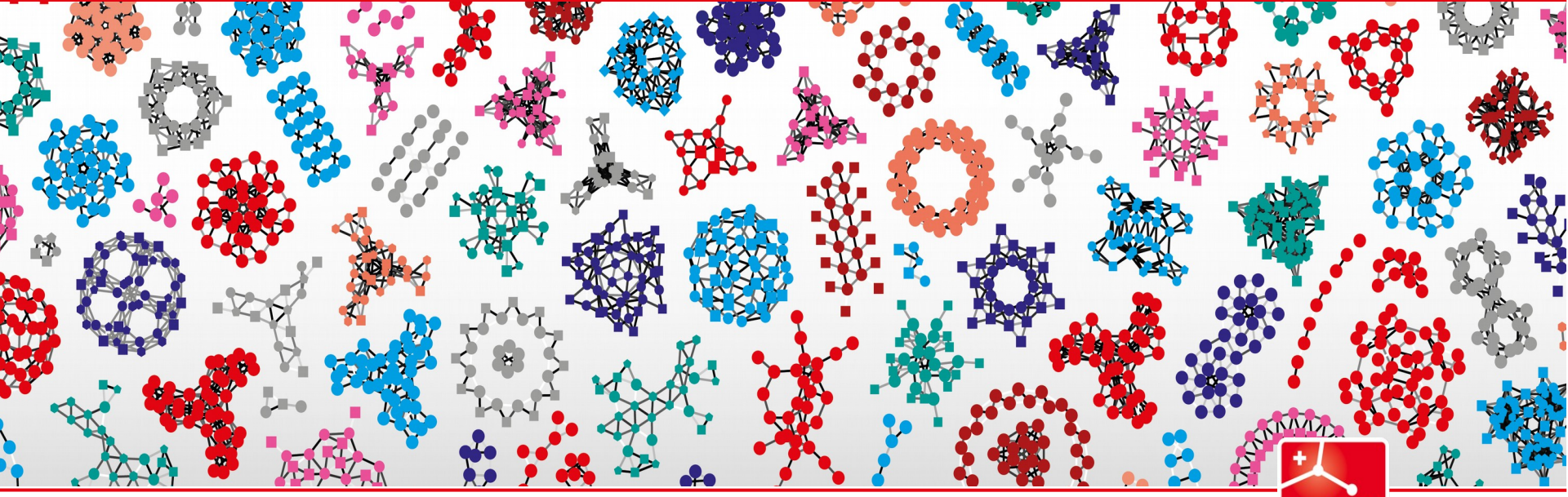


First steps with Python in life sciences

Wandrille Duchemin
Robin Engler
Rocio Rama Ballesteros

Swiss Institute of Bioinformatics

Course schedule

Day 1

morning:
- Introduction to Jupyter Notebook
- Python basics: variables, functions and object types.

afternoon:
- Code flow: conditional statements (if… else), loops (while…, for …) and functions.

Day 2

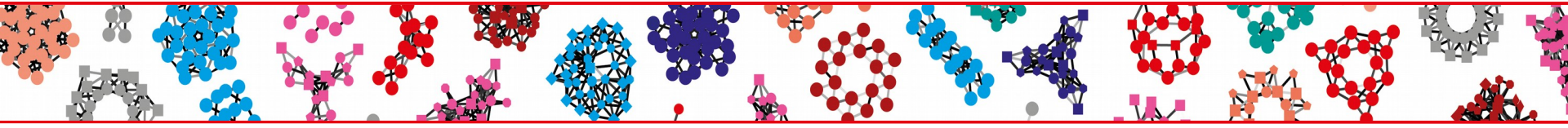
morning:
- Code flow: conditional statements (if… else), loops (while…, for …) and functions.

afternoon:
- Reading / writing files.
- Python modules: import and re-use existing code.

Day 3

Additional modules :
- Biopython
- Numpy
- Scipy
- Matplotlib
- Pandas

Course introduction overview



01 — What is python? Why use it?

