotting with matplotlib [Johanna]

11:00-11:30: Break

11:30-12:00: Classes: Object oriented programming [Trystyn]

12:00-12:30 A quick tour of the Python standard Library [Romain]

12:30-12:45 Wrap-up [Romain]



light roast comics

Let's get started! What is Python?

No need to
compile

User-Friendly
→ Close to English

Interpreted, object-oriented, high-level language

The notion of object is
central in Python
(see Trystyn's talk
tomorrow)

Let's get started! What is Python?

Python was invented in the late 80s / Early 90s by Guido Van Rossum in the Netherland who called it after the *Monty Python*

Releases:

- First Version v0.9.0 released in 02/1991
- v1.0 released in 01/1994.
- v2.0 released in 10/2000 → v2.7 is still used by a lot of people
but discontinued
- v3.0 released in 12/2008 → We use v3.5+ it in the workshop
- v4.0 released in ??????????



Python is distributed under the General Public Licence (GPL)

What is Python? How popular?

Worldwide, Apr 2020 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	30.61 %	+3.9 %
2		Java	18.45 %	-1.9 %
3		Javascript	7.91 %	-0.4 %
4		C#	7.27 %	-0.0 %
5		PHP	6.07 %	-1.1 %
6		C/C++	5.76 %	-0.2 %
7		R	3.8 %	-0.2 %
8		Objective-C	2.4 %	-0.4 %
9		Swift	2.23 %	-0.2 %

Source: PYPL

Python is one of the most popular language in the world

→ Two important consequences

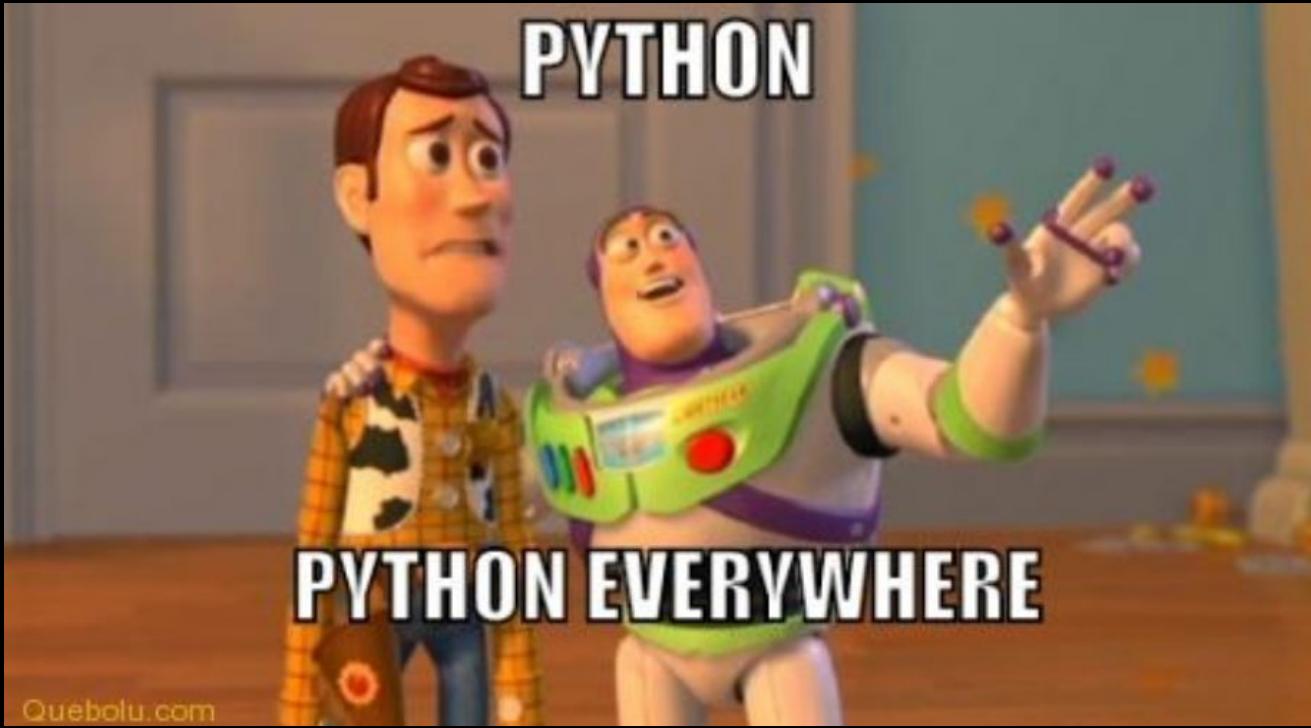
1- Huge community

→ You can find help everywhere

2-Countless pieces of codes/libraries

→ Someone already did what you want to do....

