Introduction

Scientific Programming in Python

About the course

Times:

- One **lecture** every *Monday 14:00 - 16:00*
- One **tutorial session** every *Wednesday 14:00 - 16:00* (*assisted working*)
- We may add a practice session if time and you demand it
- No mandatory attendance
- Lecture will hopefully be filmed

About the course

Grading & Credits

- Weekly homework, done individually
- Homework corrected automatically, and you get the tests, too!
- 11 homework-sheets plus bonus exercises once in a while
- Pass **9** (normal + bonus) to get the Schein
- 4-ECTS Schein for the “Profilbildender Wahlbereich”
- *You come here because you want to learn*
- **Schein will not be graded and doesn’t count for any module!**

About us

Rüdiger

- 4th semester Master, 10th overall
- 3 years of Python experience
- rbusche@uos.de

Chris

- 3rd Master, 10th coxi-semester overall
- ~3 years Python
- cstenkamp@uos.de

- If you have any questions or find errors on the slides, don't hesitate to write us an email or in the forum!
- If you have suggestions for content, please also write!
- If you encounter errors in the homework, please do so via a github-issue! The repository for homework is https://github.com/scientificprogrammingUOS

Why scientific programming?

Science

- Build and organize knowledge
- Test explanation about our world
- Systematically
- Objectively
- Transparently
- Reproducibly

Otherwise it’s not science.

Cf. PyData Ann Arbor: Katy Huff | Doing Our Best: Practices in Open, Reproducible, Scientific Computing

Programming helps us

- Build and organize knowledge → by building databases of scientific results
- Test explanation about our world → by automating experiments
- Systematically → by reducing the risk of human error
- Objectively → computers are not subject to human biases
- Transparently → by using open source tools and sharing our analyses
- Reproducibly → by codifying our analyses we make them repeatable

You need to write **clean code**!

Cf. [PyData Ann Arbor: Katy Huff | Doing Our Best: Practices in Open, Reproducible, Scientific Computing]

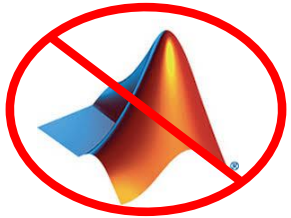
Why Python?

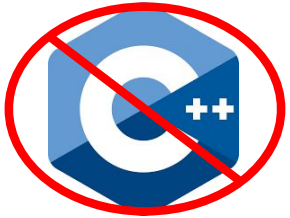
Python

- Create by Guido van Rossum in the 90s
- Now open source project developed by the Python Software foundation
- High-level language (no hardware-knowledge necessary)
- Interpreted and dynamically typed language
- Consistent and minimal syntax
- → easy to learn and write
- Great ecosystem and great community!

Python is better

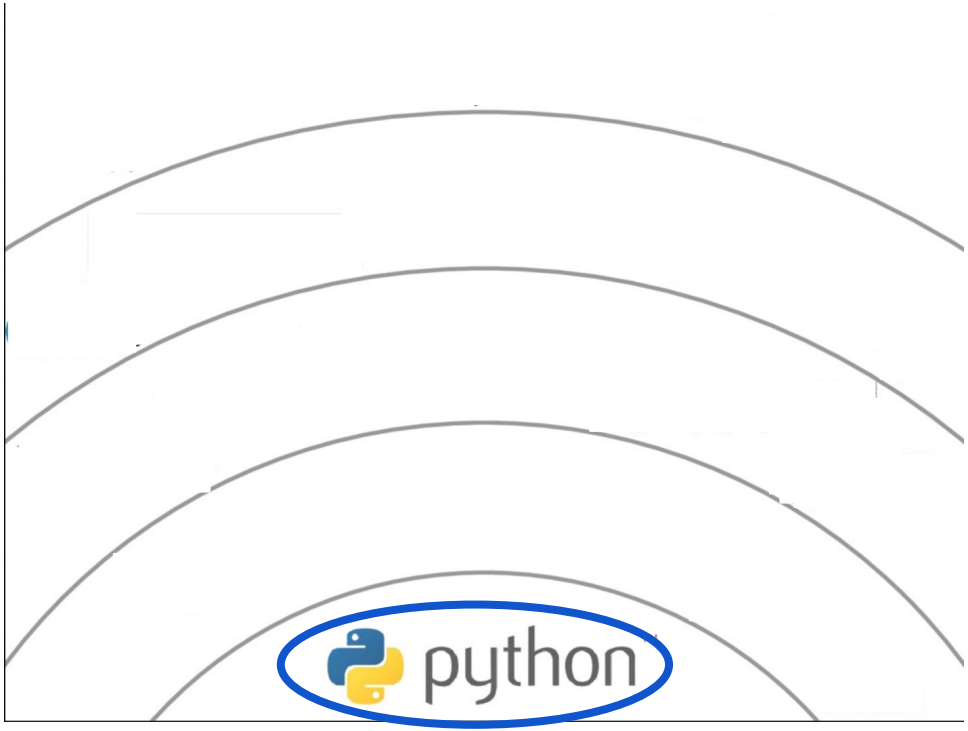
- Better than R → As a general purpose programming language you can do anything with Python not just statistics
- Better than Matlab → As a free and open source project you can save money and actually share your results
- Better than C++ (at least for science) → With a great ecosystem and a great community you can get stuff done, instead of trying to figure out documentation yourself
- Better than Java → Get more done with less code and without overly complex object orientation.





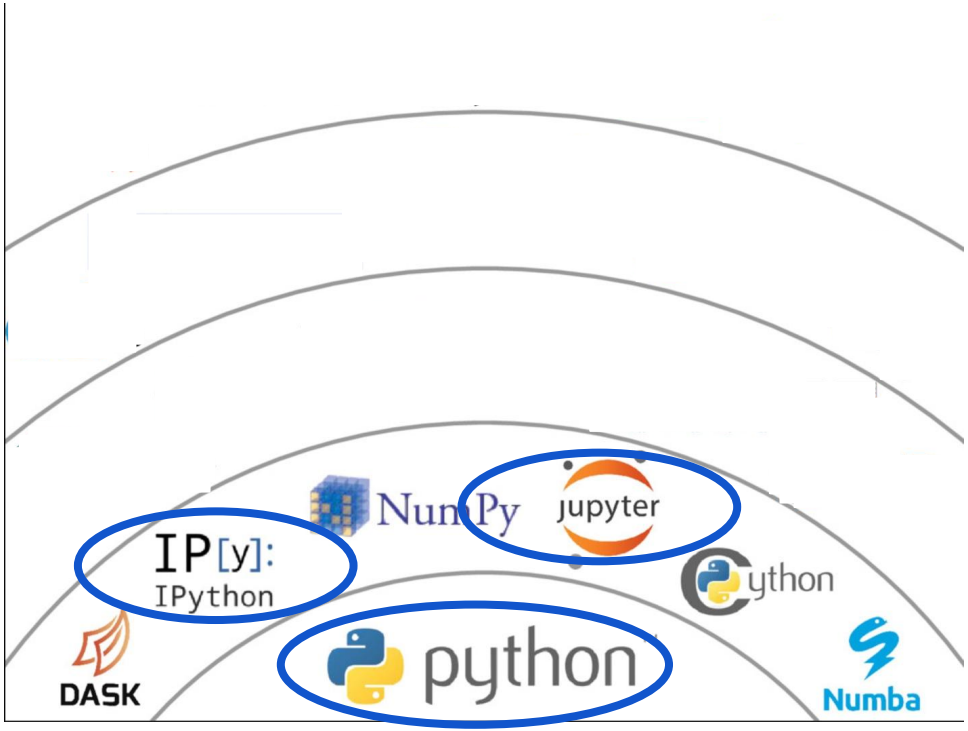
https://pixabay.com/de/schweizer-taschenmesser-taschenmesser-154314/

Our path through scientific python



https://www.datacamp.com/community/blog/python-scientific-computing-case

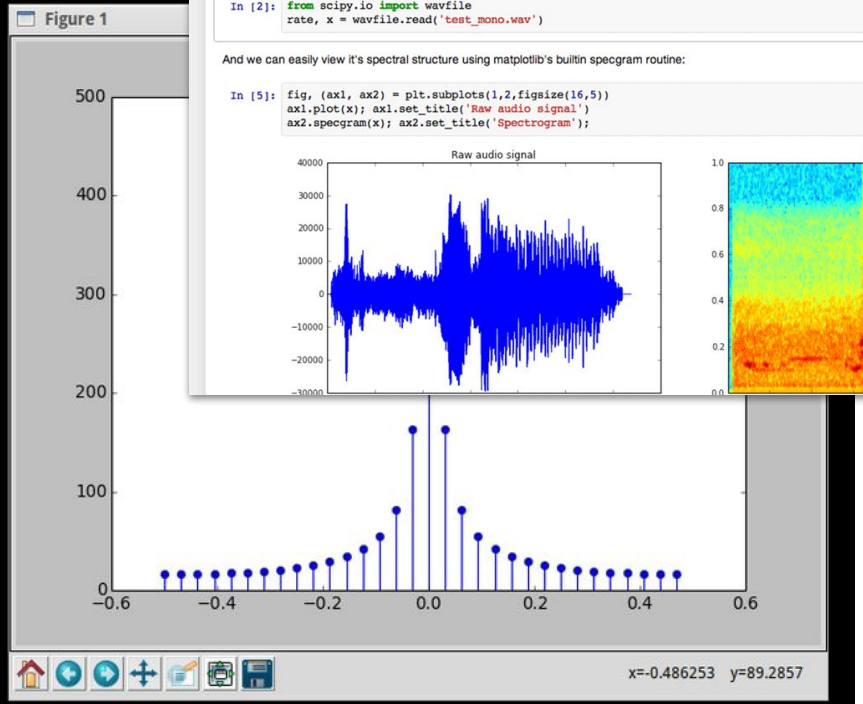
Our path through scientific python

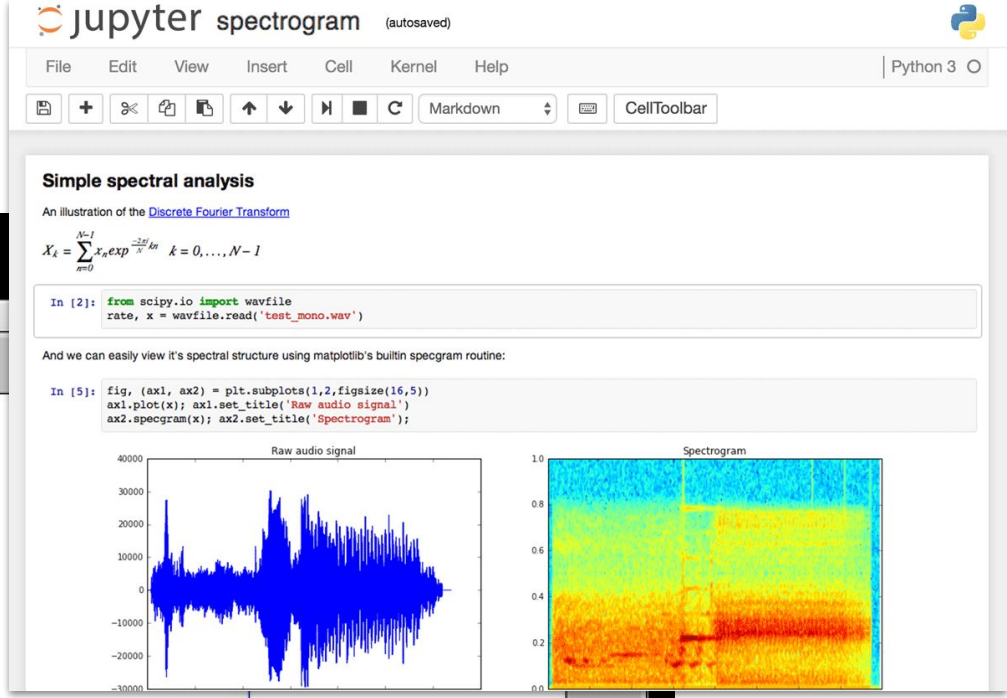


https://www.datacamp.com/community/blog/python-scientific-computing-case

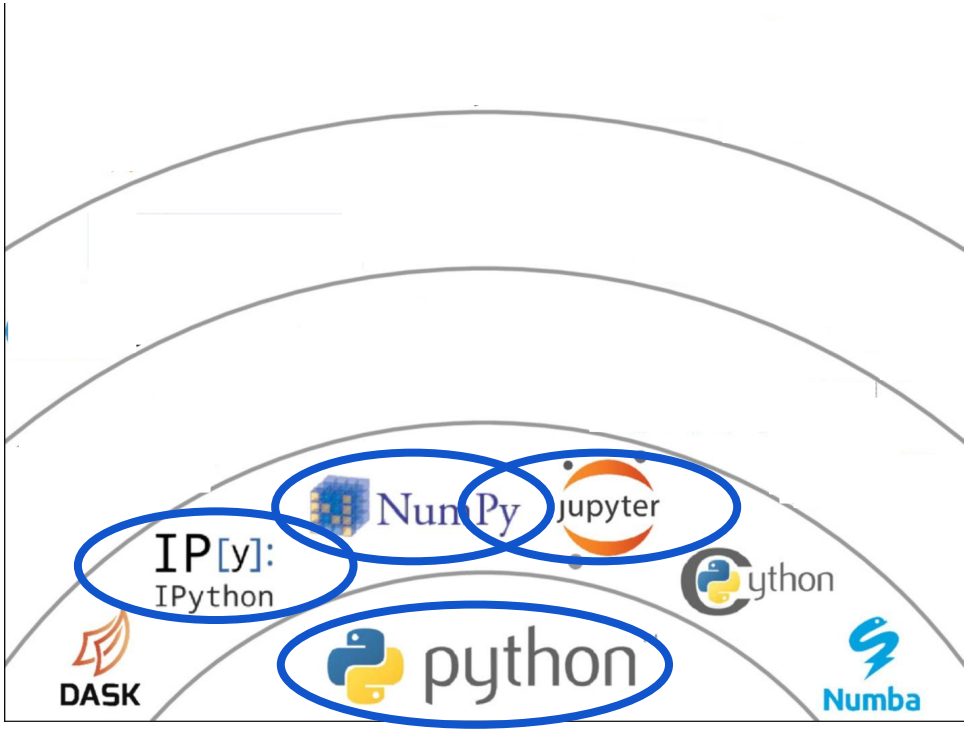
Python, Jupyter, IPython

```
Python 3.2.3 (default, Sep 25 2013, 18:25:56)
Type "copyright", "credits" or "license" for more information.

IPython 1.1.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.
Using matplotlib backend: TkAgg

In [1]: from numpy.fft import *

In [2]: a = arange(32)

In [3]: A = fft(a)

In [4]: f = fftfreq(32)

In [5]: stem(f,abs(A))
Out[5]: <Container object of 3 artists>

In [6]:
```