02 — Computer ressources and how (not) to use them

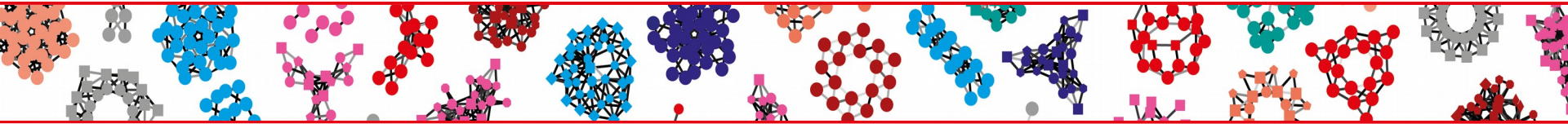
03 — FAIR practices in coding

04 — Practice, practice and more practice

But first - getting to know you

- Have you programmed before ? Which language ?
- Any experience with command line ?
- Why do you want to program :
  - to analyze data ?
  - to create scripts that serve as *glue* in my pipeline ?
  - to implement my cool new model ?
  - to become one of the cool kids ?
  - to have something to do on my Sundays ?

Course introduction overview



01 — **What is python ? Why use it ?**

02 — **Computer ressources and how (not) to use them**

03 — **FAIR practices in coding**

04 — **Practice, practice and more practice**

What is Python ?

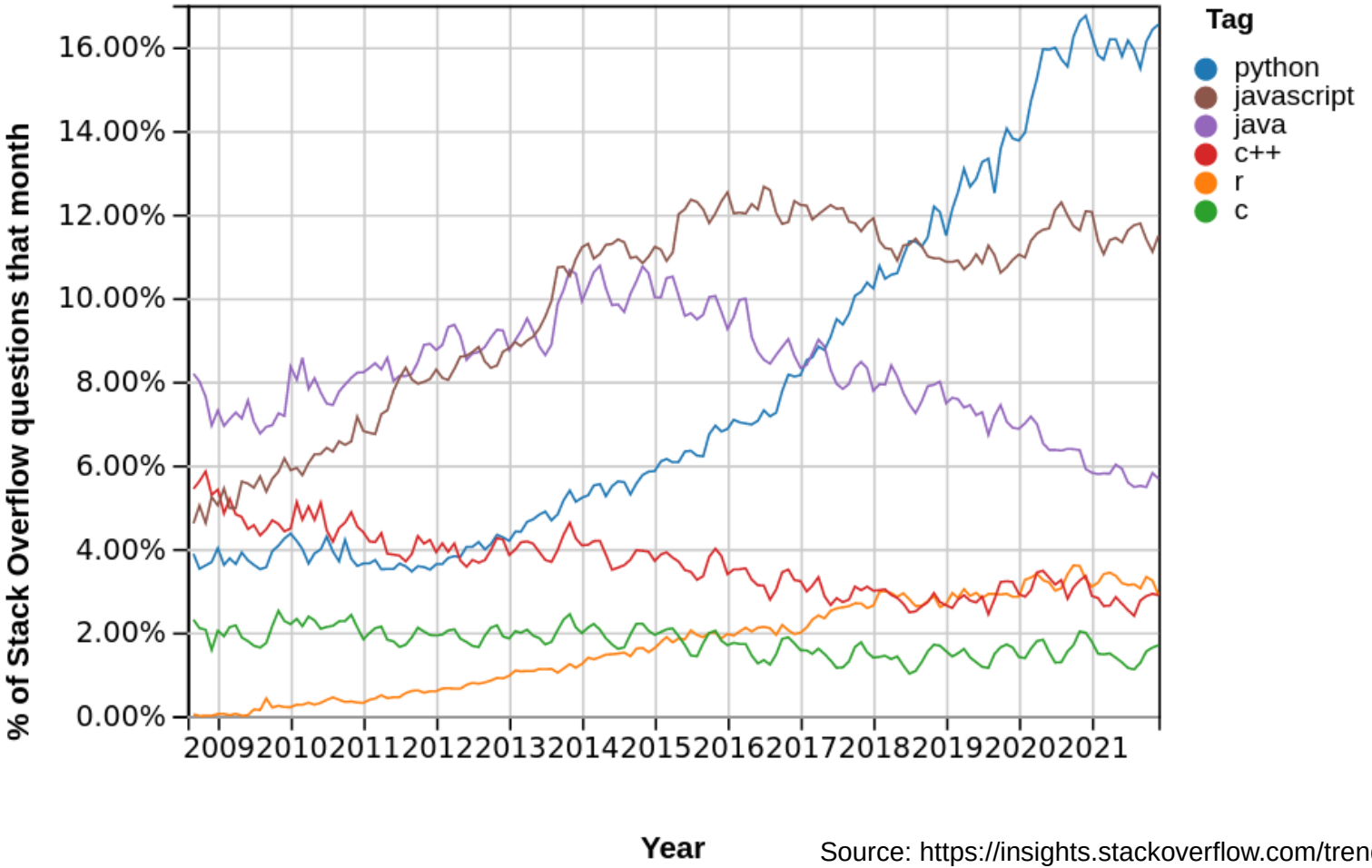
"Python is an **interpreted**, **high-level**, **general-purpose** programming language." wikipedia