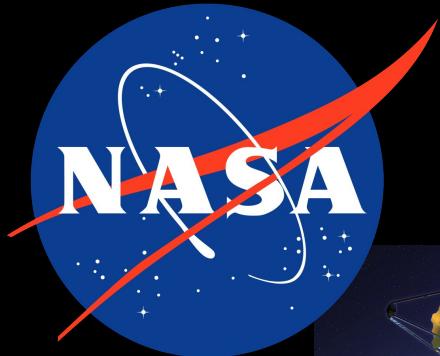
Where is it used?



Where is it used? Science!!!



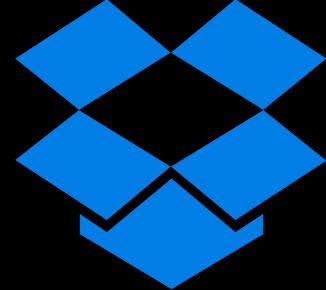
esa



Where is it used? Websites!



N



Quora



Learning Python...OK..But which one??

Python 2.X

Released in 2000

+A lot of material are still in python 2

-No more bug fixes or security update

-Support for a lot of libraries will stop soon

[End of Life: 2020](#)

Python 3.X

Release in 2008 with some compatibility problem

+Most of libraries are Py3.X compatible

+New version of libraries are always going to Py3

+This is the current state of the language

We use python 3.5+ for this camp

Python 2

Python 3

CIVIL WAR
CAPTAIN AMERICA
MAY 6, 2016

Learning Python...OK..But which one??

Python 2.X

Released in 2000

+A lot of material are still in python 2

-No more bug fixes or security update

-Support for a lot of libraries will stop soon

End of Life: 2020

Python 3.X

Release in 2008 with some compatibility problem

+Most of libraries are Py3.X compatible

+New version of libraries are always going to Py3

+This is the current state of the language

We use python 3.5+ for this camp

Python 2

Python 3

CAPTAIN AMERICA
CIVIL WAR

MAY 6, 2016

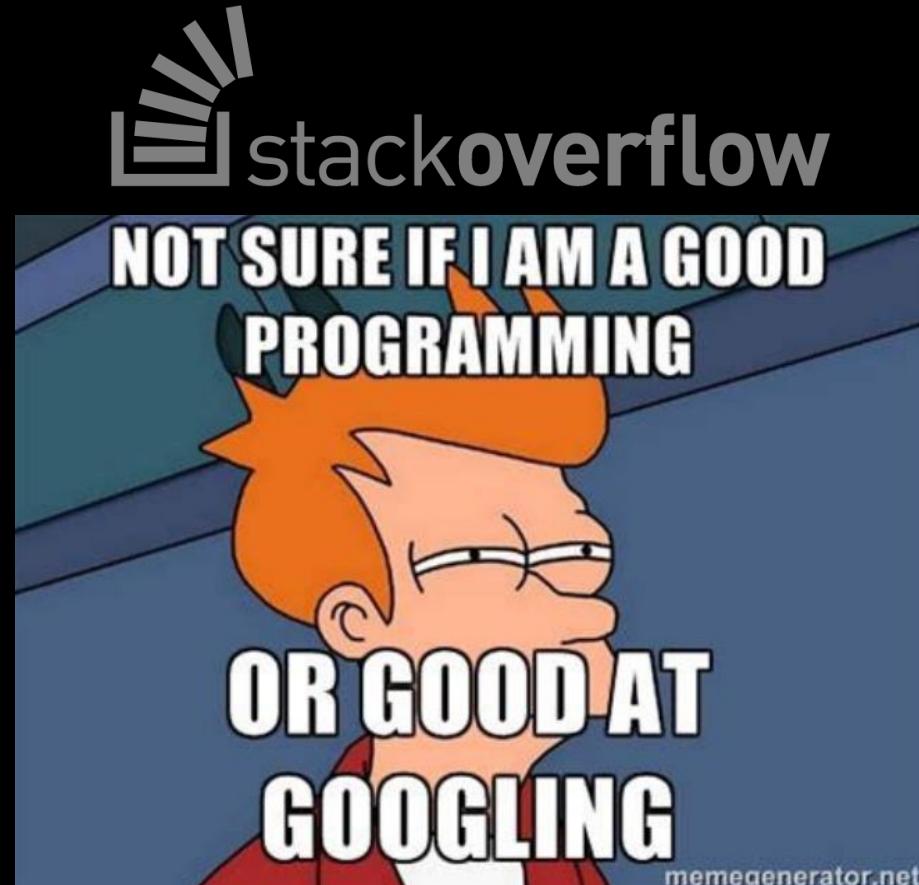
Learning Python? How long does it take?

