

Swiss Institute of  
Bioinformatics

# First steps with Python in life sciences

Wandrille Duchemin, Robin Engler, and Bulak Arpat

# General information

---

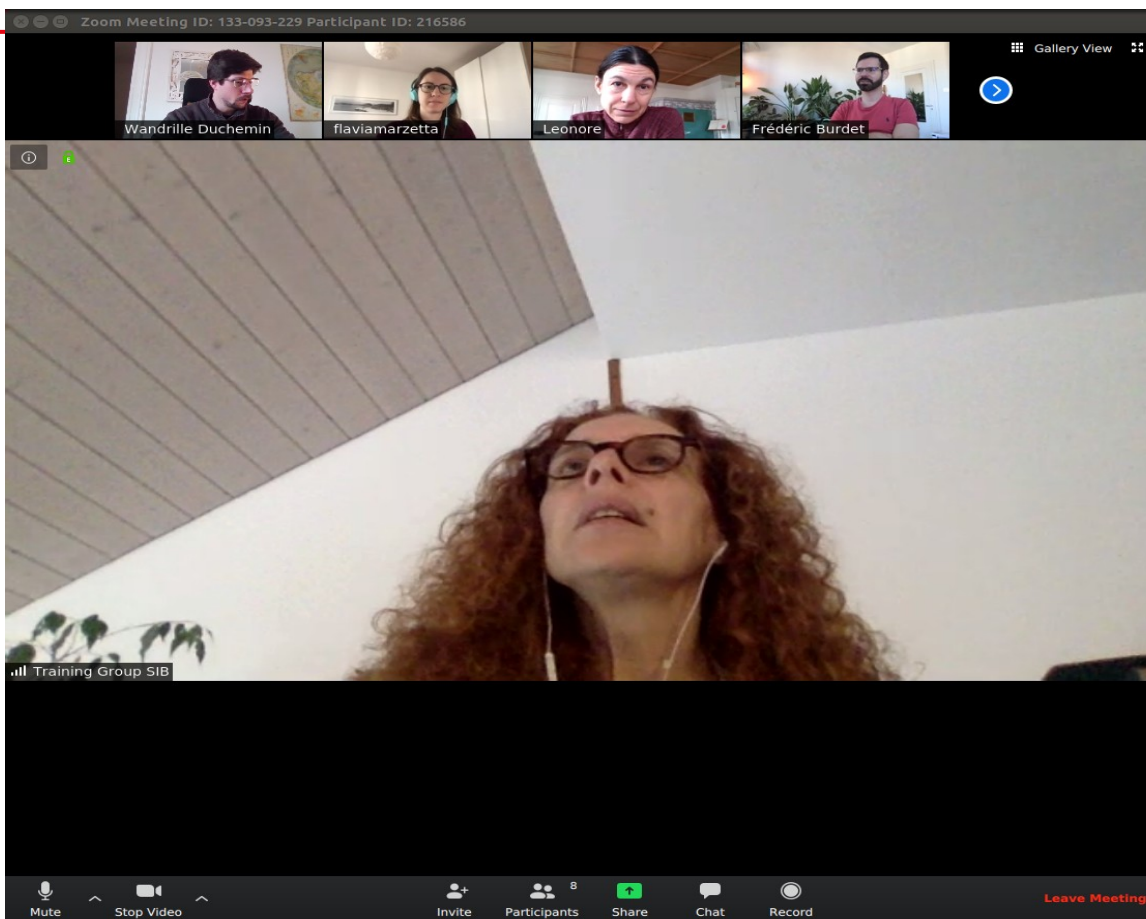
Course page : <https://edu.sib.swiss/course/view.php?id=455>