- **Interpreted**: no compilation of the program prior to execution
  - + platform independence (portability), dynamic typing
  - - usually slower, buggy program may still run
- **High level**: abstracted from details of the machine (e.g. memory management)
  - + focus on the application itself
  - - possibly counter-intuitive behaviors (e.g. mutable vs. immutable objects).
- **General-purpose**: not domain specific, can be used in a broad range of software application
  - + wide user base, usable for any purpose
  - - core language fairly simple → modules for domain specific uses

Python - a brief history

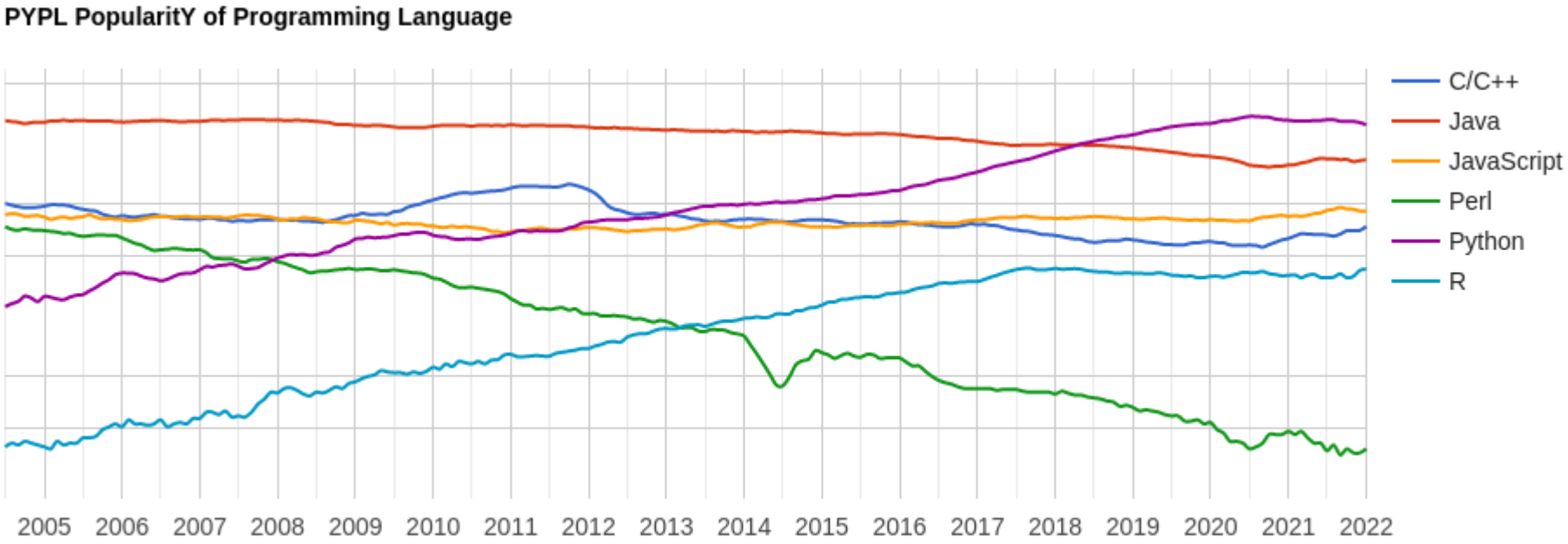
- **1991**: First version of Python (0.9.0) publicly released by Guido van Rossum.
  - Its name is a tribute to the British comedy group “Monty Python”, it later adopted the two snakes as logo symbol.
- **2000**: Python 2.0
- **2008**: Python 3.0 (backward incompatible with python2)
- ~~**2015**~~ **2020** : end of Python 2.7 support
- **Current version** : 3.8 (as of February 2020)