FOREVER*

→ As any language you learn, it is long and requires practice!
And you never stop learning...

The best way: take a project and do it in Python

The second best way: take a broken code and debug it...



Where do we write
Python codes?

Where do I write python code?

You have multiple choices!

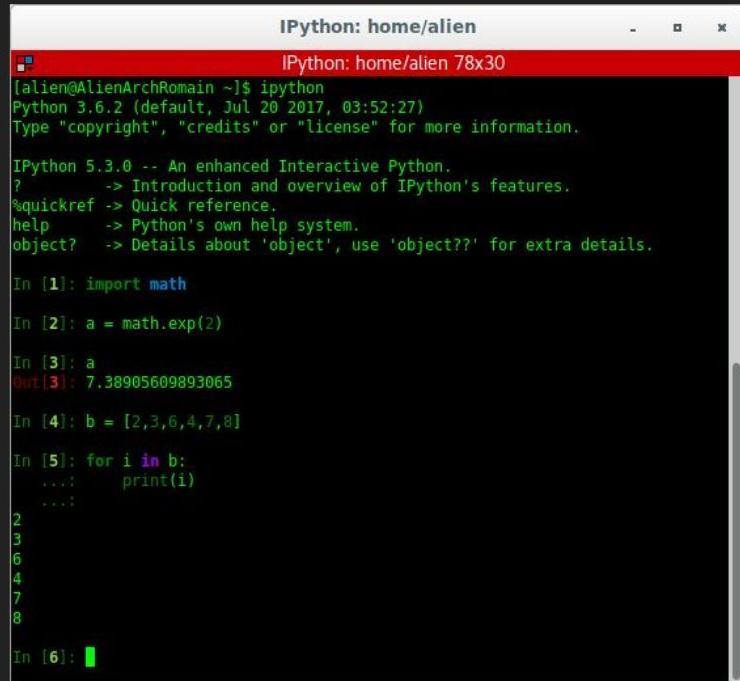
In the python interpreter
(with python or ipython)

-Easy to use (you write your line of code
and press enter)