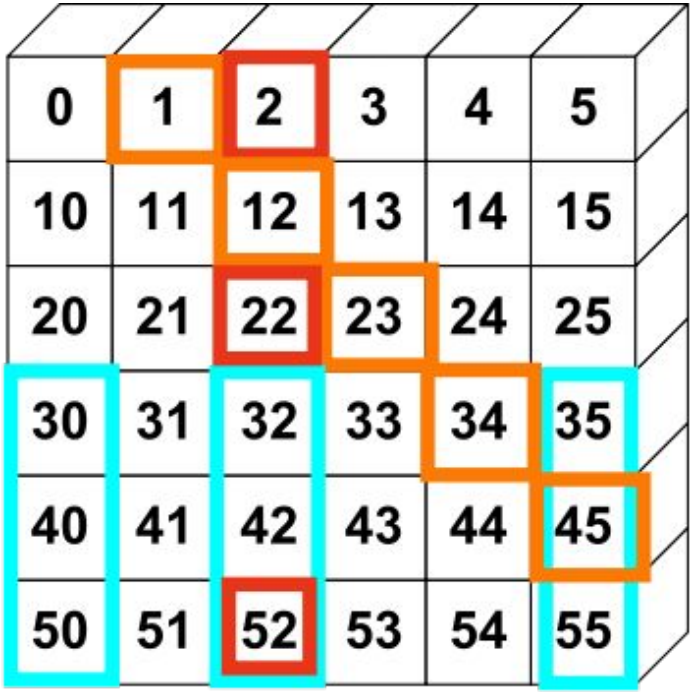
Our path through scientific python



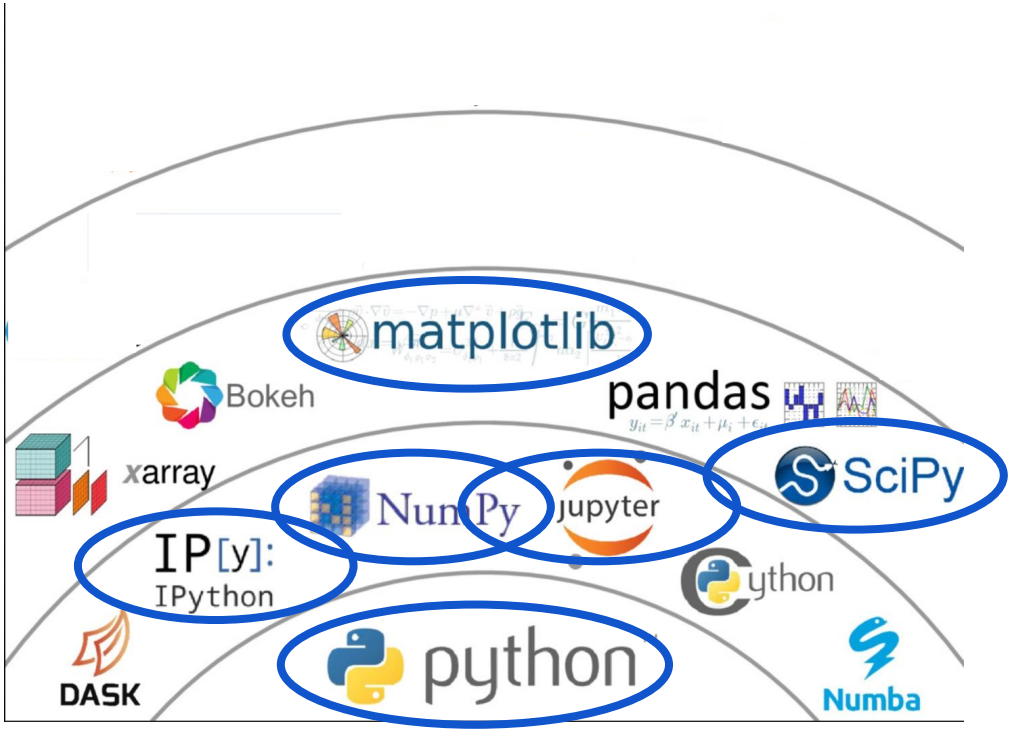
https://www.datacamp.com/community/blog/python-scientific-computing-case

NumPy

```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
       [40, 42, 45]])
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```

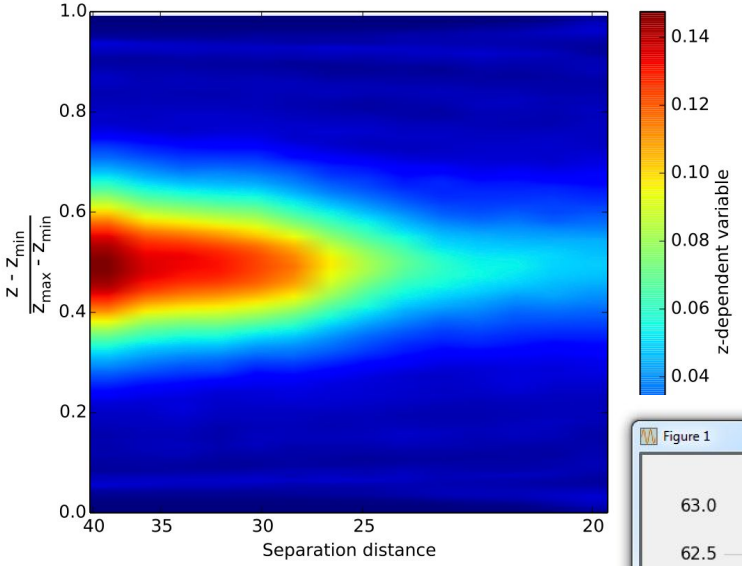


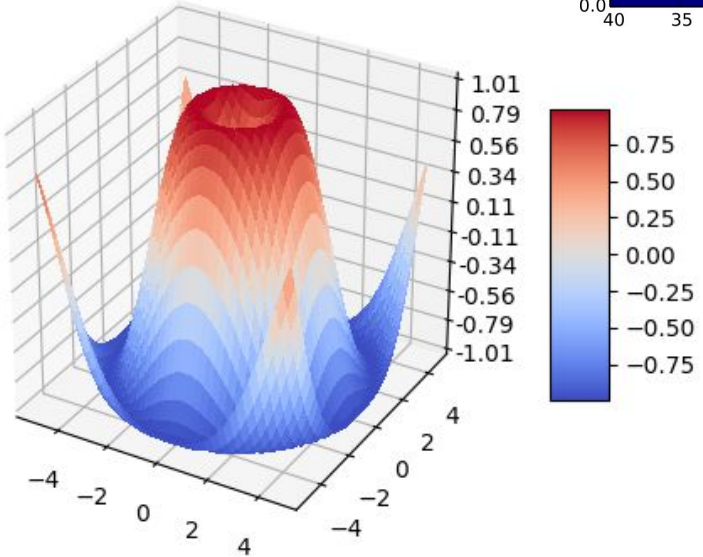
Our path through scientific python



https://www.datacamp.com/community/blog/python-scientific-computing-case

Matplotlib


1.0
0.8
0.6
0.4
0.2
0.0
$\frac{z - z_{min}}{z_{max} - z_{min}}$
40
35
30
25
20
Separation distance
0.14
0.12
0.10
0.08
0.06
0.04
z-dependent variable


1.01
0.79
0.56
0.34
0.11
-0.11
-0.34
-0.56
-0.79
-1.01
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−4
−2
0
2
4

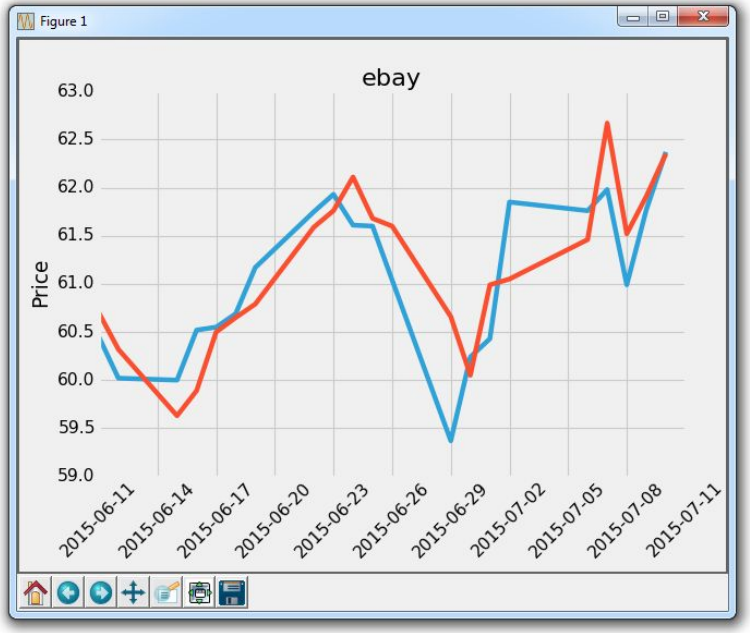
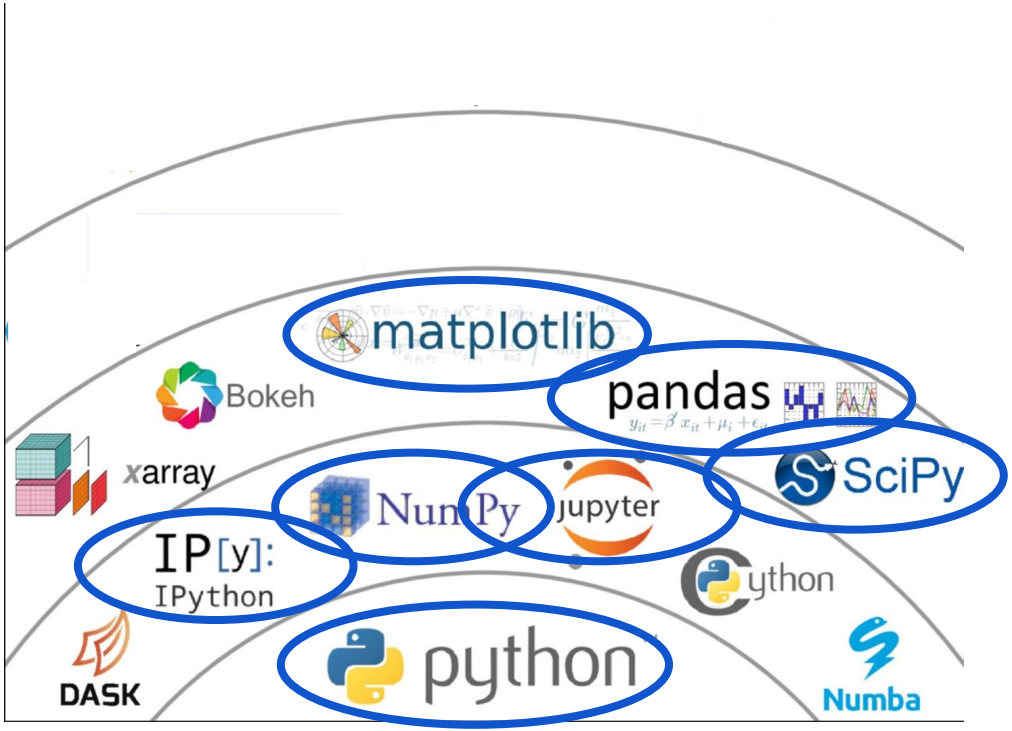


Figure 1
ebay
Price
63.0
62.5
62.0
61.5
61.0
60.5
60.0
59.5
59.0
2015-06-11
2015-06-14
2015-06-17
2015-06-20
2015-06-23
2015-06-26
2015-06-29
2015-07-02
2015-07-05
2015-07-08
2015-07-11

Our path through scientific python



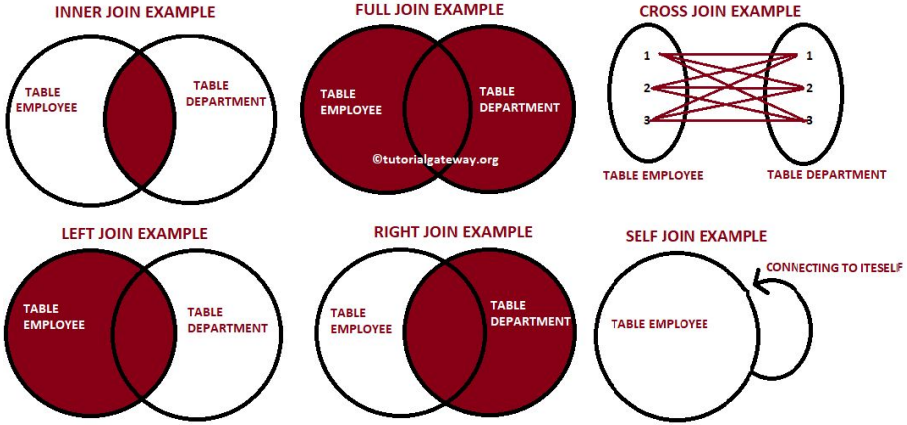
https://www.datacamp.com/community/blog/python-scientific-computing-case

Pandas

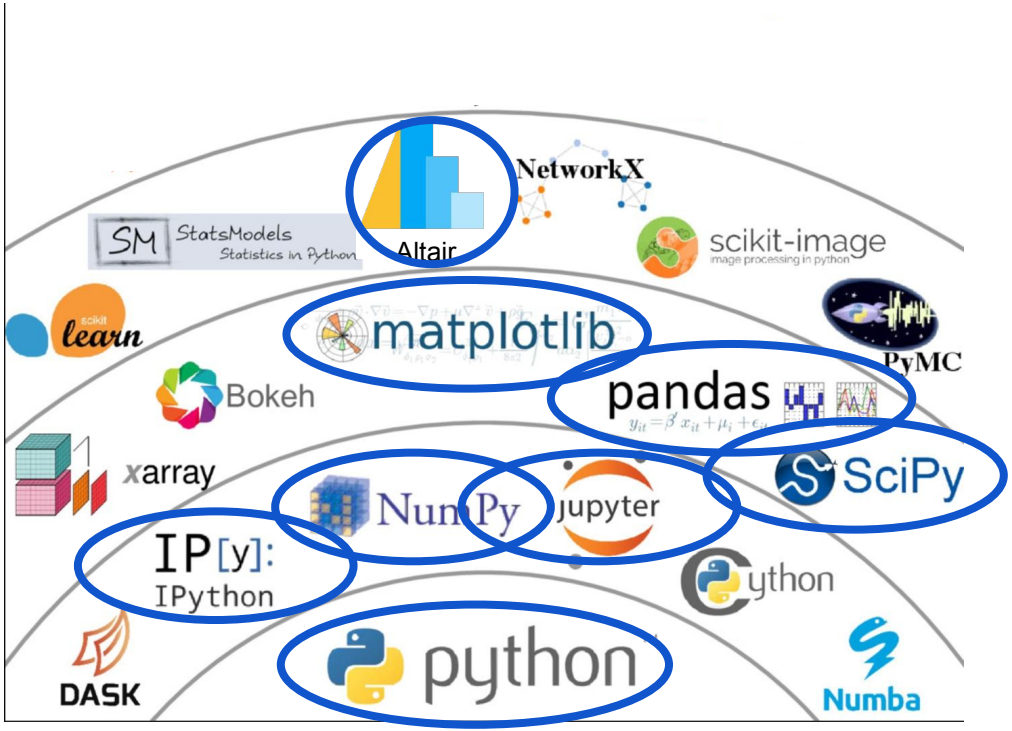
```python
# Create a dataframe with dates as your index
States = ['NY', 'NY', 'NY', 'NY', 'FL', 'FL', 'GA', 'GA', 'FL', 'FL']
data = [1.0, 2, 3, 4, 5, 6, 7, 8, 9, 10]
idx = pd.date_range('1/1/2012', periods=10, freq='MS')
df1 = pd.DataFrame(data, index=idx, columns=['Revenue'])
df1['State'] = States