Python - is it used ?



Source: https://insights.stackoverflow.com/trends?tags=python%2Cr

Python - is it used ?



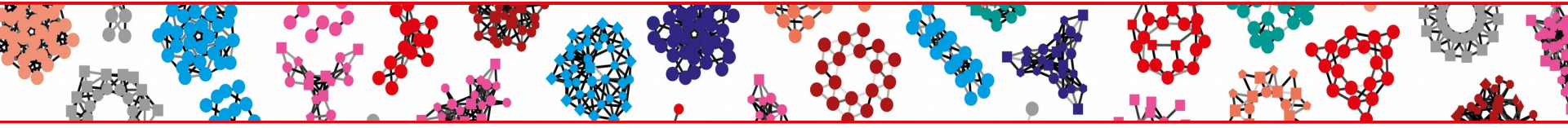
Source: http://pypl.github.io/PYPL.html

Python – sum up

“Python is an interpreted, high-level, general-purpose programming language.” wikipedia

- Easy to learn - you will experience that first hand !
- Portable: coding for Windows, MacOS or Linux is (almost) the same.
- Broad range of applications: can be used for anything !
- Widely used, including in science application:
  - Huge community to get help/tutorials.
  - Huge number of modules for domain specific applications.

Overview



01 What is python ? Why use it ?

02 Computer ressources and how (not) to use them

03 FAIR practices in coding

04 Practice, practice and more practice

A computer’s ressources

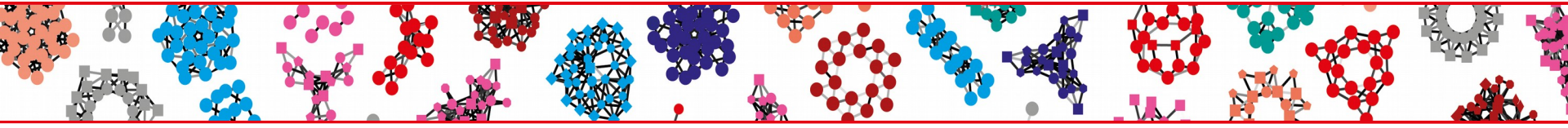
- **CPU**: main computing unit. Nowadays **multi-core** and **multi-threaded**
- **GPU**: graphical processor: like CPU but with **lots of slower cores**
- **RAM**: ‘short term memory’ of the computer. Fast read/write access.
- **Hard disk**: ‘long term memory’ of the computer. Slow access.
    - HDD : older, ‘cheap’, slow
    - SSD : newer, ‘expensive’, faster
- … external? file/computation accessed through the network?