Student account : pls20

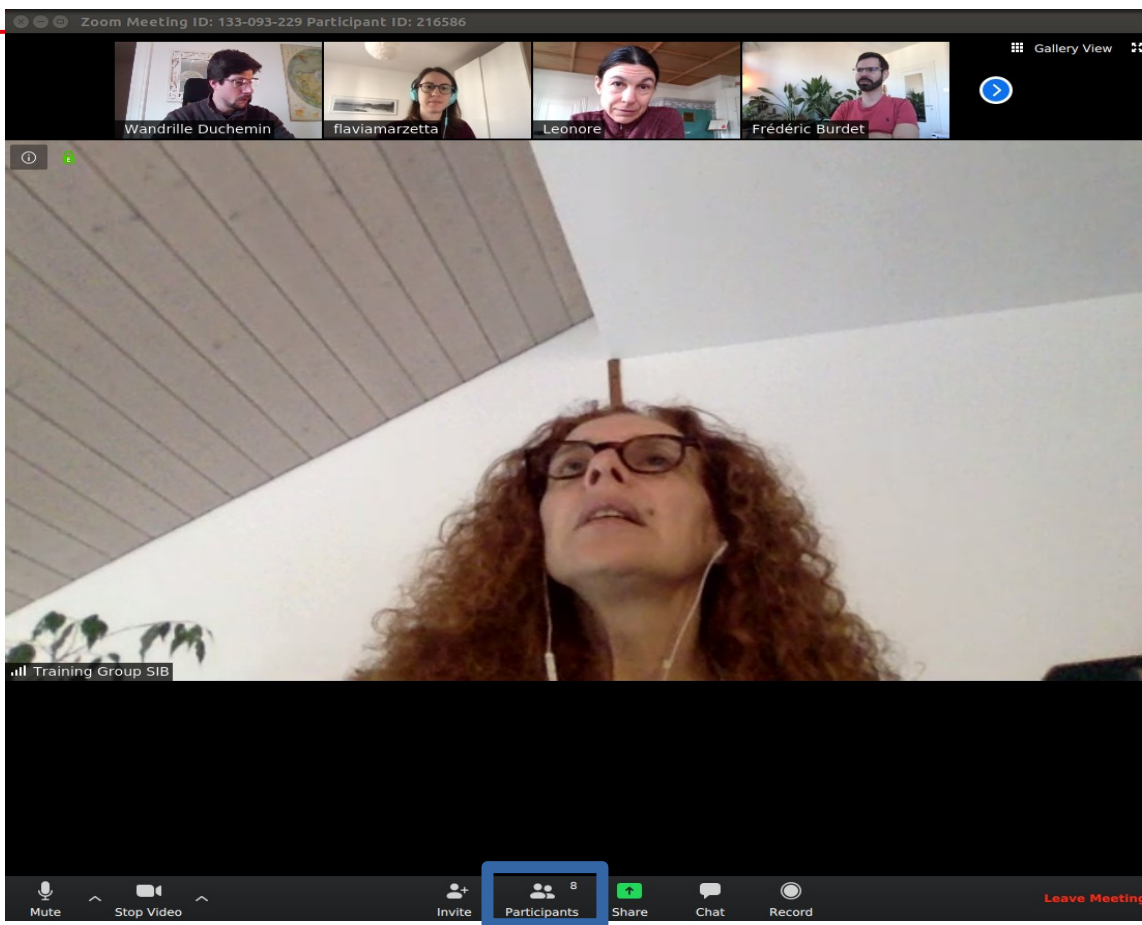
password: SIB-pls20

Etherpad : <https://public.etherpad-mozilla.org/p/python-ge2018>

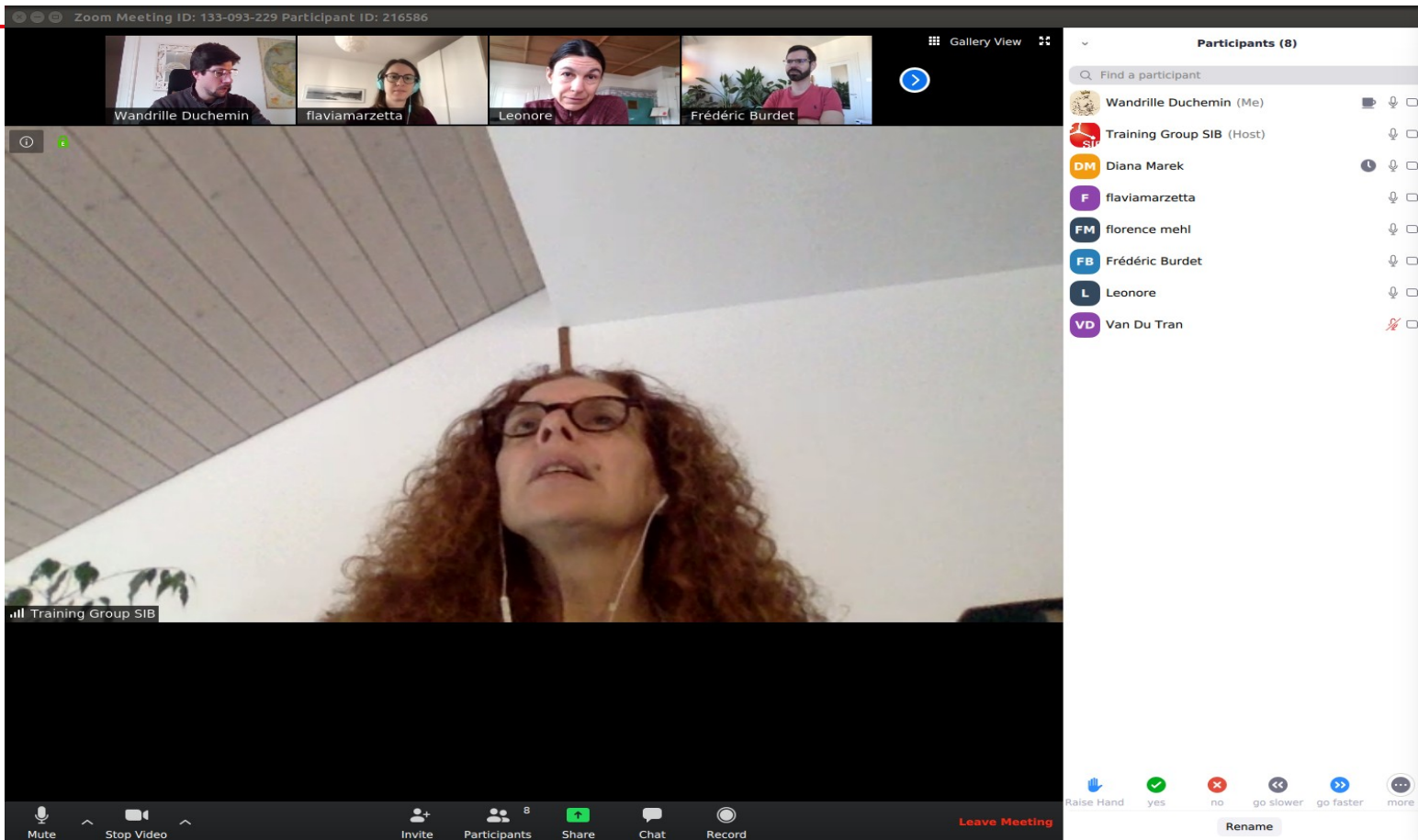
# Online meeting client



# Online meeting client

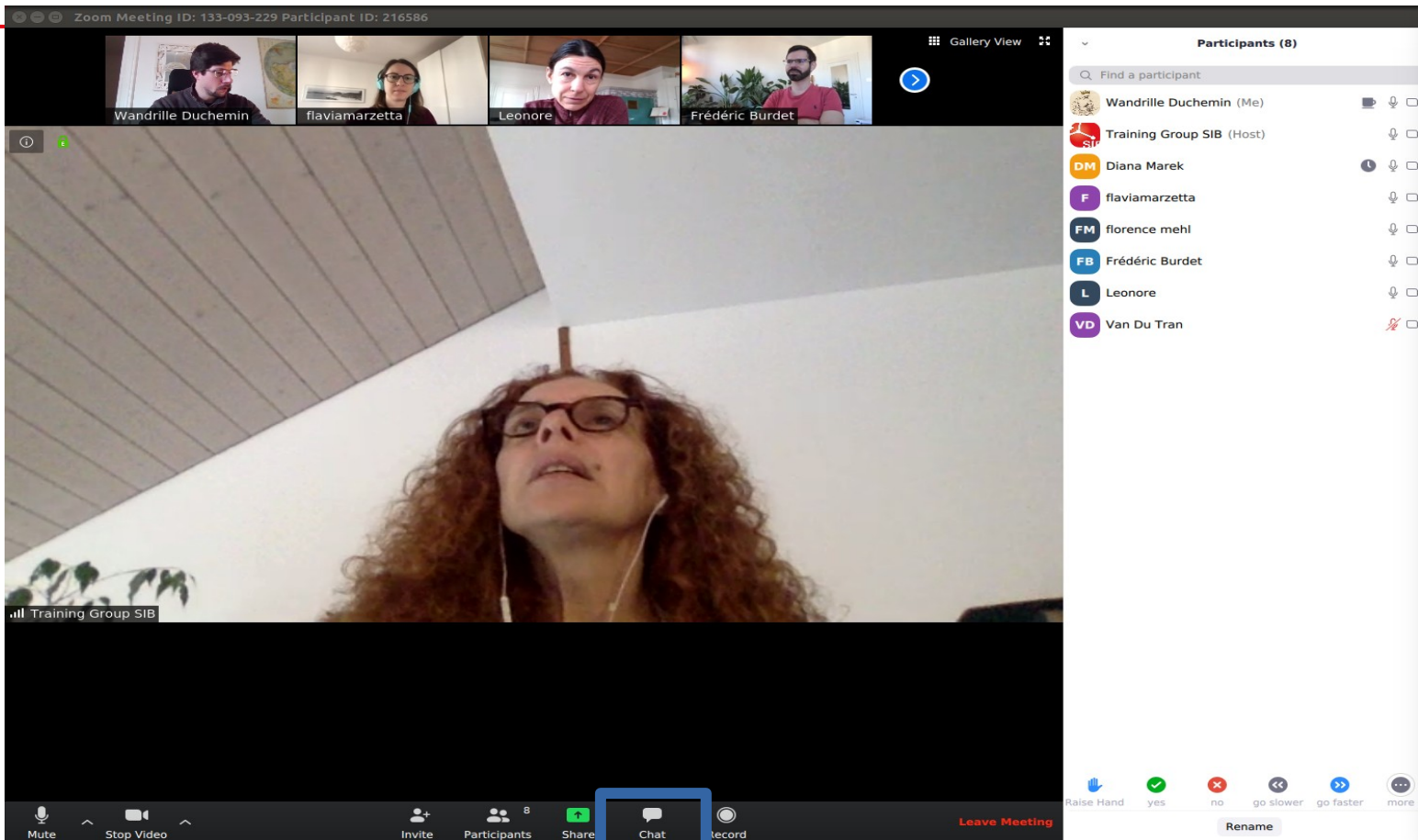


# Online meeting client

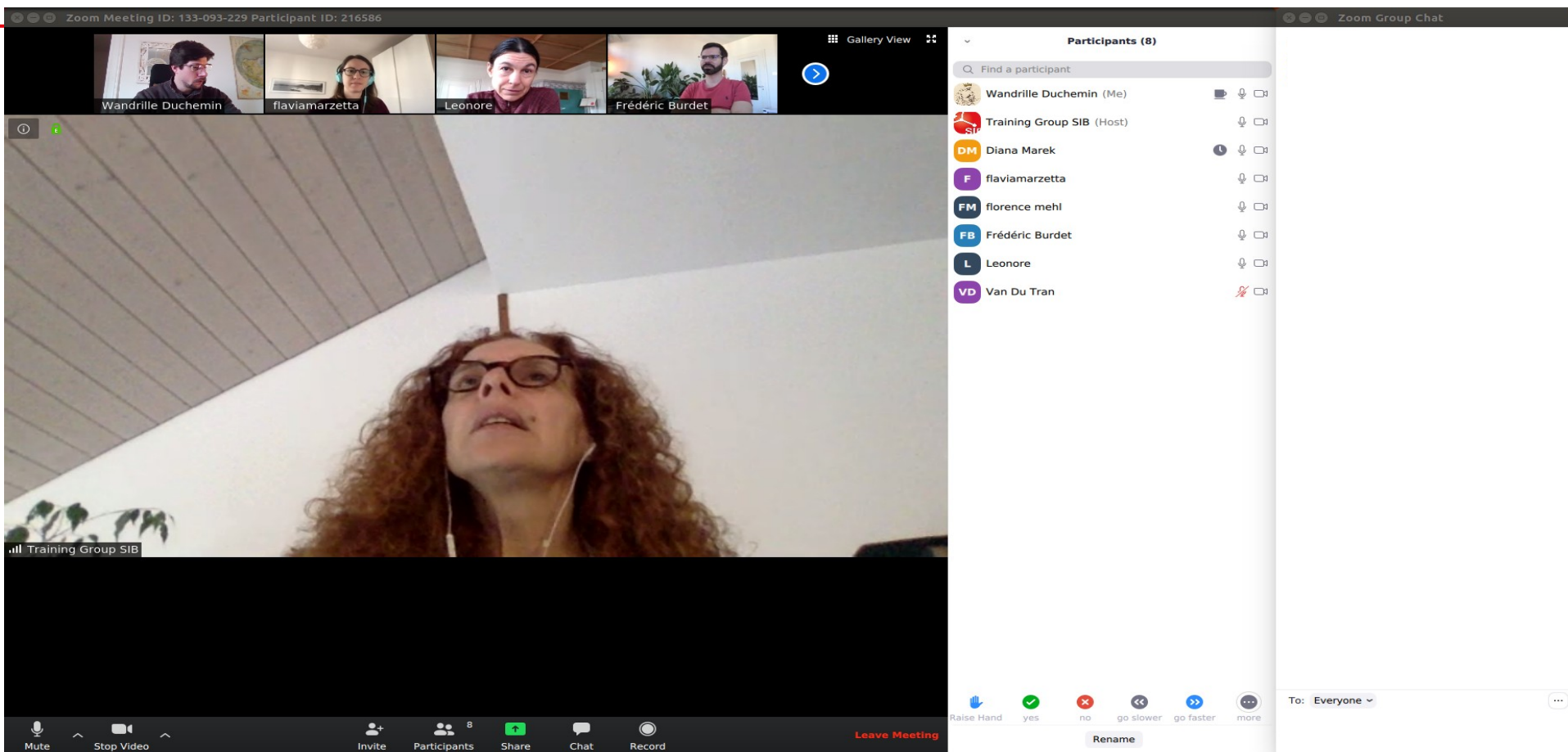




# Online meeting client



# Online meeting client



# Online meeting client

Zoom Meeting ID: 133-093-229 Participant ID: 216586

Gallery View

Participants (8)

Find a participant

- Wandrille Duchemin (Me)
- Training Group SIB (Host)
- DM Diana Marek
- F flaviamarzetta
- FM florence mehl
- FB Frédéric Burdet
- L Leonore
- VD Van Du Tran

Mute Stop Video Invite Participants Share Chat Record Leave Meeting

Raise Hand yes no go slower go faster more

To: Everyone



# During practicals - breakrooms

The screenshot displays a Zoom meeting interface. At the top, the meeting ID is 133-093-229 and the participant ID is 216586. The main video area shows a large view of a participant with curly hair and glasses, with a smaller gallery view of four other participants (Wandrille Duchemin, flaviamarzetta, Leonore, and Frédéric Burdet) at the top. The bottom toolbar includes buttons for Mute, Stop Video, Invite, Participants, Share, Chat, Record, and Leave Meeting. A blue box highlights the Mute and Stop Video buttons. On the right, the