# Create a second dataframe
data2 = [10.0, 10.0, 9, 9, 8, 8, 7, 7, 6, 6]
idx2 = pd.date_range('1/1/2013', periods=10, freq='MS')
df2 = pd.DataFrame(data2, index=idx2, columns=['Revenue'])
df2['State'] = States
```

```python
# Combine dataframes
df = pd.concat([df1,df2])
df
```

	Revenue	State
2012-01-01	1.0	NY
2012-02-01	2.0	NY
2012-03-01	3.0	NY
2012-04-01	4.0	NY
2012-05-01	5.0	FL
2012-06-01	6.0	FL
2012-07-01	7.0	GA
2012-08-01	8.0	GA

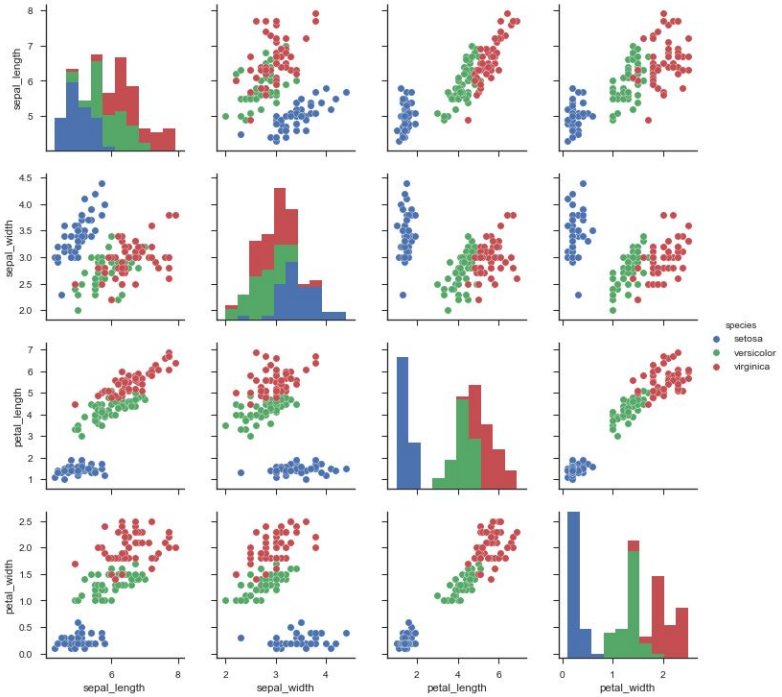


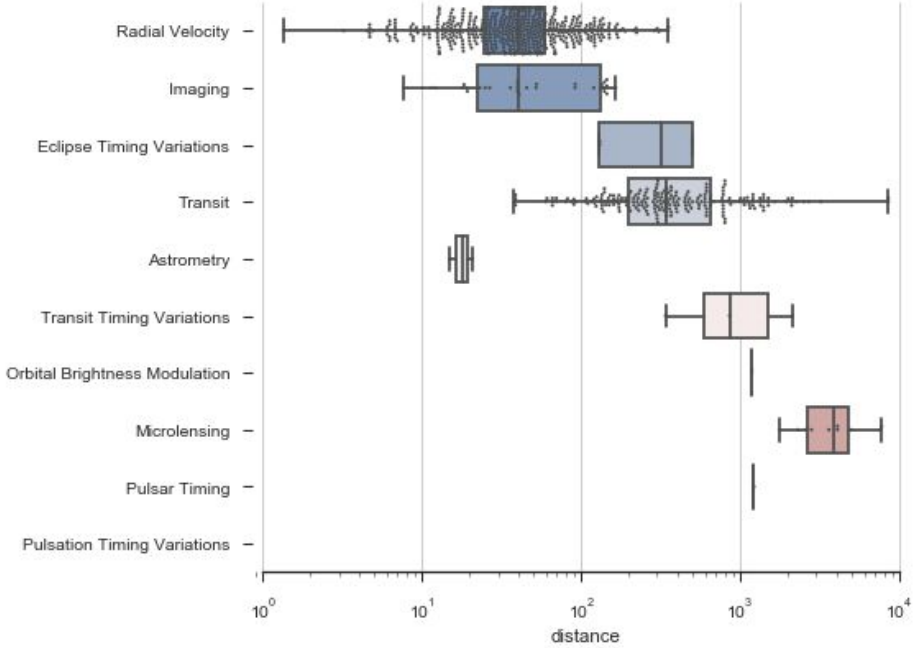
Our path through scientific python



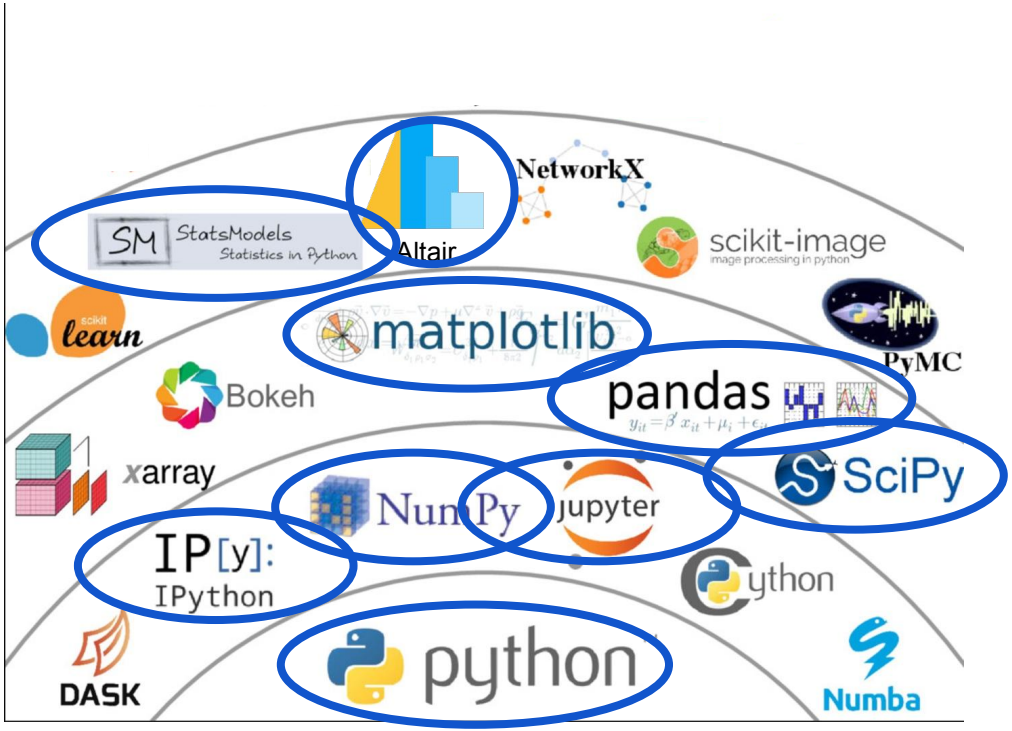
https://www.datacamp.com/community/blog/python-scientific-computing-case

Statistical visualization





Our path through scientific python



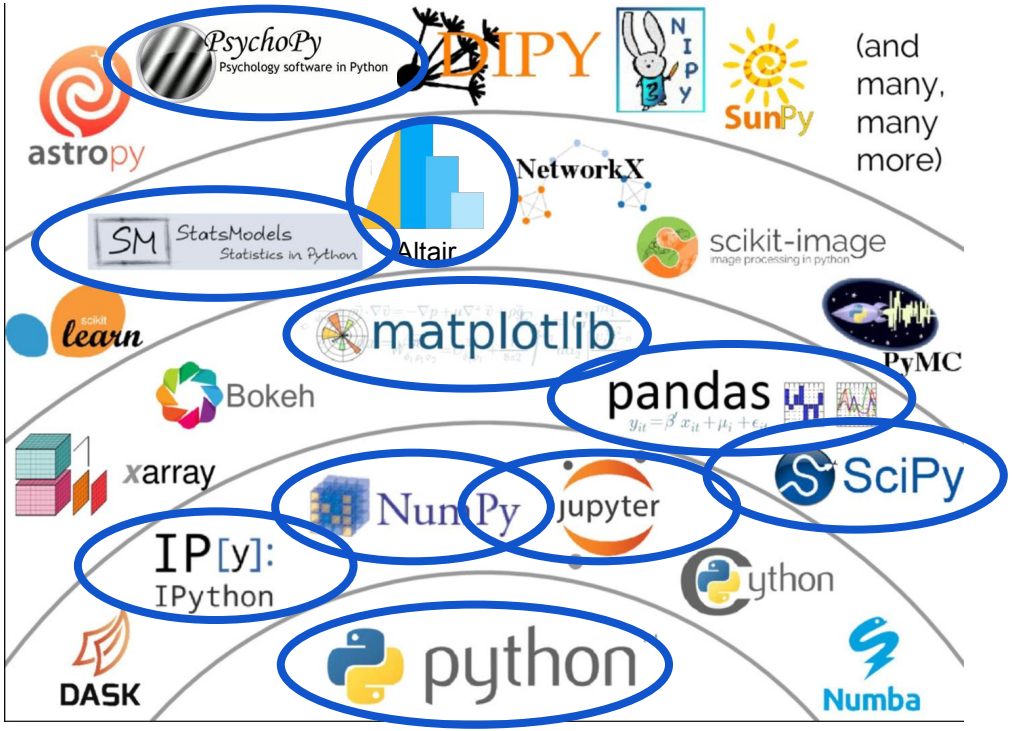
https://www.datacamp.com/community/blog/python-scientific-computing-case

Statsmodels