A computer’s ressources

Courtesy of R.Cabezon and M.Jacquot

Event	Latency	Scaled
1 CPU cycle	0.3 ns	1 s
Level 1 cache access	0.9 ns	3 s
Level 2 cache access	2.8 ns	9 s
Level 3 cache access	12.9 ns	43 s
Main memory access (DRAM, from CPU)	120 ns	6 min
Solid-state disk I/O (flash memory)	50–150 µs	2–6 days
Rotational disk I/O	1–10 ms	1–12 months
Internet: San Francisco to New York	40 ms	4 years
Internet: San Francisco to United Kingdom	81 ms	8 years
Internet: San Francisco to Australia	183 ms	19 years
TCP packet retransmit	1–3 s	105–317 years
OS virtualization system reboot	4 s	423 years
SCSI command time-out	30 s	3 millennia
Hardware (HW) virtualization system reboot	40 s	4 millennia
Physical system reboot	5 m	32 millennia

Overview



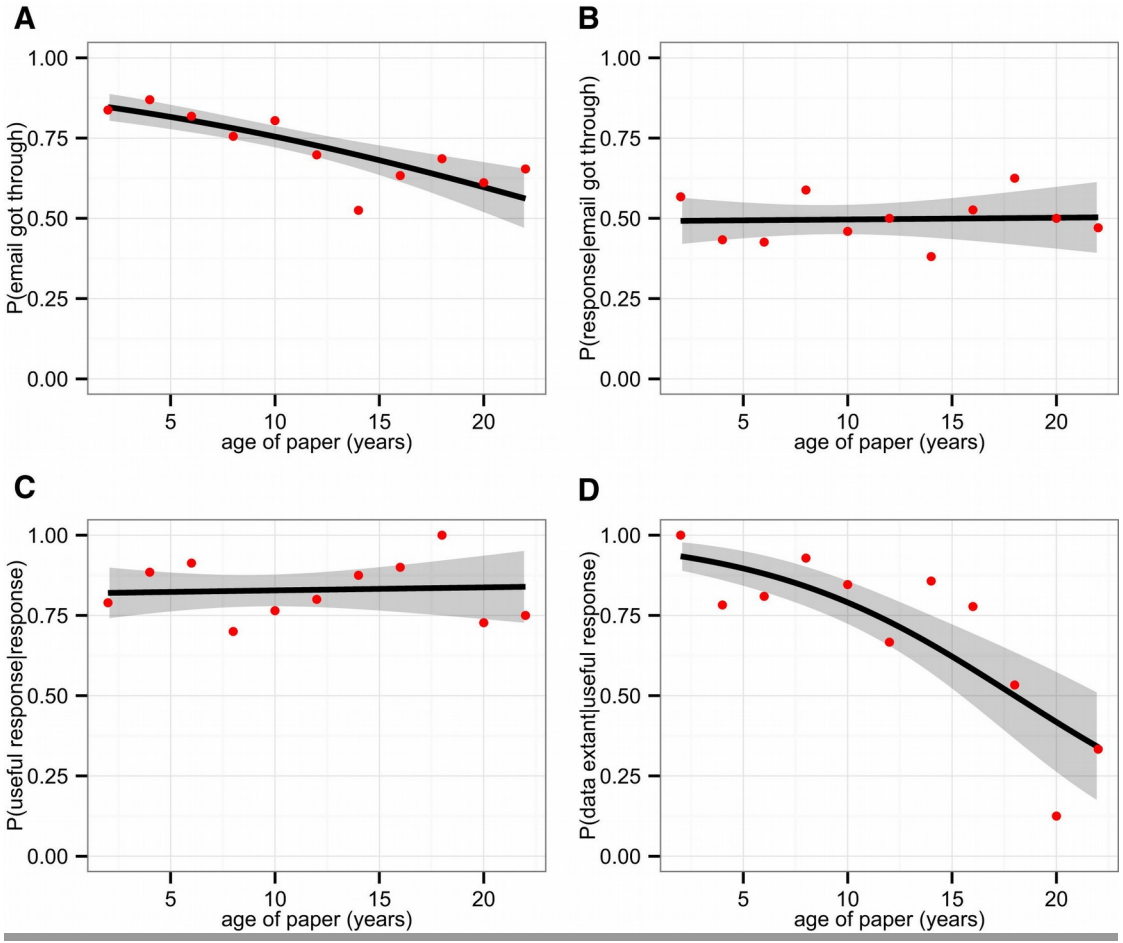
01 What is python ? Why use it ?