Participants list shows 8 participants, including Wandrille Duchemin (Me), Training Group SIB (Host), Diana Marek, flaviamarzetta, florence mehl, Frédéric Burdet, Leonore, and Van Du Tran. The Zoom Group Chat window is also visible on the right.

Zoom Meeting ID: 133-093-229 Participant ID: 216586

Gallery View

Participants (8)

Find a participant

- Wandrille Duchemin (Me)
- Training Group SIB (Host)
- DM Diana Marek
- F flaviamarzetta
- FM florence mehl
- FB Frédéric Burdet
- L Leonore
- VD Van Du Tran

Mute Stop Video Invite Participants Share Chat Record Leave Meeting

Zoom Group Chat

To: Everyone

# During practicals - breakrooms

The screenshot displays a Zoom meeting interface. At the top, the meeting ID is 133-093-229 and the participant ID is 216586. The main video area shows a large view of a participant with curly hair and glasses, looking upwards. Above this, a gallery view shows four smaller video feeds of other participants: Wandrille Duchemin, flaviamarzetta, Leonore, and Frédéric Burdet. A blue box highlights the 'Share' button in the bottom toolbar. To the right, the 'Participants (8)' list is visible, showing the names of all participants and their status (audio on/off, video on/off). The bottom toolbar includes buttons for Mute, Stop Video, Invite, Participants, Share, Chat, Record, and Leave Meeting. The 'Share' button is highlighted with a blue box.