```
In [5]: results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)', data=dat).fit()
```

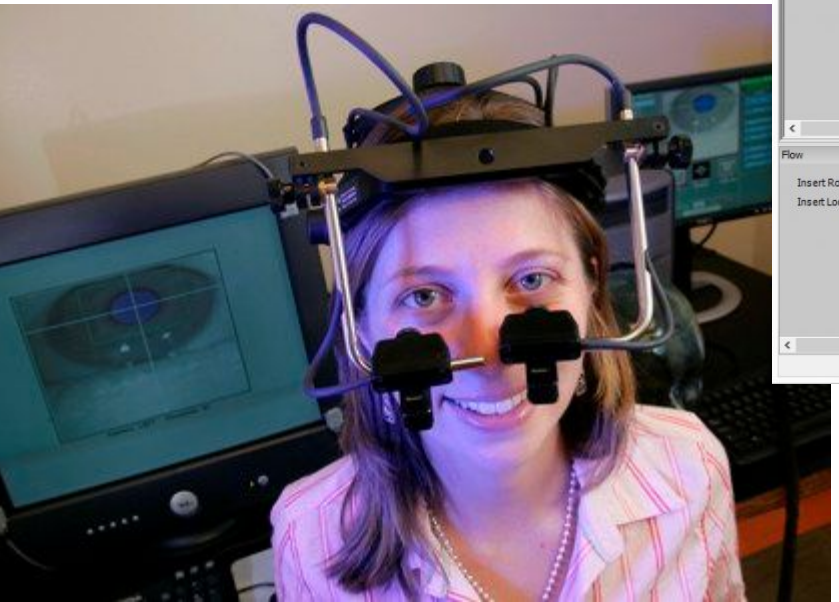
```
                            OLS Regression Results
==============================================================================
Dep. Variable:                Lottery   R-squared:                       0.348
Model:                            OLS   Adj. R-squared:                  0.333
Method:                 Least Squares   F-statistic:                     22.20
Date:                Tue, 28 Feb 2017   Prob (F-statistic):           1.90e-08
Time:                        21:38:05   Log-Likelihood:                -379.82
No. Observations:                  86   AIC:                             765.6
Df Residuals:                      83   BIC:                             773.0
Df Model:                           2
Covariance Type:            nonrobust
==================================================================================
                     coef    std err          t      P>|t|      [0.025      0.975]
----------------------------------------------------------------------------------
Intercept        246.4341     35.233      6.995      0.000     176.358     316.510
Literacy          -0.4889      0.128     -3.832      0.000      -0.743      -0.235
np.log(Pop1831)  -31.3114      5.977     -5.239      0.000     -43.199     -19.424
==============================================================================
Omnibus:                        3.713   Durbin-Watson:                   2.019
Prob(Omnibus):                  0.156   Jarque-Bera (JB):                3.394
Skew:                          -0.487   Prob(JB):                        0.183
Kurtosis:                       3.003   Cond. No.                         702.
==============================================================================
```

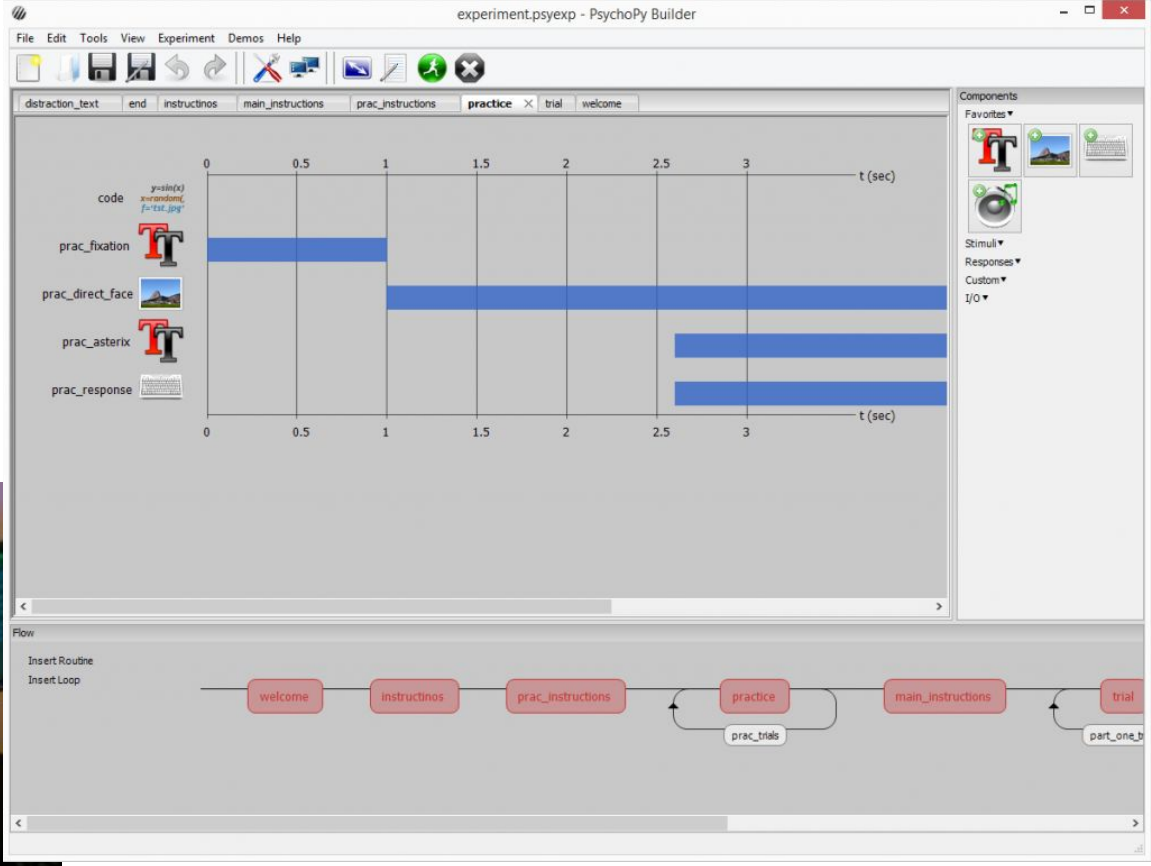
Our path through scientific python



https://www.datacamp.com/community/blog/python-scientific-computing-case

PsychoPy




experiment.psyexp - PsychoPy Builder