02 Computer ressources and how (not) to use them

03 FAIR practices in coding

04 Practice, practice and more practice

The reproducibility crisis



Survey of 516 studies:

- - 17% data availability per year.
- Only 19% retrieval rate after 10 year...

Vines et al. (2014) Curr. Biol. doi.org/10.1016/j.cub.2013.11.014

The **FAIR** guiding principles

Findable: Metadata and data should be easy to find for both humans and computers (unique global identifier, rich description, machine readable and searchable).

Accessible: The data can be retrieved using a standard communication protocol (e.g. https or sftp). Where needed, authentication and authorization procedures are available and documented.

Interoperable: the (meta)data should be based on standardized vocabulary and ontologies (categories and their relations), so that it integrates with existing applications and workflows.

Reusable: Metadata and data should be well described so that data can be replicated and/or combined in different research settings (rich metadata, clear license term, origin of data, data meets domain-relevant community standards).

M.D.Wilkinson, et al. The FAIR Guiding Principles for scientific data management and stewardship. Sci Data. 2016;3:160018. doi:10.1038/sdata.2016.18. https://www.go-fair.org

FAIR applied to code

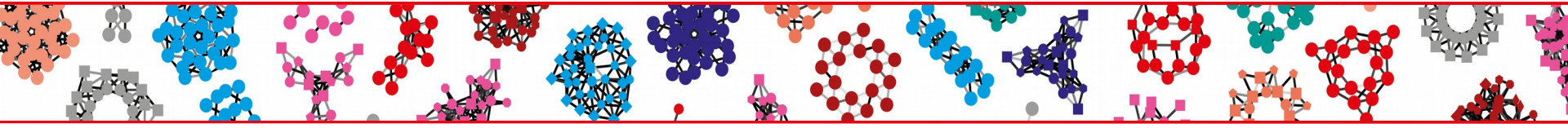
- Write code that also acts as documentation, and clearly communicates the analysis.
- Apply the standard you would expect of a ‘wet-lab’ protocol.
- Will a reasonably competent colleague understand your code ?
- Will you understand your code in 6 month ??

To achieve this, you should:

- Comment as much as needed.
- Use explicit names when naming things (variables, functions, classes).
- Possibly use a support that allows to easily mix code and text, e.g. Jupyter notebook or Jupyter-lab.
- Also, you can look at:

  Schwen LO, Rueschenbaum S (2018) Ten quick tips for getting the most scientific value out of numerical data. PLoS Comput Biol 14(10): e1006141. https://doi.org/10.1371/journal.pcbi.1006141.

Overview



01 What is python ? Why use it ?

02 Computer ressources and how (not) to use them

03 FAIR practices in coding

04 **Practice, practice and more practice**

Step 1: how do I use python ?

Three main modes of interaction:

- Interactive console.
- Python code file (.py files).
- Jupyter Notebook (.ipynb files).
  - Because of its ability to mix nicely formatted text (Markdown) with code, Jupyter Notebook / Jupyter Lab is well suited for data analysis.
  - Jupyter Notebook will be used for this course.