Zoom Meeting ID: 133-093-229 Participant ID: 216586

Gallery View

Participants (8)

Find a participant

- Wandrille Duchemin (Me)
- Training Group SIB (Host)
- DM Diana Marek
- F flaviamarzetta
- FM florence mehl
- FB Frédéric Burdet
- L Leonore
- VD Van Du Tran

Share

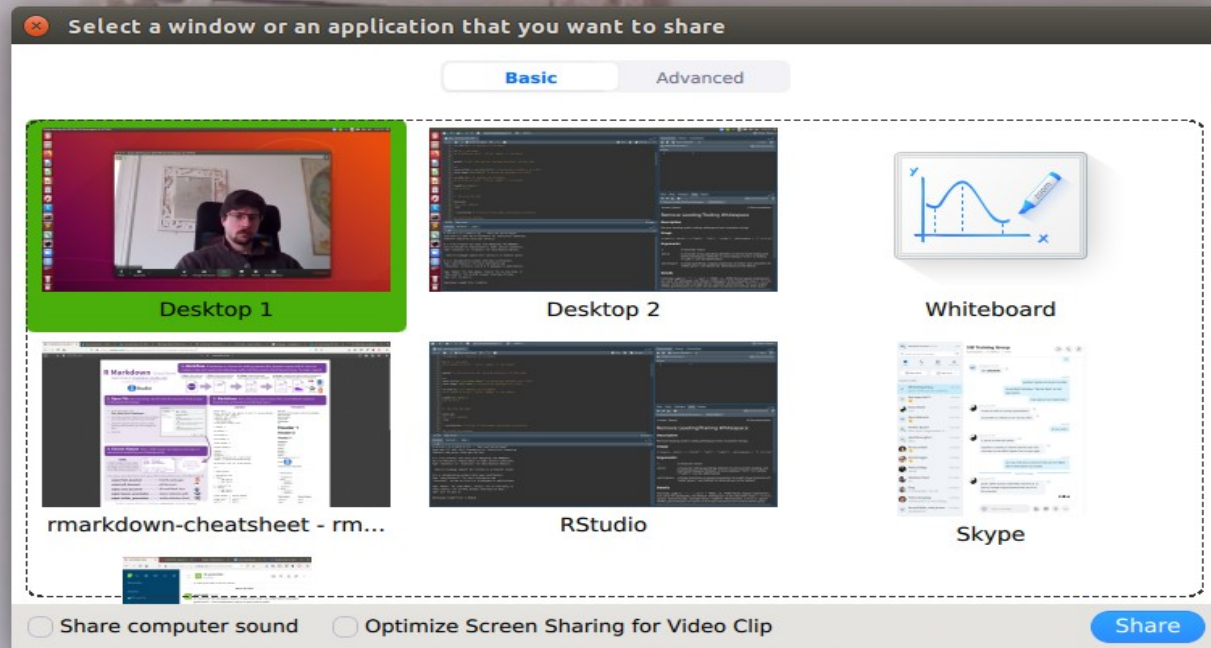
Leave Meeting

Zoom Group Chat

To: Everyone

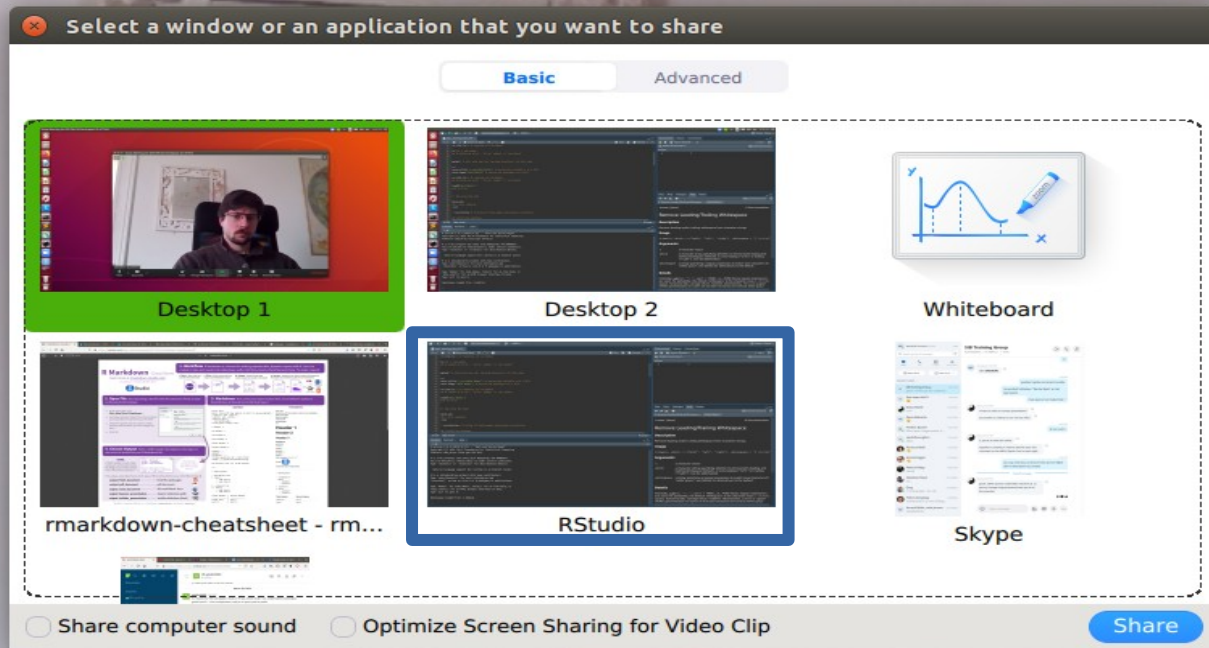
# During practicals - breakrooms

Zoom Meeting ID: 878-789-163 Participant ID: 477934



# During practicals - breakrooms

Zoom Meeting ID: 878-789-163 Participant ID: 477934





# During practicals - breakrooms

The screenshot displays the RStudio IDE interface during a remote session. The main editor window shows an R script titled 'day1\_morning\_intro.R.R\*'. The code includes comments and workspace management functions. A pink oval highlights a comment block, and a pink arrow labeled 'Training G' points to it. The console window at the bottom shows the R version (3.6.2) and system information. The right sidebar shows the 'Global Environment' and 'R Documentation' for the 'trimws' function.

```
1 #' ---
2 #' title: "first steps with R - April 2020 - day1 morning"
3 #' author: "Wandrille Duchemin"
4 #' date: "07/02/2020"
5 #' ---
6
7 # code to go with the first day presentation of the "first steps with R" course
8 # subject : introduction to R
9
10
11 #' ## workspace management
12
13 a=3 #declaring a variable named a
14 ls()# lists the created variables
15
16 rm(list=ls()) # removing all variables
17
18 ls()# -> now empty
19 #a # yields an error : "Error: object 'a' not found"
20
21
22 getwd() # this give you the "working directory" of this code
23
24 a=3
25 save(a,file="a_variable.RData") # saving the variable a to a file
26 save.image("day1.RData") # saving the workspace to a file
27
28 rm(list=ls()) # removing all variables
29 (No functions defined) Error: object 'a' not found
30
31 (Top Level) :
```

Environment is empty

R Documentation

## Remove Leading/Trailing Whitespace

### Description

Remove leading and/or trailing whitespace from character strings.