File Edit Tools View Experiment Demos Help
distraction_text
end
instructinos
main_instructions
prac_instructions
practice
trial
welcome
code
prac_fixation
prac_direct_face
prac_asterix
prac_response
t (sec)
Components
Favorites
Stimuli
Responses
Custom
I/O
Flow
Insert Routine
Insert Loop
welcome
instructinos
prac_instructions
practice
main_instructions
trial
prac_trials

Your workflow with Python

Extracting your data

NumPy

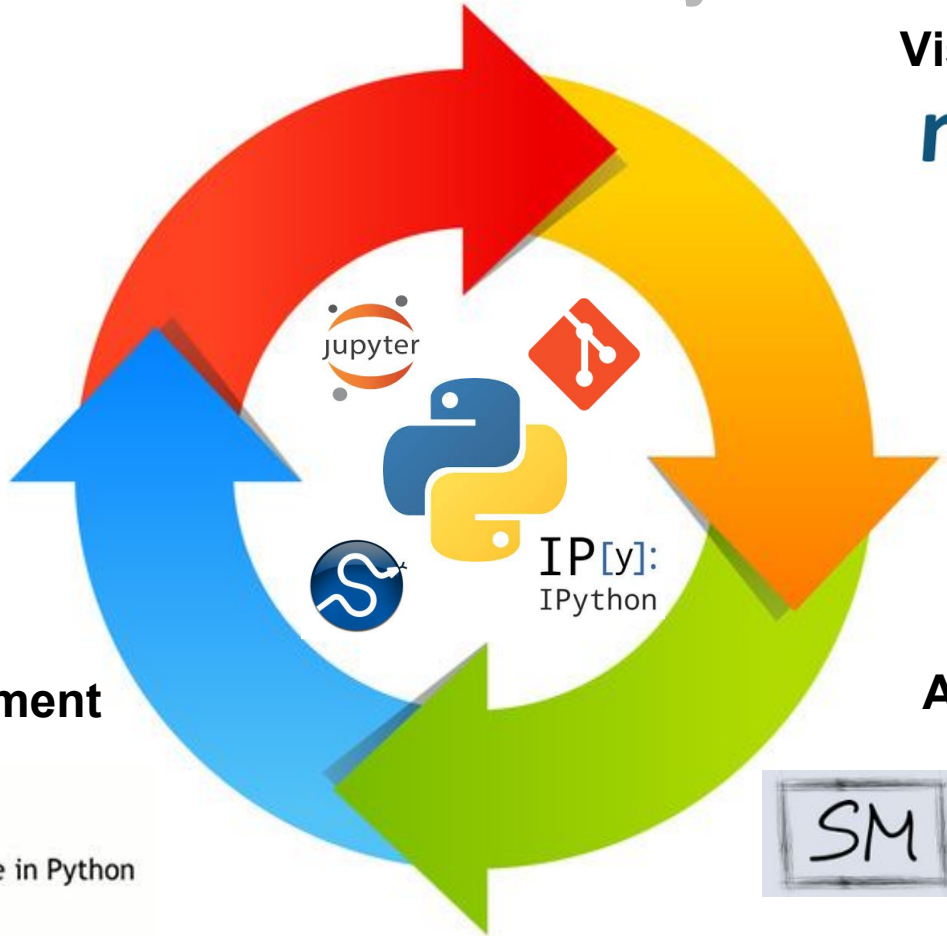
pandas

$y_i t = \beta' x_{it} + \mu_i + \epsilon_{it}$

Visualizing your data

matplotlib

plotnine



Making your experiment

PsychoPy

Psychology software in Python

Analyzing your data

SM StatsModels

Statistics in Python

Outline

1. Intro & Organization
2. Basic Python
3. Advanced Python
4. Numerical computing with NumPy
5. Visualizations with Matplotlib
6. Framing your data with Pandas
7. Cleaning data with Pandas
8. Analyzing with Pandas (or room for other stuff)
9. Statistical visualization with ggplot
10. Advanced statistics with statsmodels
11. Creating Experiments with Psychopy
12. Tools and other libraries (or room for other stuff)

Outline

1. Intro & Organization
2. Basic Python
3. Advanced Python
4. Numerical computing with NumPy
5. Visualizations with Matplotlib
6. Framing your data with Pandas
7. Cleaning data with Pandas
8. Analyzing with Pandas (or room for other stuff)
9. Statistical visualization with ggplot
10. Advanced statistics with statsmodels
11. Creating Experiments with Psychopy
12. Tools and other libraries (or room for other stuff)

Basic Programming in Python:

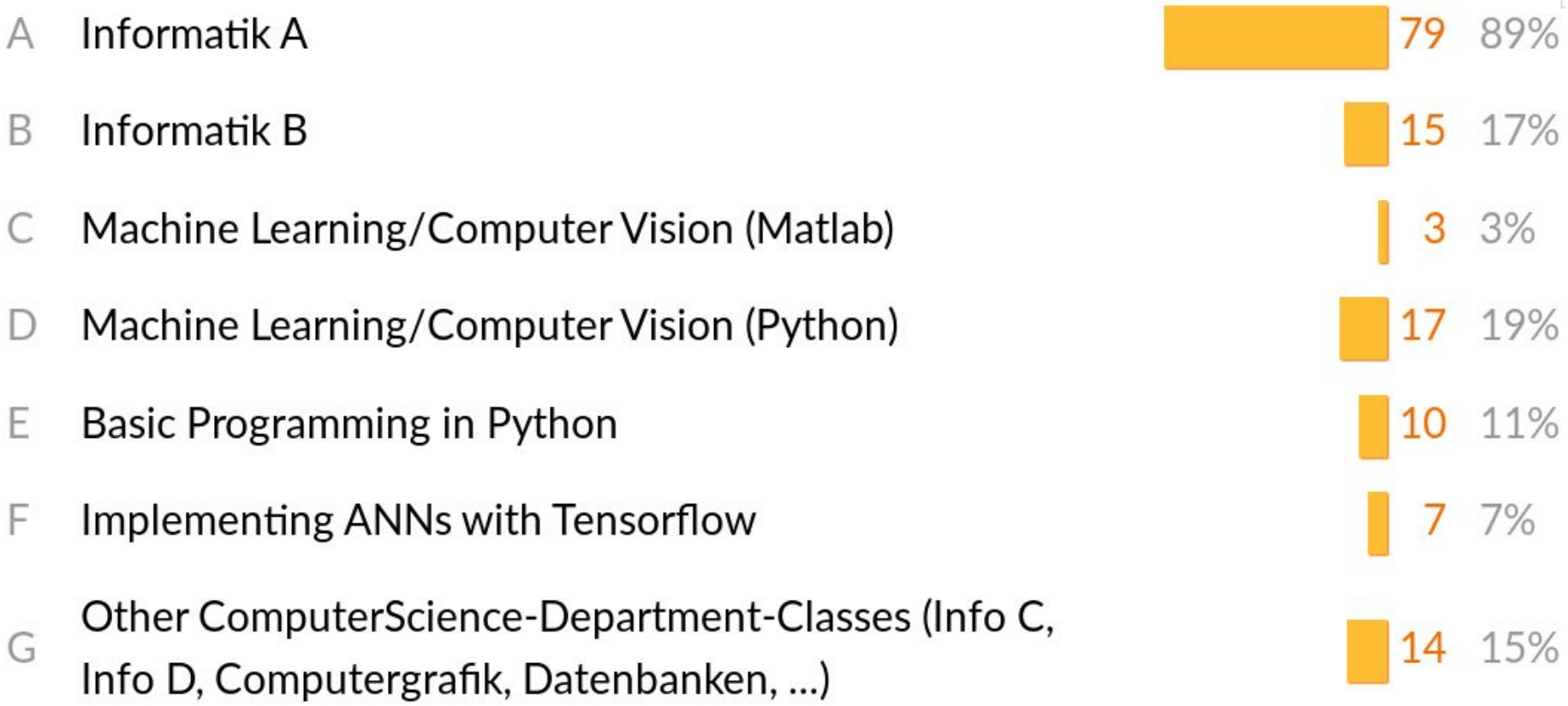
Structure

- **Week 1: Introduction**
- Week 2: Syntax & Variables
- Week 3: Control Structures
- Week 4: Lists & Collections
- Week 5: RegEx & Strings
- Week 6: Sorting & I/O
- Week 7:Debugging, Errors & Strategies
- Week 8: Python Packages
- Week 9: Practical Python & Good practices
- Week 10: Object Oriented Programming
- Week 11: Time, Space and documentation
- Week 12: Numpy & Matplotlib
- Week 13: Outlook & wrapping up