Python – using the console

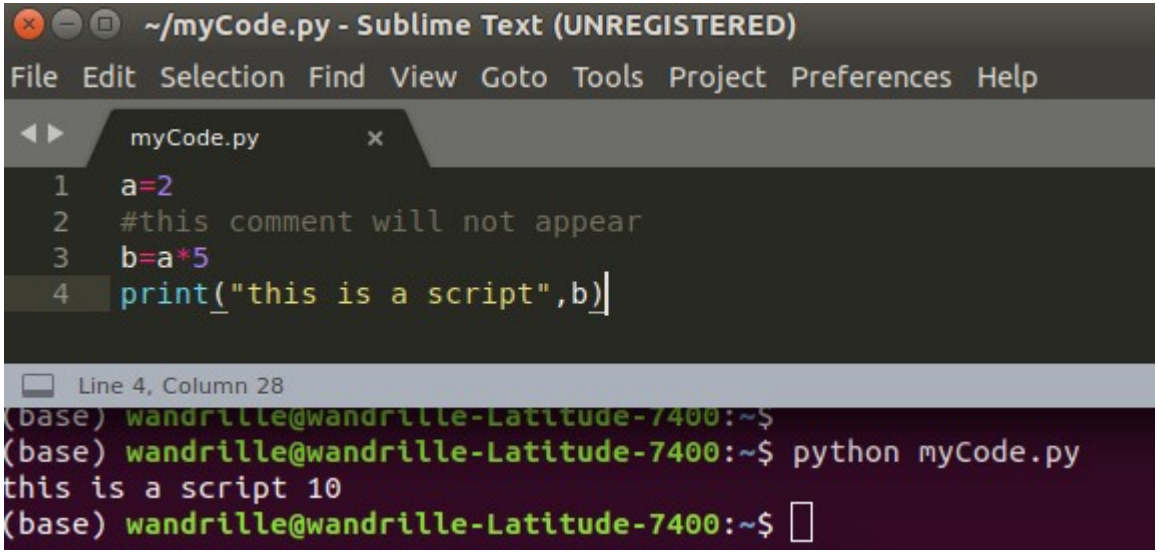
```
(base) wandrille@wandrille-Latitude-7400:~$ python
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> b=4
>>> a*b + 7
19
>>> print("this is an interactive console")
this is an interactive console
>>>
```

- **Interactive** : the code is executed as you press ‘Enter’
- Great for quickly **testing** things out.
- But, you keep **no trace** of your workflow/environment.

Only use it to test little bits of code

Python – writing a code in a .py file

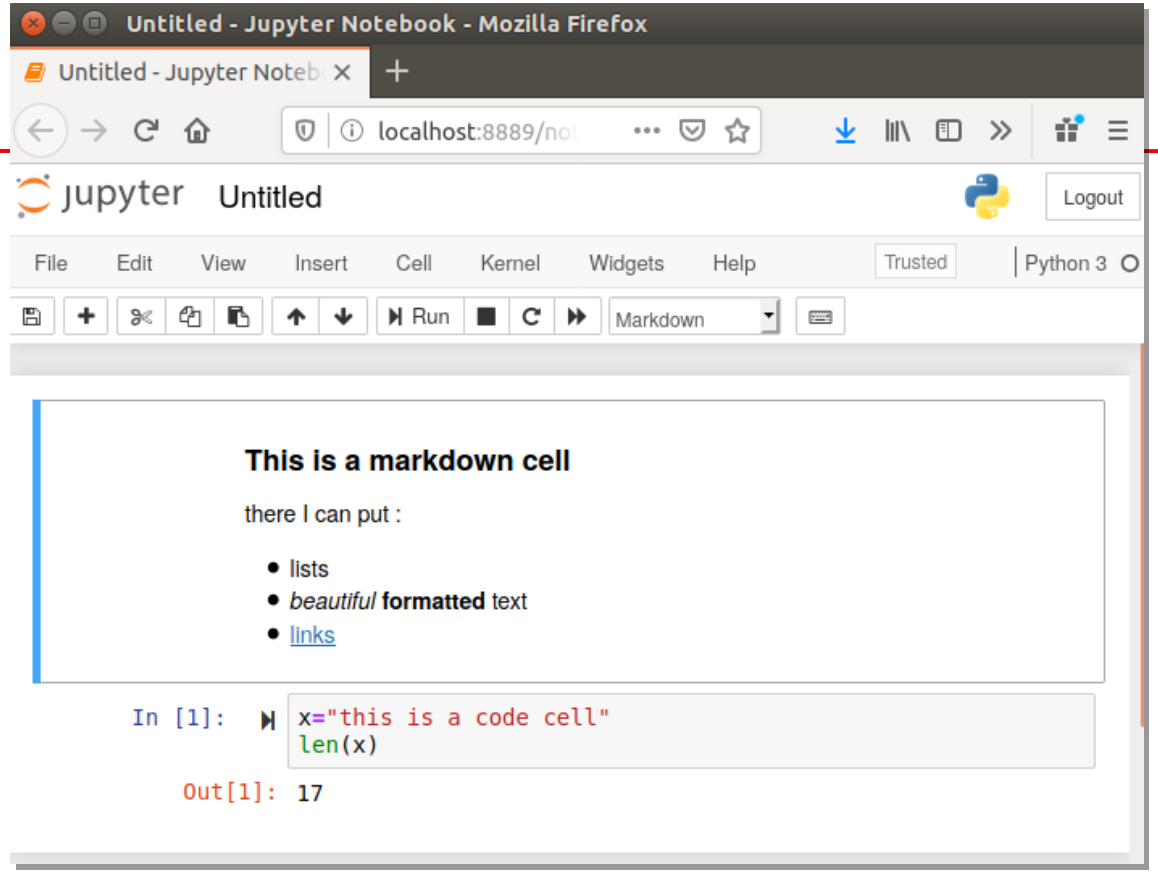
- Write a script in a .py text file, then execute it.
- Main way python code is shared.
- Ideal for standalone programs and modules.
- Code and results are kept separate (may be a good or a bad thing).