### Usage

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \\t\\r\\n"])
```

### Arguments

x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

### Details

Internally, `sub(re, "", x, perl = TRUE)`, i.e., PCRE library regular expressions are used. For portability, the default 'whitespace' is the character class `[ \\t\\r\\n]` (space, horizontal tab, carriage return, newline). Alternatively, `[\\h\\v]` is a good



# During practicals - breakrooms

The screenshot displays the RStudio IDE interface during a practical session. The main editor window shows R code for workspace management. A pink oval highlights a comment block, with a pink arrow labeled "Training G" pointing to it. The top toolbar has "Annotate" and "Remote Control" buttons highlighted with red boxes. The bottom console shows R version 3.6.2 and system information. The right sidebar shows the "Global Environment" and "R Documentation" for "trimws".

```
1 #' ---
2 #' title: "first steps with R - April 2020 - day1 morning"
3 #' author: "Wandrille Duchemin"
4 #' date: "07/02/2020"
5 #' ---
6
7 # code to go with the first day presentation of the "first steps with R" course
8 # subject : introduction to R
9
10
11 #' ## workspace management
12
13 a=3 #declaring a variable named a
14 ls()# lists the created variables
15
16 rm(list=ls()) # removing all variables
17
18 ls()# -> now empty
19 #a # yields an error : "Error: object 'a' not found"
20
21
22 getwd() # this give you the "working directory" of this code
23
24 a=3
25 save(a,file="a_variable.RData") # saving the variable a to a file
26 save.image("day1.RData") # saving the workspace to a file
27
28 rm(list=ls()) # removing all variables
29 (No functions defined)
30
31 (Top Level) :
```

Environment is empty

R Documentation

## Remove Leading/Trailing Whitespace

### Description

Remove leading and/or trailing whitespace from character strings.

### Usage

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \\t\\r\\n"])
```

### Arguments

x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

### Details

Internally, `sub(re, "", *, perl = TRUE)`, i.e., PCRE library regular expressions are used. For portability, the default 'whitespace' is the character class `[ \\t\\r\\n]` (space, horizontal tab, carriage return, newline). Alternatively, `[\\h\\v]` is a good

# During practicals - breakrooms

The screenshot displays the RStudio interface during a live session. The main editor window shows R code for workspace management. A pink oval highlights a comment block, with a pink arrow labeled "Training G" pointing to it. The console window shows R version information and help text. The right sidebar shows the "Global Environment" and "R Documentation" for the "trimws" function.

```
1 #' ---
2 #' title: "first steps with R - April 2020 - day1 morning"
3 #' author: "Wandrille Duchemin"
4 #' date: "07/02/2020"
5 #' ---
6
7 # code to go with the first day presentation of the "first steps with R" course
8 # subject : introduction to R
9
10
11 #' ## workspace management
12
13 a=3 #declaring a variable named a
14 ls()# lists the created variables
15
16 rm(list=ls()) # removing all variables
17
18 ls()# -> now empty
19 #a # yields an error : "Error: object 'a' not found"
20
21
22 getwd() # this give you the "working directory" of this code
23
24 a=3
25 save(a,file="a_variable.RData") # saving the variable a to a file
26 save.image("day1.RData") # saving the workspace to a file
27
28 rm(list=ls()) # removing all variables
29 (No functions defined) Error: object 'a' not found
30
31 (Top Level) :
```

Environment is empty

R Documentation

## Remove Leading/Trailing Whitespace

### Description

Remove leading and/or trailing whitespace from character strings.

### Usage

```
trimws(x, which = c("both", "left", "right"), whitespace = "[ \\t\\r\\n"])
```

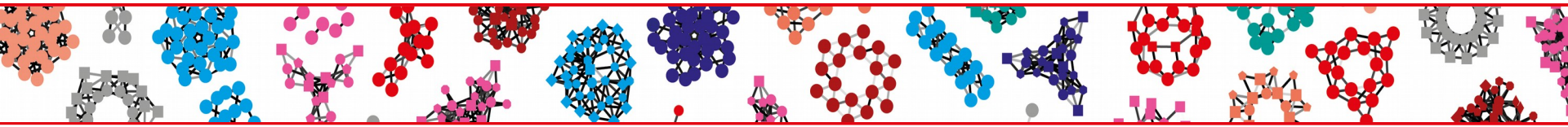
### Arguments

x	a character vector
which	a character string specifying whether to remove both leading and trailing whitespace (default), or only leading ("left") or trailing ("right"). Can be abbreviated.
whitespace	a string specifying a regular expression to match (one character of) "white space", see Details for alternatives to the default.

### Details

Internally, `sub(re, "", x, perl = TRUE)`, i.e., PCRE library regular expressions are used. For portability, the default 'whitespace' is the character class `[ \\t\\r\\n]` (space, horizontal tab, carriage return, newline). Alternatively, `[\\h\\v]` is a good

# Overview



01

• What is python ? Why use it ?

02

• Computer resources and how (not) to use them

03

• FAIR practices in coding

04

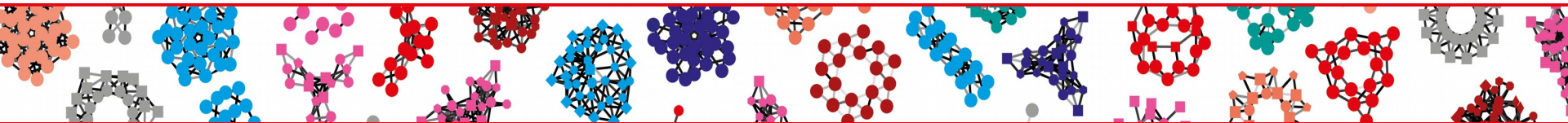
• Practice, practice and more practice

# 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 ?

# Overview



01

• What is python ? Why use it ?