- Week 14: TBA

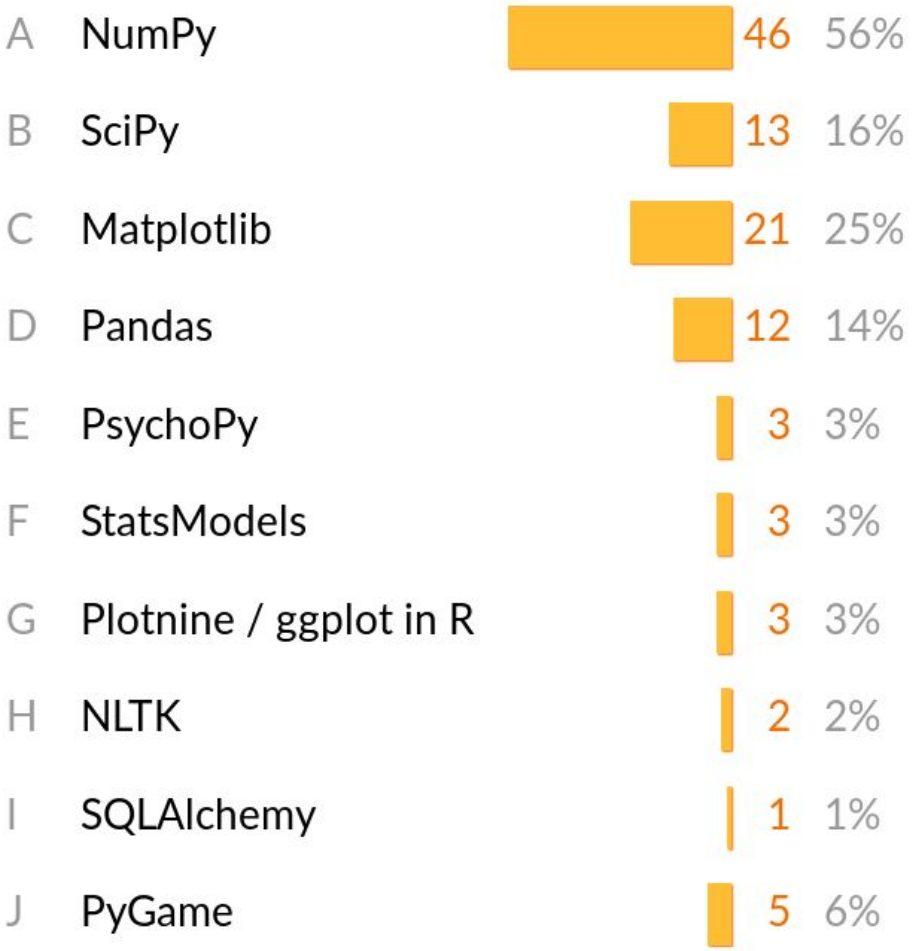
Cliqr

http://vt.uos.de/7j3zd

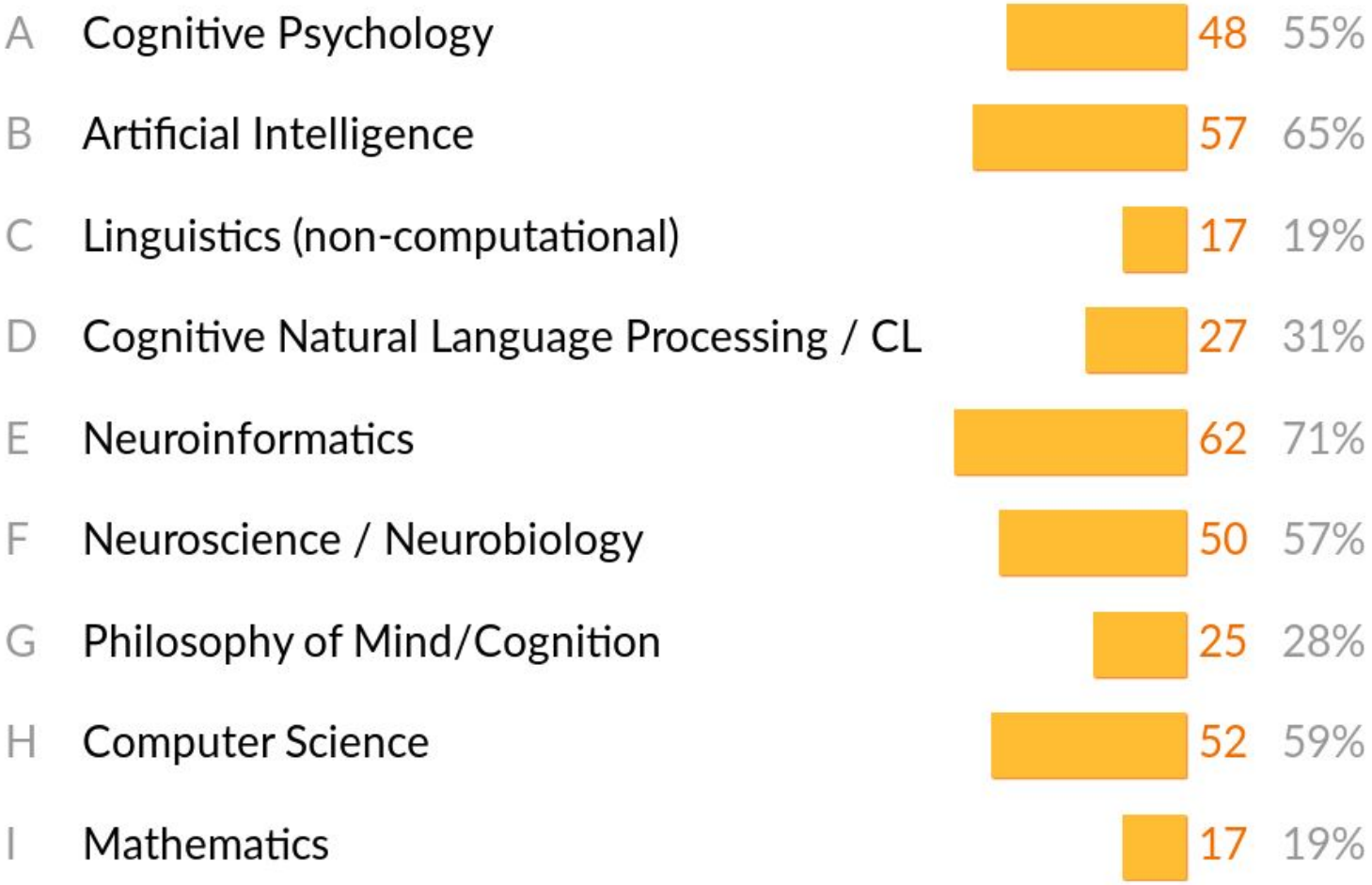
What programming-classes did you take so far?



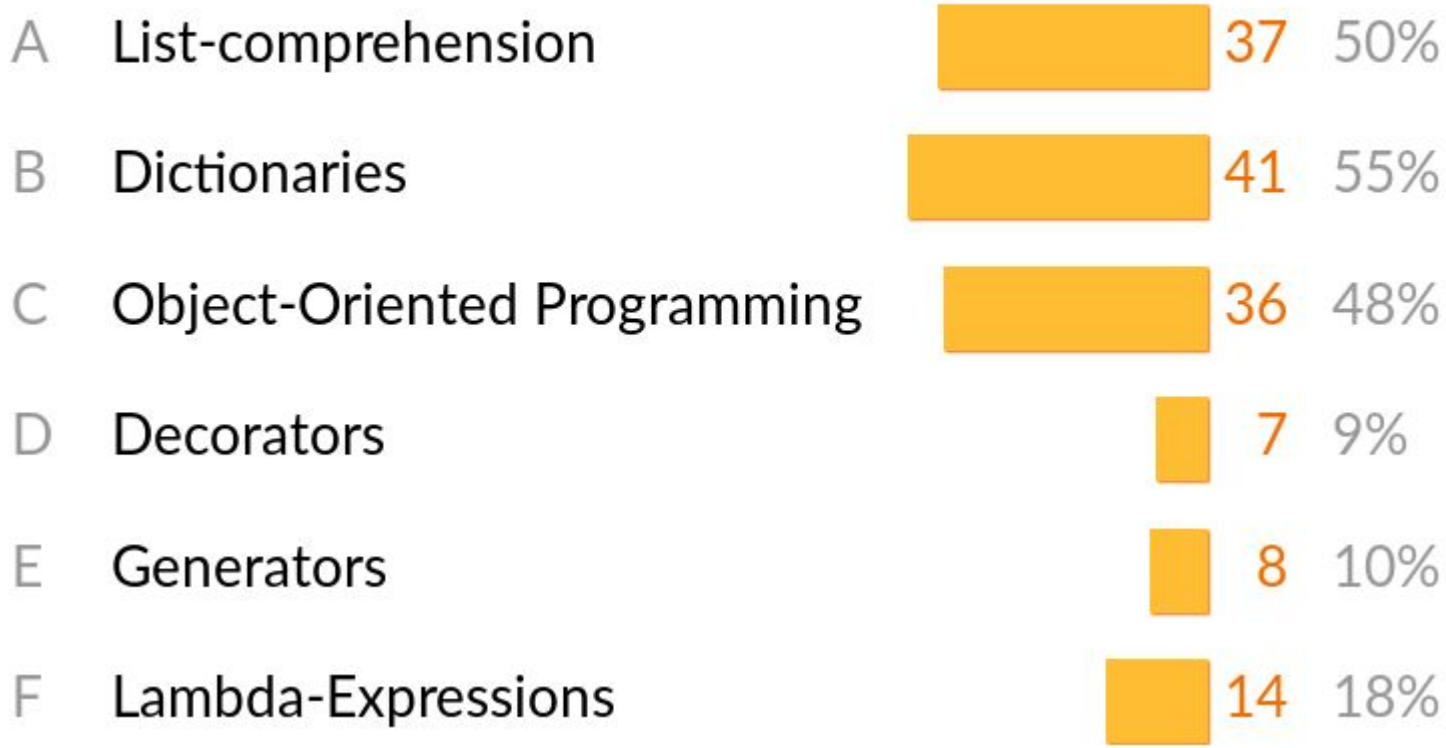
What libraries have you used so far?



What modules do you plan to take in your major?

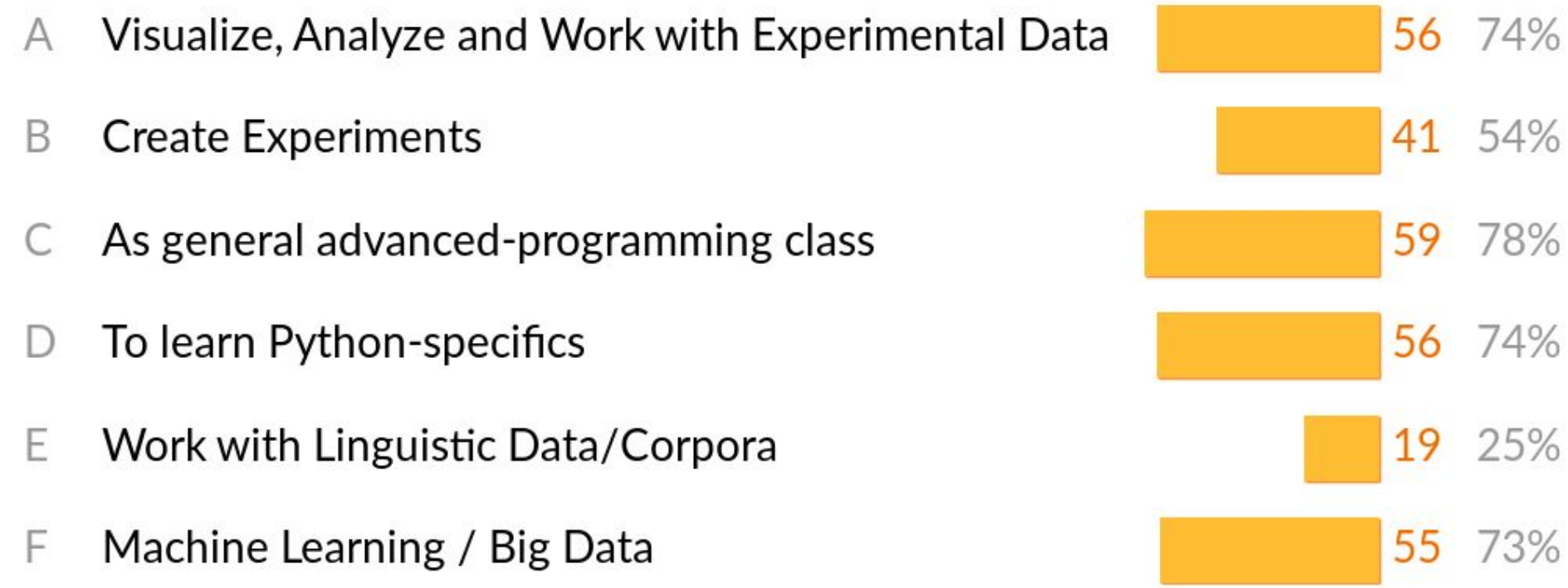


Which Python-Concepts did you use so far?

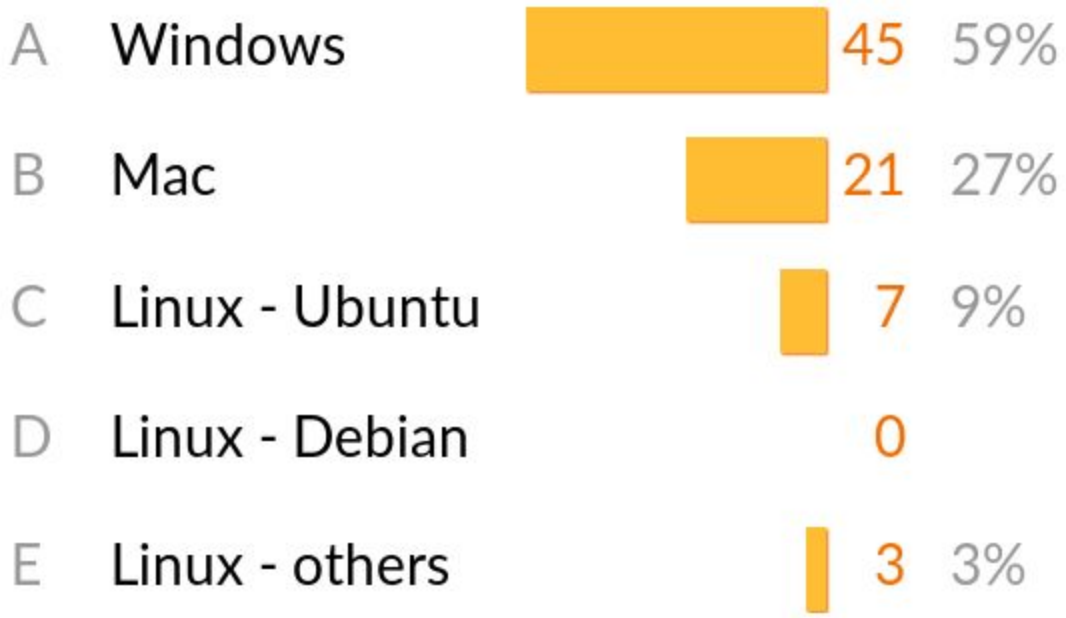


Do you have basic knowledge of...

What do you want to learn to code for?



Which Operating System do you use?



*We know these questions are not 100% exhaustive, as we for example missed the option “None of the above”. We think however they provide a good overview.

Setup for the course

Virtual environments

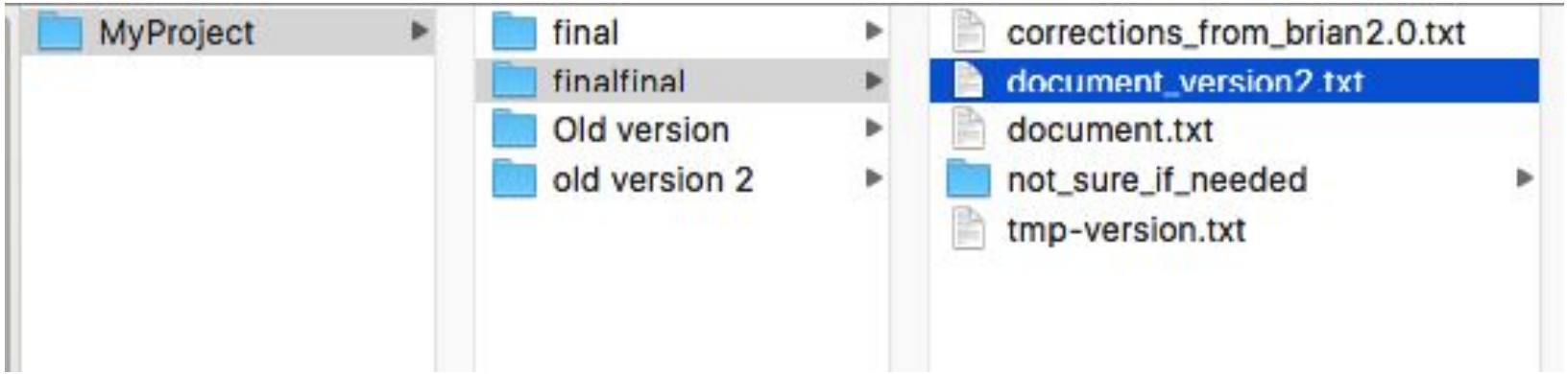
- Virtual environments allow you to have different Python versions with different packages side by side
- Working with the default Python leads to a mess or can even corrupt your operating system
- Conda is the easiest option to get Python virtual envs on all platforms
- Cheat sheet https://conda.io/docs/_downloads/conda-cheatsheet.pdf

[These slides are basing on the ones created by Lukas Kalbertodt, all credits to him]

Getting started with git

- Git is a free and open source VCS (version control system)
- It allows you to track changes to files over time and even have multiple versions in parallel
- VCS are commonly used for programming projects, but can also be useful for any other project

Why version control?



Version control

Tracks & logs changes in your files with…

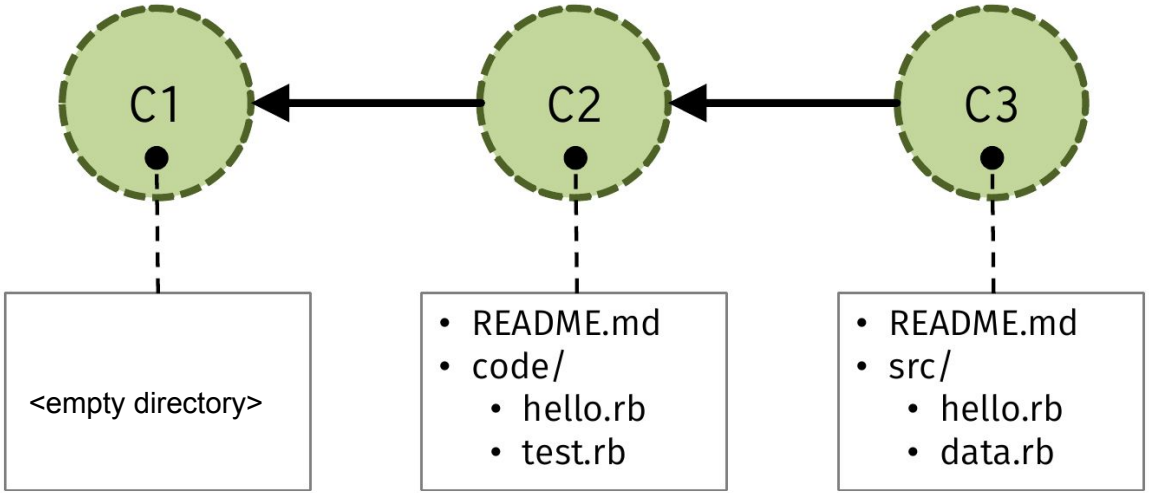
- Author
- Timestamp
- Description

Allows…

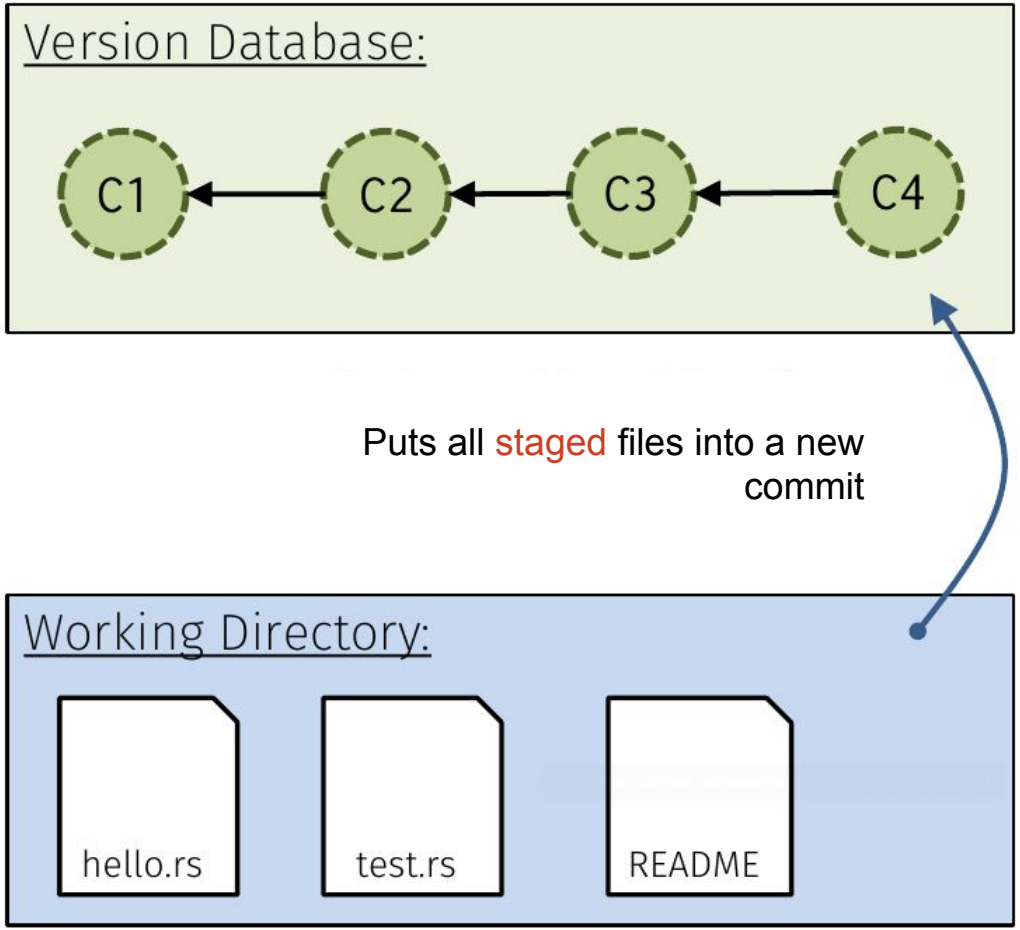
- Restoring old versions
- Having multiple parallel versions
- Analyzing your code

A commit

- Snapshot of the whole project at certain time
- A commit saves…
  - Its predecessor
  - Changes in the files (delta from predecessor)
  - Author, time, commit message
- Identified by Hash (eg. C1, C2, ..)



The repository



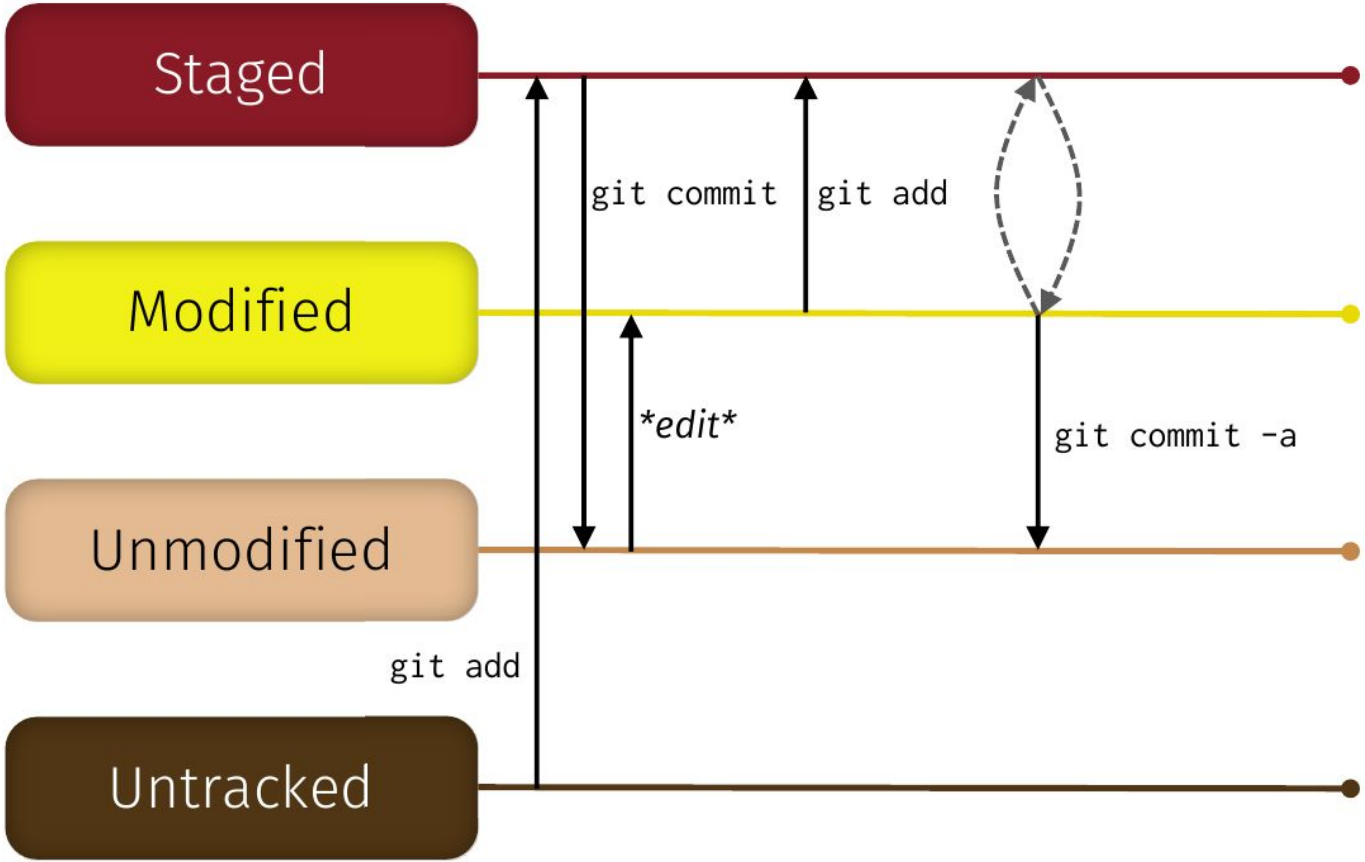