Good for general purpose coding, “operational script”.

Jupyter notebook / Jupyter Lab

- Browser based interface (but runs locally on your machine).
- Interlace Markdown and code ‘cells’.
- Execute code cell by cell.
- **Commentary, code, and results together in the same file**.
- Visually pleasant and fairly ergonomic.



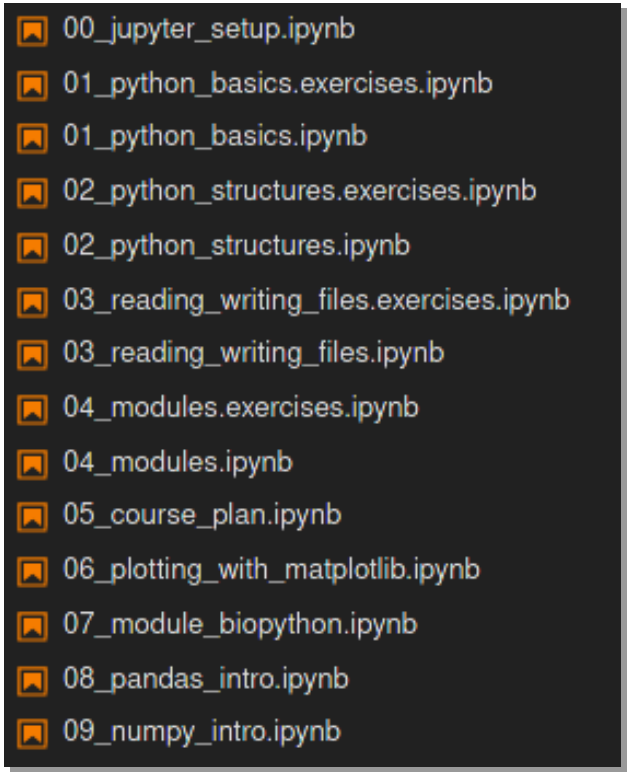
The ability to have documentation, code and results in a single file makes it ideal for **data analysis**: helps reproducibility of results.

Jupyter Notebook will be used in this course.

Step 2: your turn to work

How this course will be taught:

- Each lesson has 2 associated notebooks (.ipynb):
  - Theory and examples (*.ipynb).
  - Exercises (*.exercises.ipynb).
- Each lesson will be introduced by going through (some of) the theory together.
- Time is allocated for you to try to do the exercises on your own, before we correct them together.
- Feel free to ask questions at anytime.



PS : don’t despair - a certain amount trial and error is normal (and somewhat necessary) when learning a language!