02

• Computer resources 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

# What is Python ?

---

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

# 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

- + focus on the application itself
- - possibly counter-intuitive behaviours

■ **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

---

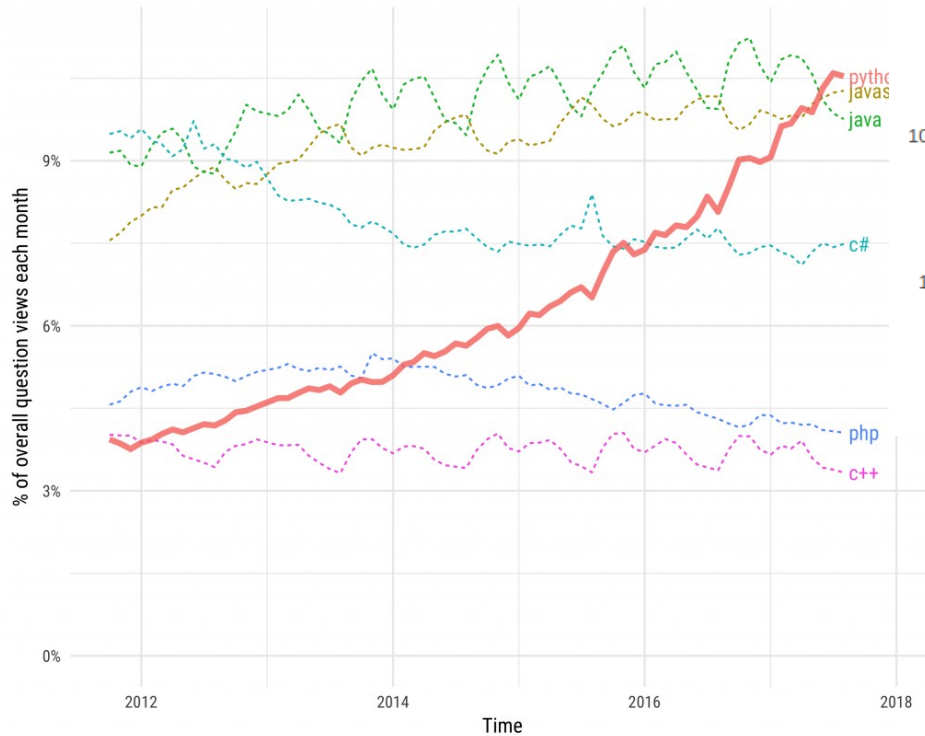


- **1990** : Python begins
- **2000** : Python 2.0
- **2008** : Python 3.0
- **2015 2020** : end of Python 2.7 support
- **Current version** : 3.8 (as of february 2020)

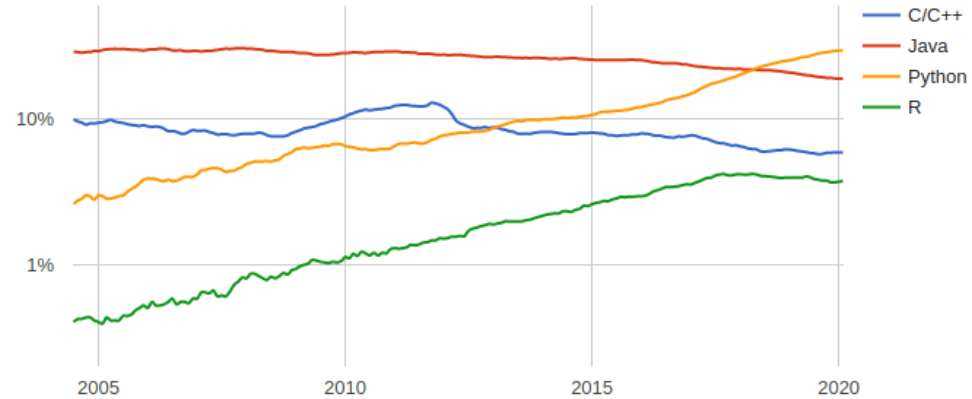
# Python - is it used ?

## Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



## PYPL Popularity of Programming Language



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

Source: <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>



# Python – sum up

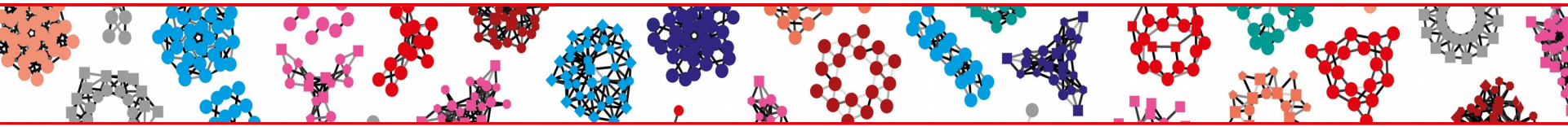
---

“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 resources and how (not) to use them**

03

• FAIR practices in coding

04

• Practice, practice and more practice

# A computer's ressources

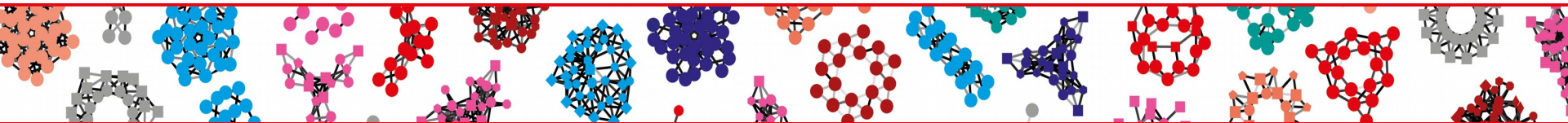
---

- **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
- **Hard disk** : 'long term memory' of the computer. Slow
  - HDD : older, 'cheap', slow
  - SSD : newer, 'expensive', faster
- ... external ? file/computation accessed through the network ?

# A computer's ressources

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 $\mu$ s	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