```
git commit
```

- Contains all commits
- Saved in a hidden folder called .git

File status

Staged	File will be committed with the next commit
Modified	File is registered for git and was changed since the last commit
Unmodified	File is registered in git, but equal to the last commit
Untracked	Git knows the file exists, but won’t do anything with it

File status


Staged
Modified
Unmodified
Untracked
git commit
git add
edit
git commit -a
git add

git status

Staged

Modified

Unmodified

Untracked

```
$ git status
On branch dev
Your branch is up-to-date with 'origin/dev'.

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        new file: bye.rs
        new file: hello.rs

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified: Cargo.toml

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        test.rs
```

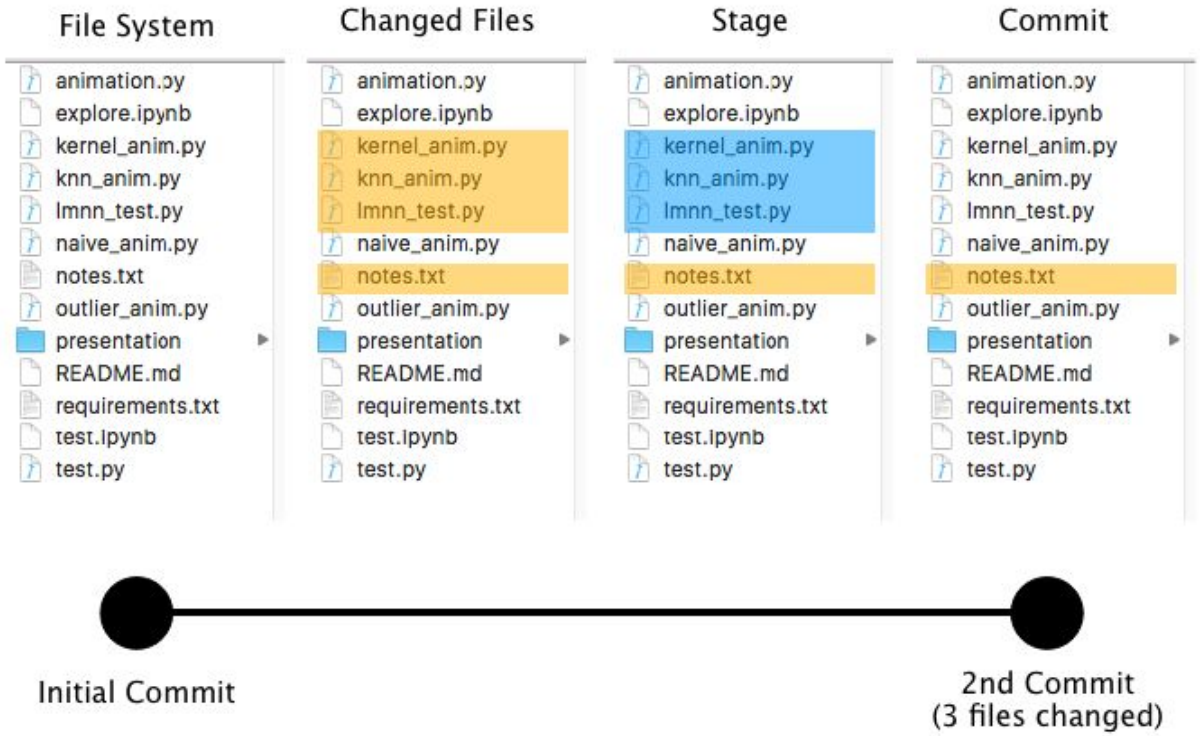
Run *git status* before any other command to know what's going on!

Git workflow

File System

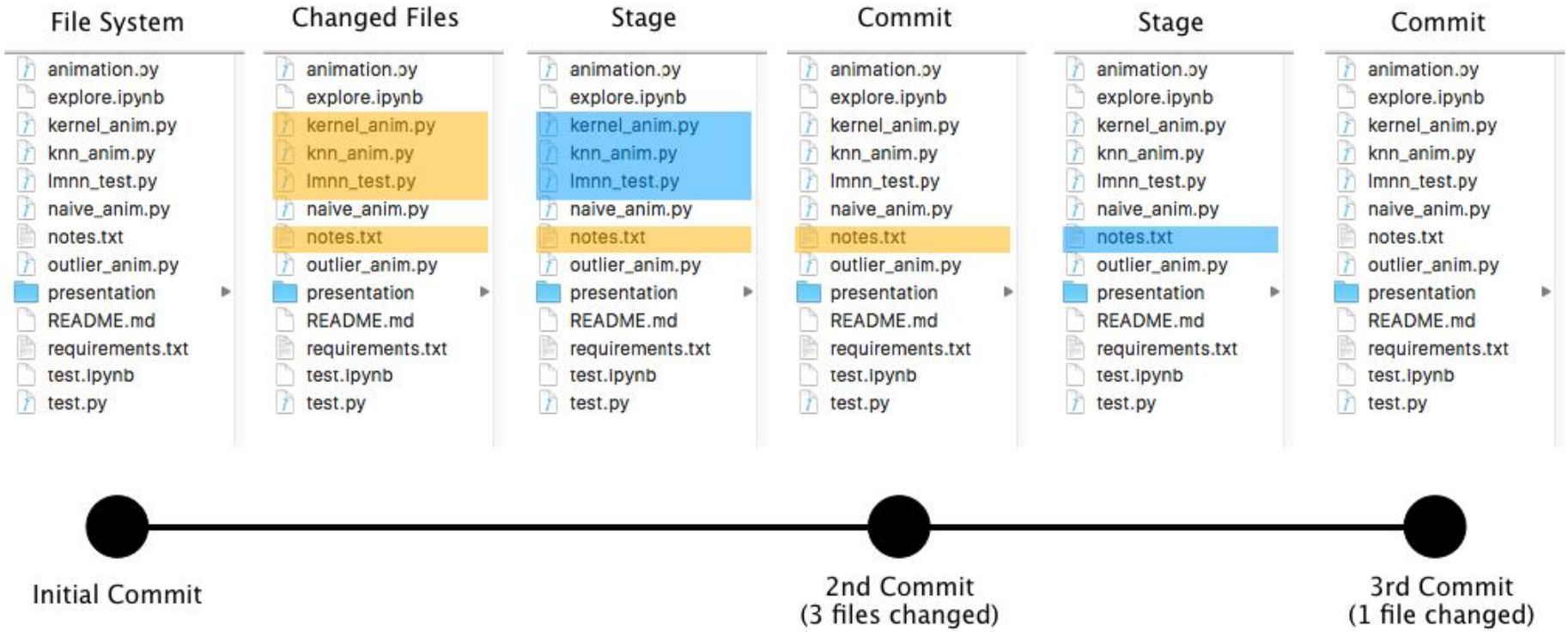
- animation.py
- explore.ipynb
- kernel_anim.py
- knn_anim.py
- lmnn_test.py
- naive_anim.py
- notes.txt
- outlier_anim.py
- presentation
- README.md
- requirements.txt
- test.ipynb
- test.py

Git workflow



https://try.github.io/

Git workflow

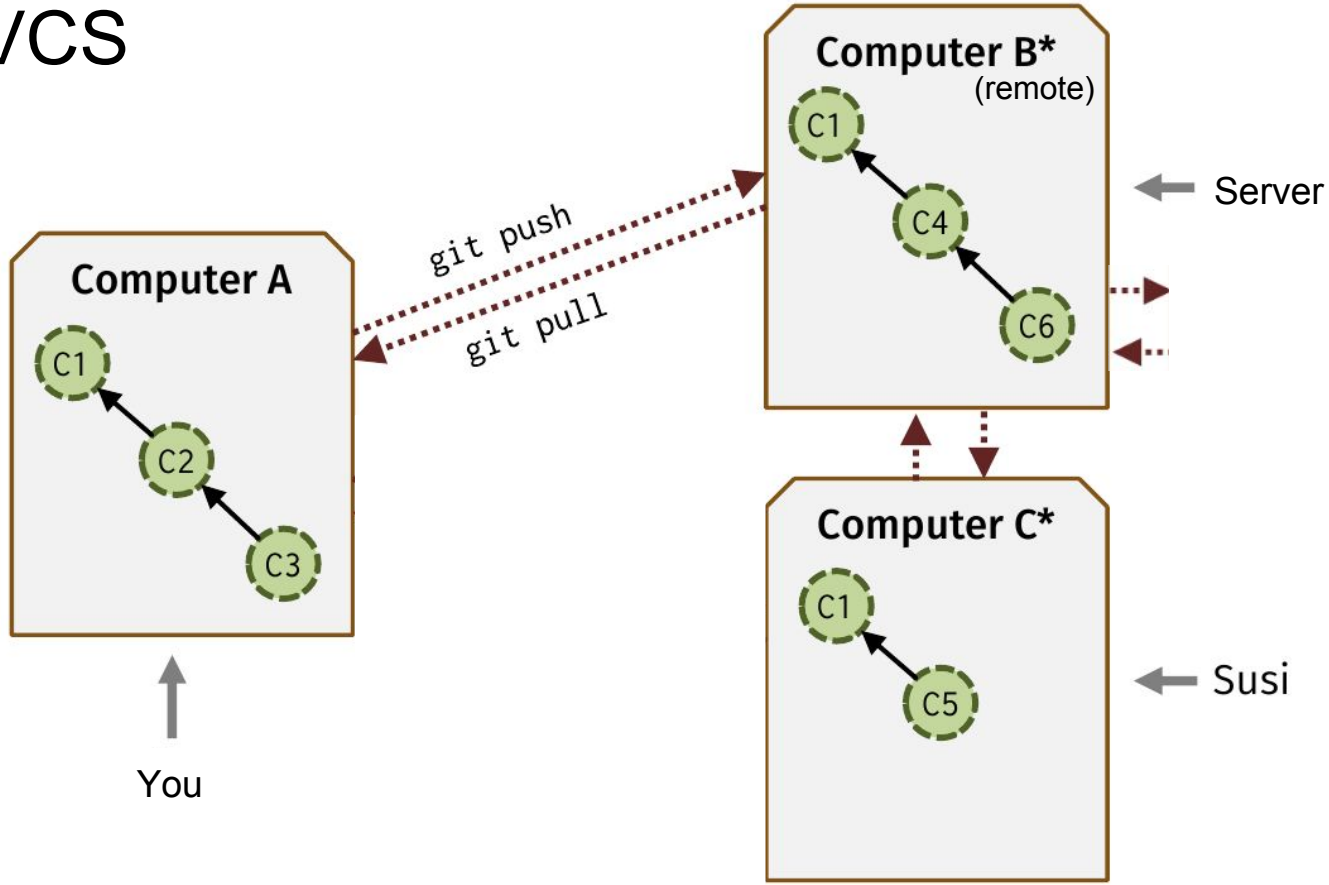


Getting started with GitHub

- Github is a website, providing a server to store your projects and also a web-interface to easily access them and to collaborate with others
- GitHub adds many additional features, like Pull-Requests with Code review, Issues, Wikis, …

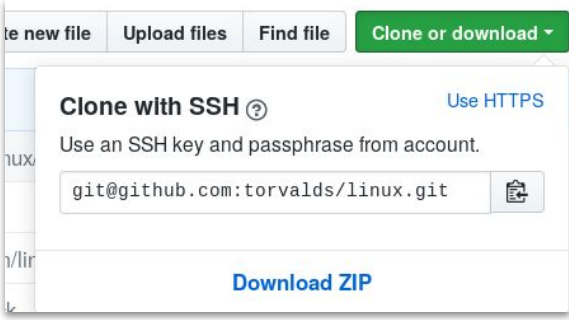
- Github is only an *external storage (+UI)* for git projects
- Git works perfectly fine locally, without GitHub
- Github allows for easy access from multiple computers
- You need to manually synchronize your local directory with GitHub!

Distributed VCS



<Demo: git>

What to do



1. Create a new repository
   - *git init* PROJECT1
   - Creates a new folder PROJECT1
   - *cd* PROJECT1

OR

1. Clone a repository
   - *git clone GIT_URL*
   - Creates a local copy of the repository and adds URL as *remote* **origin**

2. Add some files
   - *echo "hallo" > file.txt*
   - *ls* — Folder contains *file.txt*
   - *git status* — Untracked files: “file.txt”
   - *git add file.txt*
   - *git status* — Changes to be commited: “file.txt”
   - *git commit -m “added a file”*
   - *git status* — nothing to commit, working tree clean

3. Push local changes
   - (Create repo: https://github.com/new
   - *git remote add origin* GIT_URL)
   - *git push origin master*

Git commands cheat-sheet

- `git init` to create a new project
- `git clone <url>` to copy an existing project from eg. GitHub or BitBucket
- `git status` to view which files changed in status
- `git diff` to view each file's difference to the last commit (or also between commits)
- `git checkout <file>` to reset a file to the last commit
- `git checkout -b <branch> <hash>` to completely restore an older commit (to a branch)
- `git checkout <branch>` to switch branches (eg. back to master)
- `git log` to view your latest commits incl. messages
- `git pull` to update your local repository to the state of the one on GitHub/BitBucket
- `git push` to update the repository on GitHub/BitBucket to your local version
- `git add <file>` such that git will stage the file to be considered in the next commit
- `git rm <file>` to delete a file from filesystem and also stop tracking it
- `git commit -m "<message>"` to create a commit of the currently staged files

Final remarks on git

- Git is made for source-code and is no dropbox! → Add only text-based (diff'able) files
- Gitignores may help to not add unnecessary files. Add a ".gitignore" file to your repository and write filename-masks you want git to completely ignore into it. A useful start is https://github.com/github/gitignore/blob/master/Python.gitignore
- Modern editors all come with git integration, however we advise to not use it until you're really familiar with git! Learning the console always helps!
- If you are not familiar with git https://try.github.io/
- For deeper looks https://git-scm.com/book/en/v2

Which IDE

Save yourself the pain and don't code in notepad, without any kind of syntax highlighting or code completion

- Atom https://atom.io/ (or for linux *apt-get install atom*)
  - Useful Packages: **Hydrogen** by *nteract* & **hydrogen-launcher** by *lgeiger*
  - Free and open source
  - Recommended for smaller projects (containing only few files)
- Pycharm https://www.jetbrains.com/pycharm/
  - Commercial, closed source, however Community edition free and Professional free as student
  - Features many components - debugger, code analysis features, git integration, ...
  - Useful for big projects, not recommended for homework of this course
- vi
  - Integrated into Unix-systems, runs *inside* the terminal
  - Hard to master, but supposedly much faster once you did
- Jupyter Notebook
  - Does not only contain code, but also formatted text and results of running your code
  - Makes for pretty notebooks that can be exported to HTML or PDF, or simply to a python-script
  - Not recommended for this class, as files are hardly diff'able and can't be handled by pytest!

Homework

- Homework is distributed via *Github classoom*
- You need a github-account to submit the homework!
- If there are mistakes in the homework, we will announce that and update the repositories, so regularly pull!

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

JarnoRFB Go back to single env for all sheets — Latest commit 796a5ad 4 hours ago

.gitignore	Makes sure you (and we) don't commit unnecessary stuff	5 hours ago
.travis.yml	Necessary to automatically correct your homework	5 hours ago
README.md	Contains the task description (seen below)	4 hours ago
hello_world.py	Contains the barebone structure of the task you'll solve	5 hours ago
requirements.txt	List of packages you'll need in your environment for this task	5 hours ago
test_hello.py	The testing-files you and we use to see if your program works as intended	5 hours ago

README.md

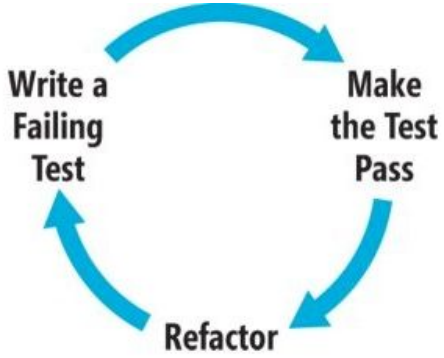
Homework 01

The purpose of this exercise is mainly to get you all set up, install python and git the correct way, and practice some git. You can follow the instructions in this file without downloading the repository yet, as you'll probably need to get git and Python first.

Pytest and Test-Driven Development

- `pytest` is a testing library included with python
- It will grab all `test_`-functions in all `test_`-files, execute them, and check for errors
- To do so, it uses assertions: `assert prime.find_prime(1)==2`

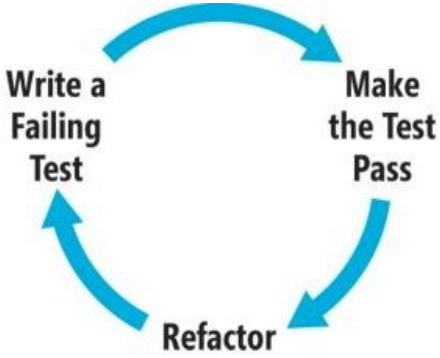
```
chris@debian:~/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01$ pytest
========================================= test session starts =========================================
platform linux -- Python 3.6.5rc1, pytest-3.5.0, py-1.5.3, pluggy-0.6.0
rootdir: /home/chris/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01, inifile:
collected 2 items