- Nice environment to test

- You get the help of the functions
directly accessible

-Does not save your work in an
external file



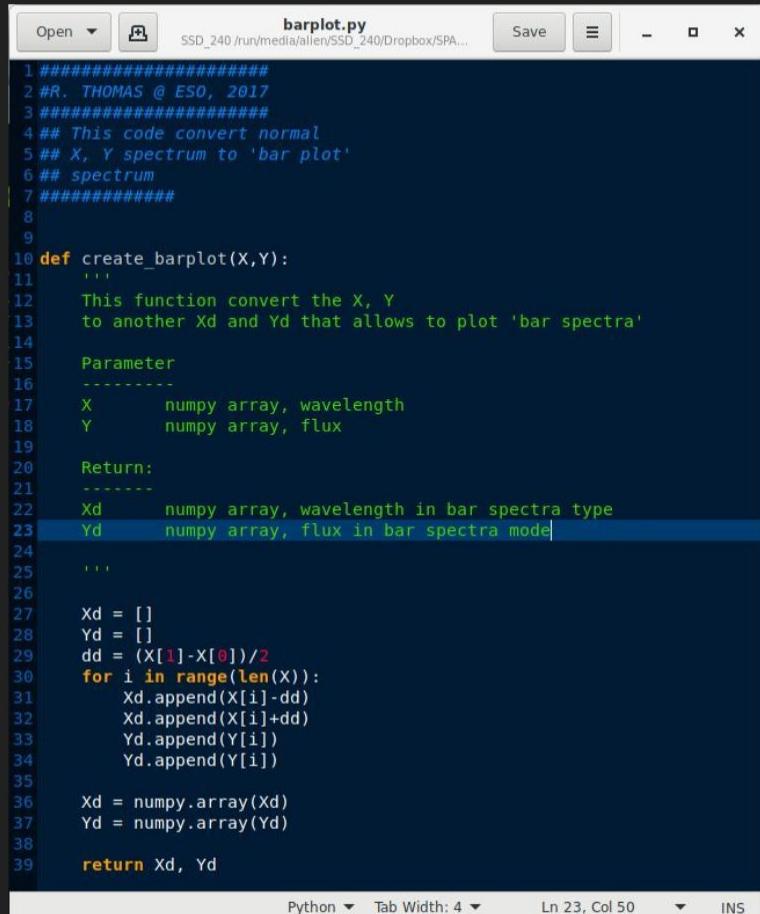
The screenshot shows a terminal window titled "IPython: home/alien" with a red header bar. The window displays Python 3.6.2 code and its output. At the top, it shows the command `[alien@AlienArchRomain ~]$ ipython`. Below that, it says "Python 3.6.2 (default, Jul 20 2017, 03:52:27)" and "Type "copyright", "credits" or "license" for more information." A series of help commands are listed: `?>` for introduction, `%quickref` for quick reference, `help` for Python's own help system, and `object?>` for details about the 'object' type. The main area shows code execution in In [1] through In [6]. In [1] imports the math module. In [2] calculates `a = math.exp(2)`. In [3] prints the value of `a`, which is `Out[3]: 7.38905609893065`. In [4] creates a list `b = [2,3,6,4,7,8]`. In [5] uses a for loop to print each element of `b`. The output of this loop is shown in In [5] and In [6], where the numbers 2, 3, 6, 4, 7, and 8 are printed sequentially.

Where do I write python code?

In an external file, with your favorite text editor or python IDE
→ You will create a *.py file

- You keep your code! (so you can share it!)
- You can create **reusable modules**
- Makes you organize your code and make it clearer
- Require a more organised work and thinking

You have multiple choices!



The screenshot shows a code editor window titled "barplot.py". The code is a Python script for creating bar plots from X and Y spectra. It includes comments explaining the function's purpose, parameters, and return values. The code uses numpy arrays for wavelength and flux, and it defines two new arrays, Xd and Yd, which represent the centered and decentered versions of the input arrays X and Y respectively. The script ends with a return statement for Xd and Yd.

```
1 #####
2 #R. THOMAS @ ESO, 2017
3 #####
4 ## This code convert normal
5 ## X, Y spectrum to 'bar plot'
6 ## spectrum
7 #####
8
9
10 def create_barplot(X,Y):
11     """
12         This function convert the X, Y
13         to another Xd and Yd that allows to plot 'bar spectra'
14
15     Parameter
16     -----
17     X      numpy array, wavelength
18     Y      numpy array, flux
19
20     Return:
21     -----
22     Xd    numpy array, wavelength in bar spectra type
23     Yd    numpy array, flux in bar spectra mode
24
25     """
26
27     Xd = []
28     Yd = []
29     dd = (X[1]-X[0])/2
30     for i in range(len(X)):
31         Xd.append(X[i]-dd)
32         Xd.append(X[i]+dd)
33         Yd.append(Y[i])
34         Yd.append(Y[i])
35
36     Xd = numpy.array(Xd)
37     Yd = numpy.array(Yd)
38
39     return Xd, Yd
```

Python Tab Width: 4 Ln 23, Col 50 INS

Where do I write python code?

You have multiple choices!