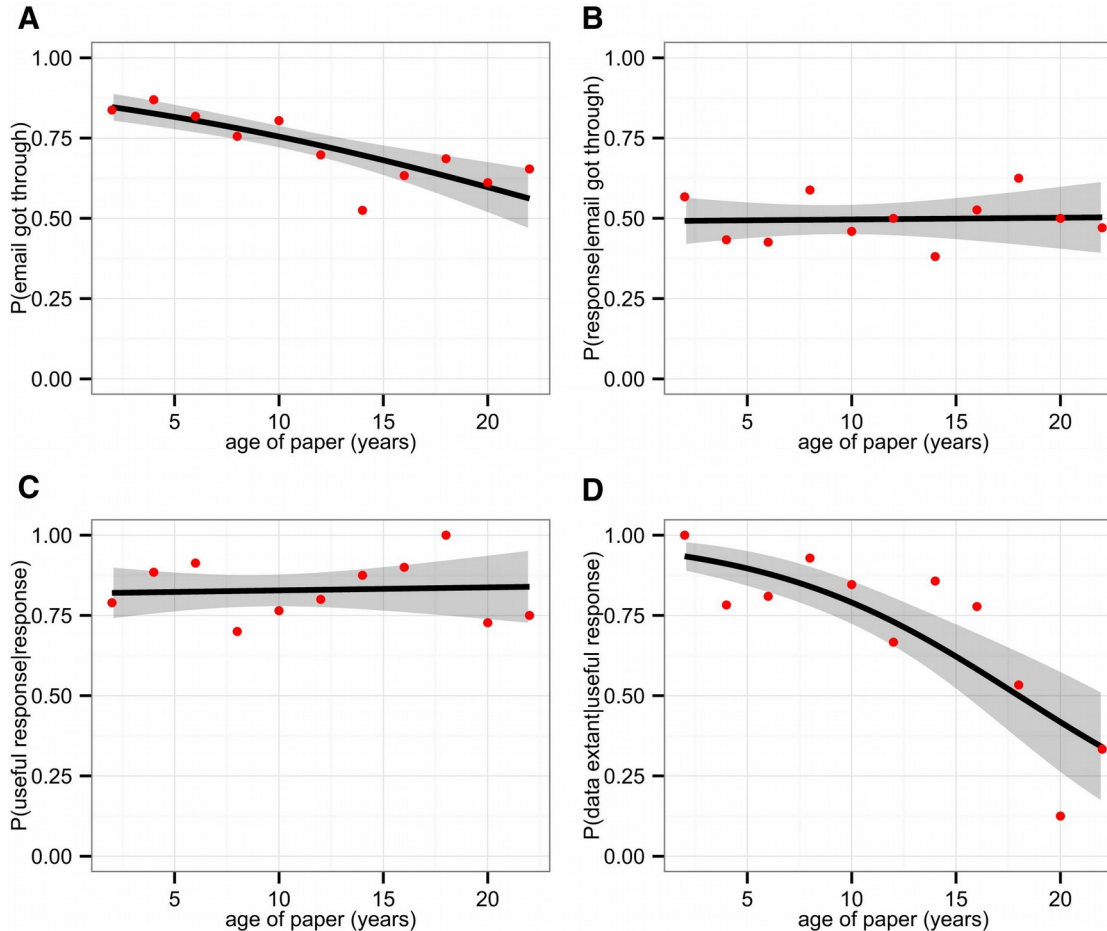
• **FAIR practices in coding**

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...

# The reproducibility crisis

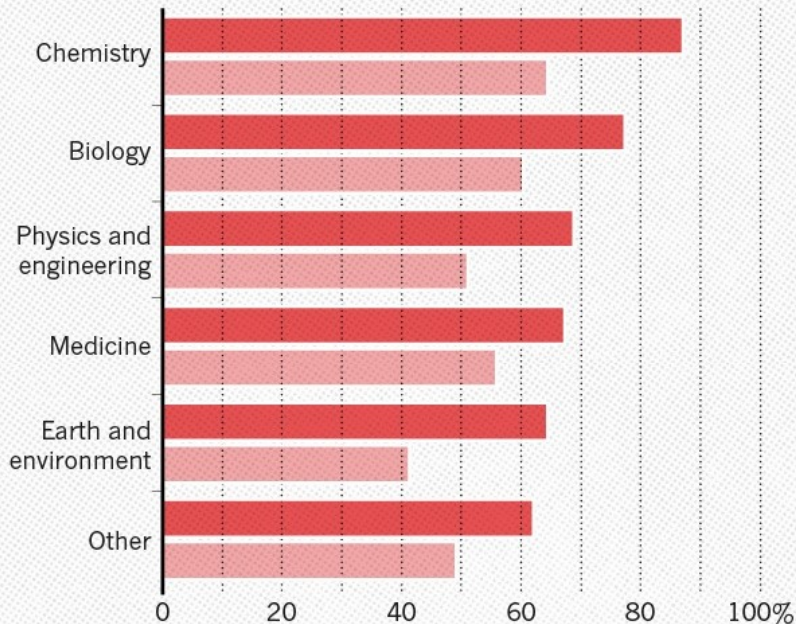
'1,500 scientists lift the lid on reproducibility'

M. Baker 2016, nature news feature

## HAVE YOU FAILED TO REPRODUCE AN EXPERIMENT?

Most scientists have experienced failure to reproduce results.

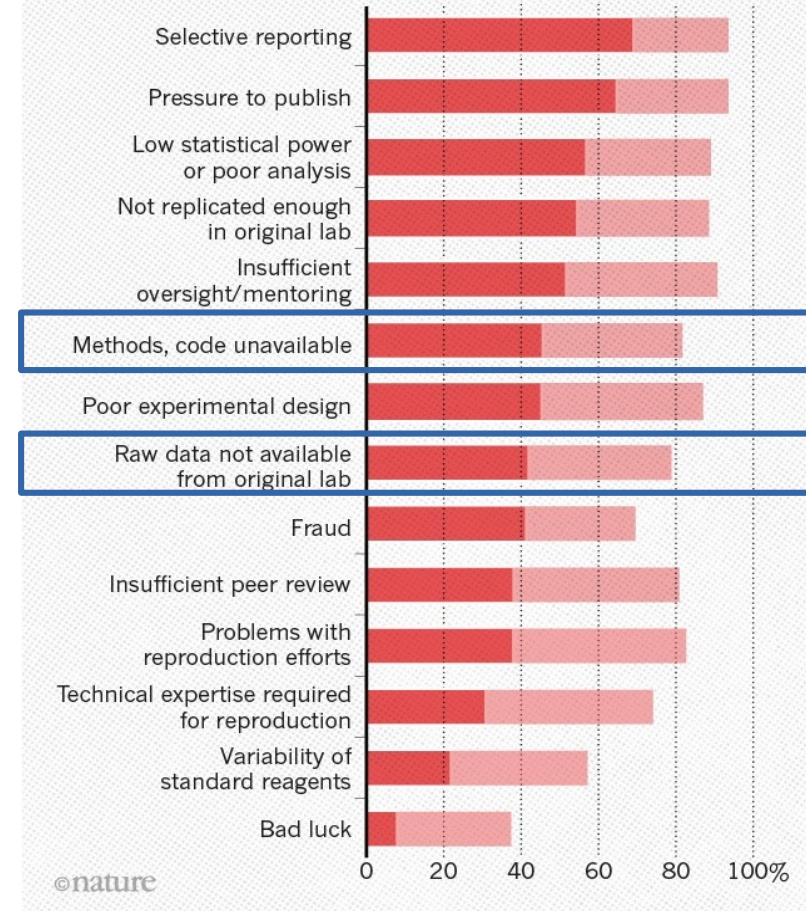
● Someone else's ● My own



## WHAT FACTORS CONTRIBUTE TO IRREPRODUCIBLE RESEARCH?

Many top-rated factors relate to intense competition and time pressure.

● Always/often contribute ● Sometimes contribute



# The **FAIR** Guiding Principles

---



**Findable** : Metadata and data should be easy to find for both humans and computers



**Accessible** : The exact conditions under which the data is accessible should be provided in such a way that humans and machines can understand them



**Interoperable** : the (meta)data should be based on standardized vocabularies, ontologies, thesauri etc. so that it integrates with existing applications or workflows



**Reusable** : Metadata and data should be well described so that they can be replicated and/or combined in different research settings

# FAIR applied to code

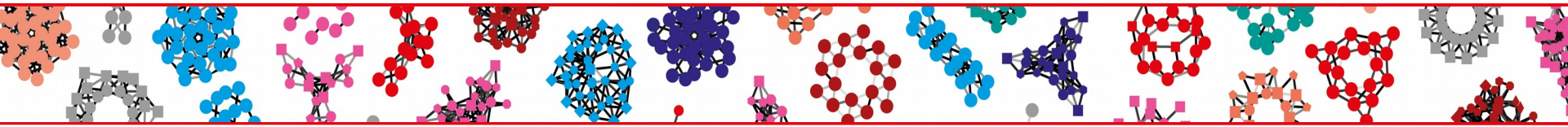
---

- ❑ 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 ??