In a python notebook

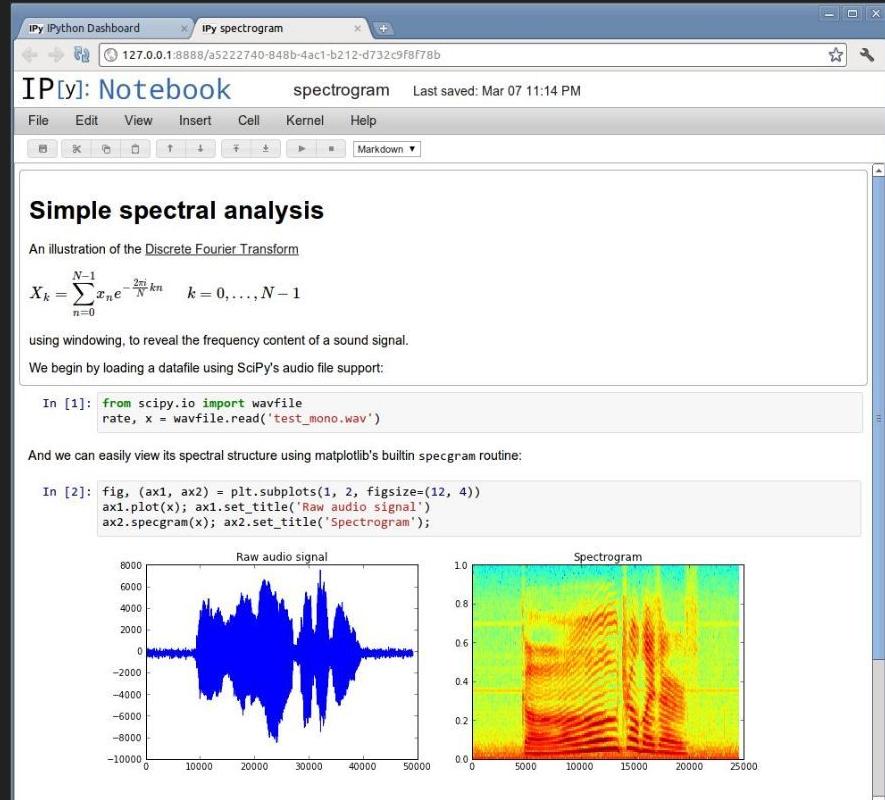
→ you will create a *.ipynb file

-Web interface

-It is an *advanced* ipython

-You keep your code! (so you can share it!)

-Especially suitable for exercices and demonstration



The Python Standard Library:

Natively available Python modules
→ They come with your installation

```
...  
future    main      dummy_thread _thread    abc     aifc    argparse      array    ast      asynchat    asyncio    asyncore    atexit    audioop    base64    bdb     binascii t  
inhex    bisect   builtins    bz2      cProfile    calendar    cgi      cgihttp    chunk    cmath    code    codecs    collections    collections.abc    colorsys    compileall    cor  
current.futures configparser contextlib    copy      copyreg    crypt    csv      ctypes    curses    curses.ascii    curses.panel    curses.textpad    datetime    dbm     dbm.dumb    dbm.gnu    d  
bm.ndbm    decimal  dirlib     dis      distutils    distutils.archive_util    distutils.bccpcompiler    distutils.distutils.ccompiler    distutils.cmd    distutils.command    distutils.command.bdist    d  
istutils.command.bdist_dumb    distutils.command.bdist_msi    distutils.command.bdist_packager    distutils.command.bdist_rpm    distutils.command.bdist_winst    distutils.command.build    d  
istutils.command.build_clib    distutils.command.build_ext    distutils.command.build_py    distutils.command.build_scripts    distutils.command.check    distutils.command.clean    distutils.c  
ommand.config    distutils.command.install    distutils.command.install_data    distutils.command.install_headers    distutils.command.install_lib    distutils.command.install_scripts    distutils.command.  
register    distutils.command.sdist    distutils.core    distutils.cygwincompiler    distutils.debug    distutils.dep_util    distutils.dir_util    distutils.dist    distutils.errors    d  
tutils.extension    distutils.fancy_getopt    distutils.file_util    distutils.filelist    distutils.log    distutils.msvccompiler    distutils.spawn    distutils.sysconfig    distutils.text_file    d  
tutils.unixcompiler    distutils.util    distutils.version    doctest    dummy_threading    email    email.charset    email.contentmanager    email.encoders    email.errors    email.generator    e  
ll.header    email.headerregistry    email.iterators    email.message    email.mime    email.parser    email.policy    email.utils    encodings.idna    encodings.mbcsc    encodings.utf_8_sig    ensurepip e  
num    errors    faulthandler    fcntl    filecmp    fileinput    fnmatch    formatter    fpectl    fractions    ftplib    functools    gc    getopt    getpass    gettext    glob    grp  
azip    hashlib    heapq    hmac    html    html.entities    html.parser    http    http.client    http.cookiejar    http.cookies    http.server    imaplib    imghdr    imp    importlib    imp  
ortlib.abc    importlib.machinery    importlib.util    inspect    io    ipaddress    itertools    json    json.tool    keyword    lib2to3    linecache    locale    logging    log  
ging.config    logging.handlers    lzma    macpath    mailbox    mailcap    marshal    math    mimetypes    mmap    modulefinder    msilib    msvcr    multiprocessing    multiproces  
sing.connection    multiprocessing.dummy    multiprocessing.managers    multiprocessing.pool    multiprocessing.sharedctypes    netrc    nis    nntplib    numbers    operator    optparse    o  
s    os.path    ossaudiodev    parser    pathlib    pdb    pickle    pickletools    pipes    pkgutil    platform    plistlib    poplib    posix    pprint    profile    pstats    pty    pwc  
py_compile    pycldr    pydoc    queue    quopri    random    re    readline    reprlib    resource    rlcompleter    runv    sched    secrets    select    selectors    shelve    shlex    sh  
til    signal    site    smtplib    smtplib    tarfile    socket    socketserver    spwd    sqlite3    ssl    stat    statistics    string    stringprep    struct    subprocess    sunau    symbol    symtable s  
ys    sysconfig    syslog    taonanny    telnetlib    tempfile    termios    test    test.support    textwrap    threading    time    timeit    tkinter    tkinter.sci  
oledtext    tkinter.tix    tkinter.ttk    token    tokenize    trace    traceback    tracemalloc    tty    turtle    tutledemo    types    typing    unicodedata    unittest    unittest.mock    u  
lib    urllib.error    urllib.parse    urllib.request    urllib.response    urllib.robotparser    uu    uuid    venv    warnings    wave    weakref    webbrowser    winreg    winsound    wsg  
iref    wsgiref.handlers    wsgiref.headers    wsgiref.simple_server    wsgiref.util    wsgiref.validate    xdrlib    xml    xml.dom    xml.dom.minidom    xml.dom.pulldom    xml.etree.E  
lementTree    xml.parsers.expat    xml.parsers.expat.errors    xml.parsers.expat.model    xml.sax    xml.sax.handler    xml.sax.saxutils    xml.sax.xmlreader    xmlrpc.client    xm  
rpc.server    zipapp    zipfile    zipimport    zlib
```