For this :

- ❑ Comment everything
- ❑ Use explicit names when naming things
- ❑ Jupyter-lab can help you
- ❑ Also, you can look at : [Ten quick tips for getting the most scientific value out of numerical data](#)

# 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 ?

---

- 3 main modes of interaction :
- Interactive console
- Python code file (.py)
- Jupyter notebook



**Live demo time**

The following slides are here for posterity



# 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 **testing** things out
- ❑ But, you keep **no trace** of your workflow/environment

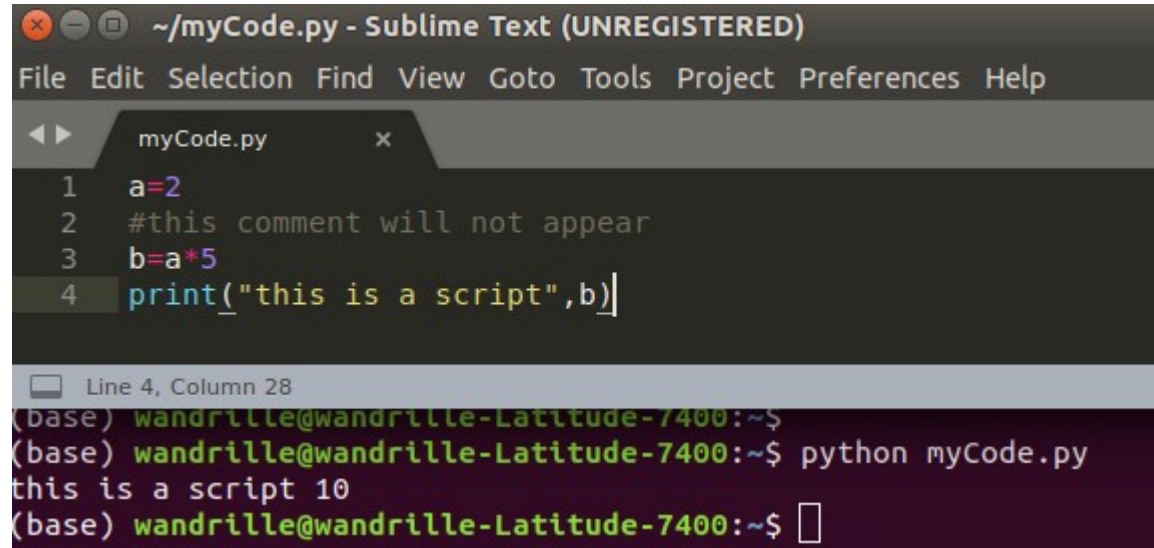


Only use it to test little bits of code



# Python – writing a code in a .py file

- ❑ Write a script, 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)



The screenshot shows a Sublime Text editor window titled `~/myCode.py - Sublime Text (UNREGISTERED)`. The editor displays a Python script in `myCode.py` with the following code:

```
1 a=2
2 #this comment will not appear
3 b=a*5
4 print("this is a script",b)
```

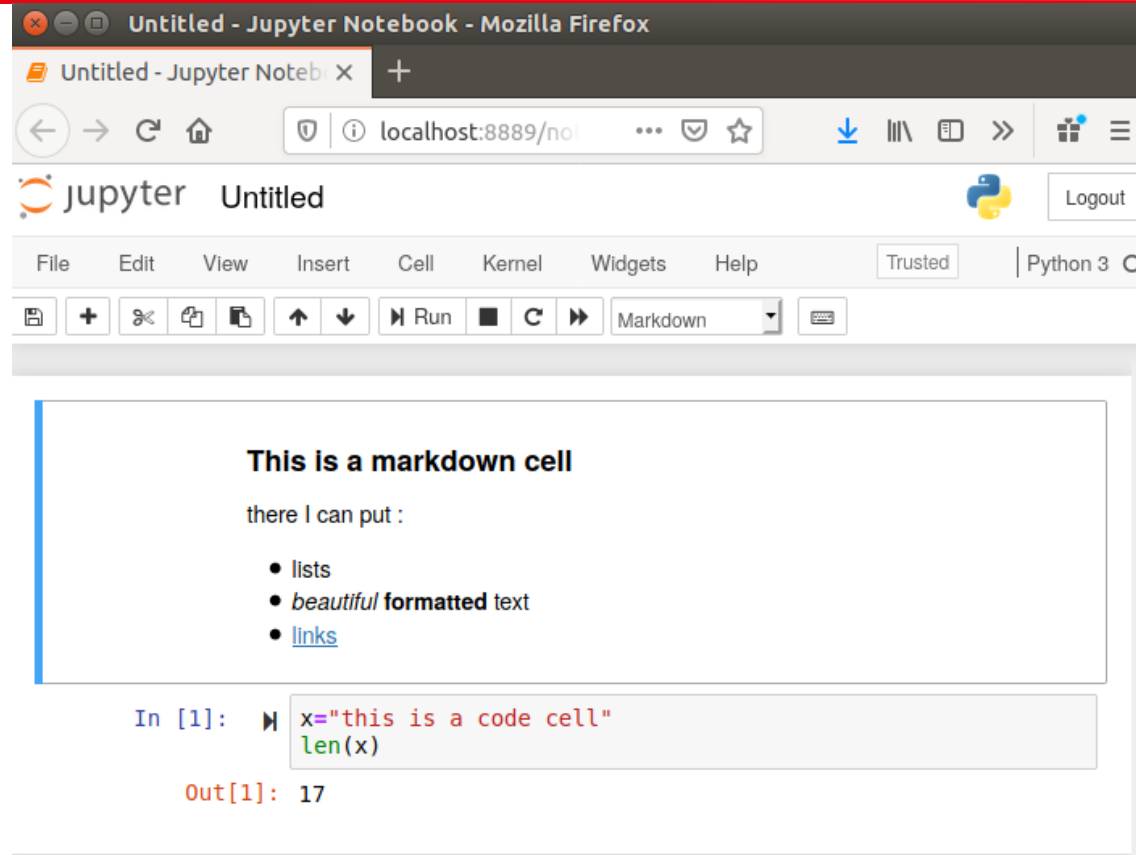
Below the editor, a terminal window shows the execution of the script:


```
(base) wandrille@wandrille-Latitude-7400:~$
(base) wandrille@wandrille-Latitude-7400:~$ python myCode.py
this is a script 10
(base) wandrille@wandrille-Latitude-7400:~$
```

 General purpose coding, “operational script”

# Python – jupyter

- Browser based interface
- Interlace markdown and python 'cells'
- Execute code cell by cell
- Commentary, code, and results together in the same file
- Visually pleasant and fairly ergonomic



 Ideal “analysis scripts” : helps reproducibility of results