Adding extra packages:

From the Python Package Index [PyPi]:

Contains more than 100,000 packages

→ Four useful commands:

- `pip search + modul`

will display a list of package with the word you are giving

- `pip install + modul [+ '--user']`

will install the package in your python distribution

- `pip uninstall + modul`

will remove the package from your distribution

- `pip freeze`

will list all the pip-installed packages in your distribution with the version number

Writing and commenting and documenting code:

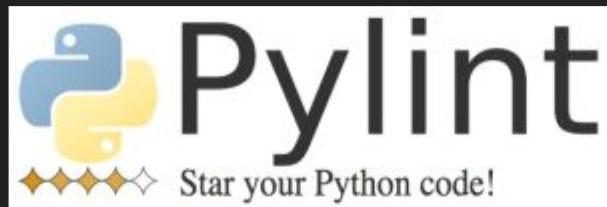
Python comes with rules about how to write a good code. They are described in the PEP8 and PEP257 documents (*Python Enhancement Proposal*). If you ignore them your code will work (don't worry) but following them will make your code more readable.

- Python is indentation sensitive
→ one indentation = 4 spaces
- Imports
→ one import per line, no wildcard imports
- Avoid too long lines
→ (max 80-100 characters / line)
- **A code is read much more often than it is written** - Guido van Rossum
→ Comment your code!!!!!!

- docsstrings: A format to document fct/classes/modules. Ex: numpydoc

```
1 def function(param1, param2):
2     """Example function.
3
4     The return type must be duplicated in the docstring to comply
5     with the NumPy docstring style.
6
7     Parameters
8     -----
9         param1
10            type, short description.
11        param2
12            type, short description.
13
14     Returns
15     -----
16        bool
17            True if successful, False otherwise.
```

Useful tools and resources



supports a number of features, from coding standards to error detection, and it also helps with refactoring (by detecting duplicated code).



If you want to look
at advanced codes
you can go there
→



Spyder & PyCharm are widely used development environment for python



Something you do not understand?
Something you do not know how to do?

I will not propose books or websites here with python lectures.

You can just type 'python lectures' , 'python for beginners' in google and you will find thousands of solutions (do not forget youtube, there are plenty of videos with python classes)

Enjoy these 2 ½ half days of Python!!!

And huge thanks to the speakers:

Robert, Johanna, Trystyn and Christian