test_prime.py .F                                                                                [100%]

================================================= FAILURES ==================================================
_____________________________________________ test_find_prime_method _____________________________________________

    def test_find_prime_method():
        assert hasattr(prime, 'find_prime'), "Your Script must have a 'find_prime'-method!"

        assert prime.find_prime(1) == 2
>       assert prime.find_prime(8) == 19
E       assert 11 == 19
E        +  where 11 = <function find_prime at 0x7f8407982bf8>(8)
E        +    where <function find_prime at 0x7f8407982bf8> = prime.find_prime

test_prime.py:19: AssertionError
================================== 1 failed, 1 passed in 0.02 seconds ==================================
chris@debian:~/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01$
```



Pytest and Test-Driven Development

- `pytest` is a testing library included with python
- It will grab all `test_`-functions in all `test_`-files, execute them, and check for errors
- To do so, it uses assertions: `assert prime.find_prime(1)==2`

```
chris@debian:~/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01$ pytest
========================================= test session starts =========================================
platform linux -- Python 3.6.5rc1, pytest-3.5.0, py-1.5.3, pluggy-0.6.0
rootdir: /home/chris/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01, inifile:
collected 2 items

test_prime.py ..                                                                                [100%]

====================================== 2 passed in 0.03 seconds ======================================
chris@debian:~/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01$
```



Once this is the result of pytest, your homework will pass!

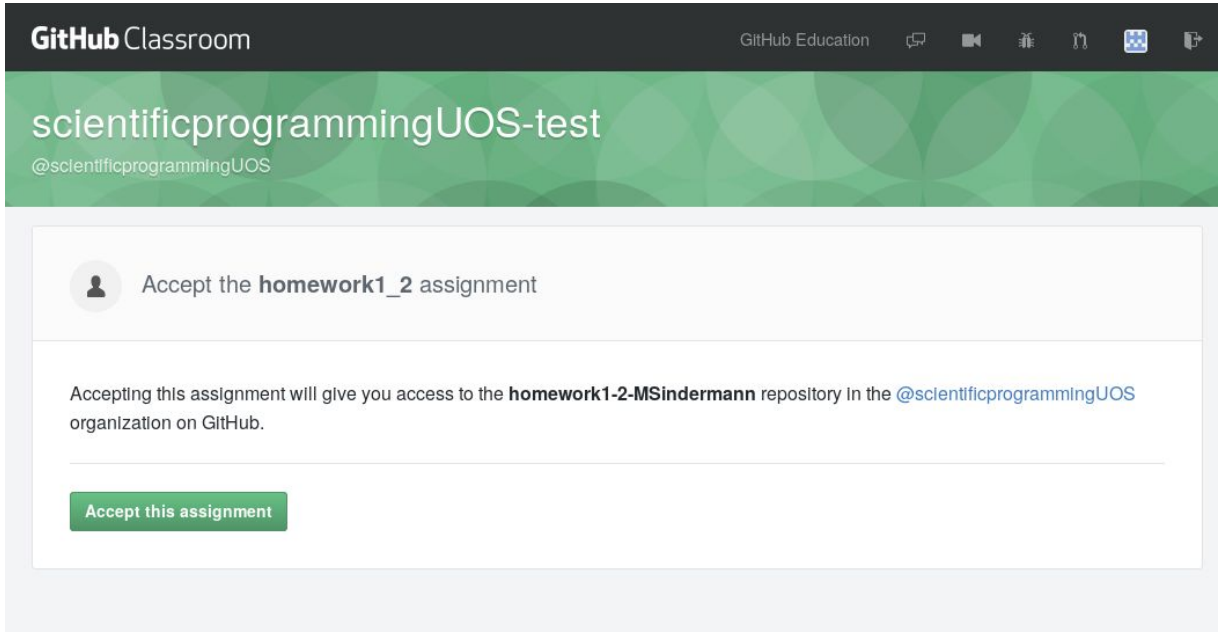
Your homework

- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

<demo: homework, virtualenv, atom>

How to download and work on Homework

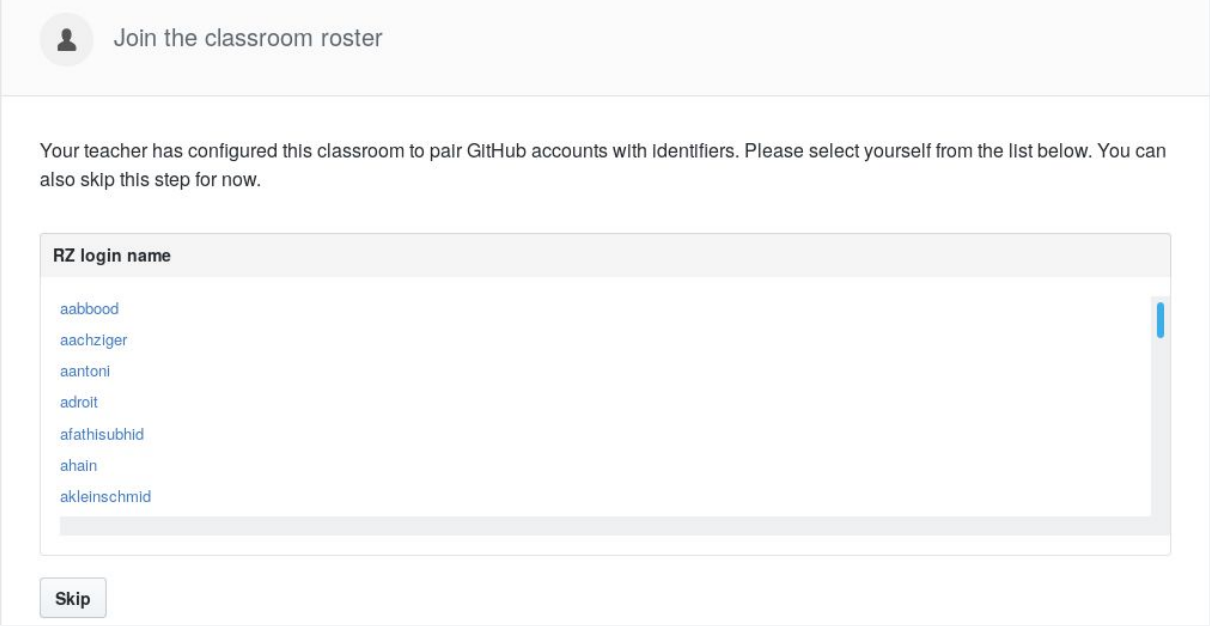
- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)



- You need to sign in to github
- Use your @uos-mail to get unlimited free repositories! https://education.github.com/pack/offers

How to download and work on Homework

- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

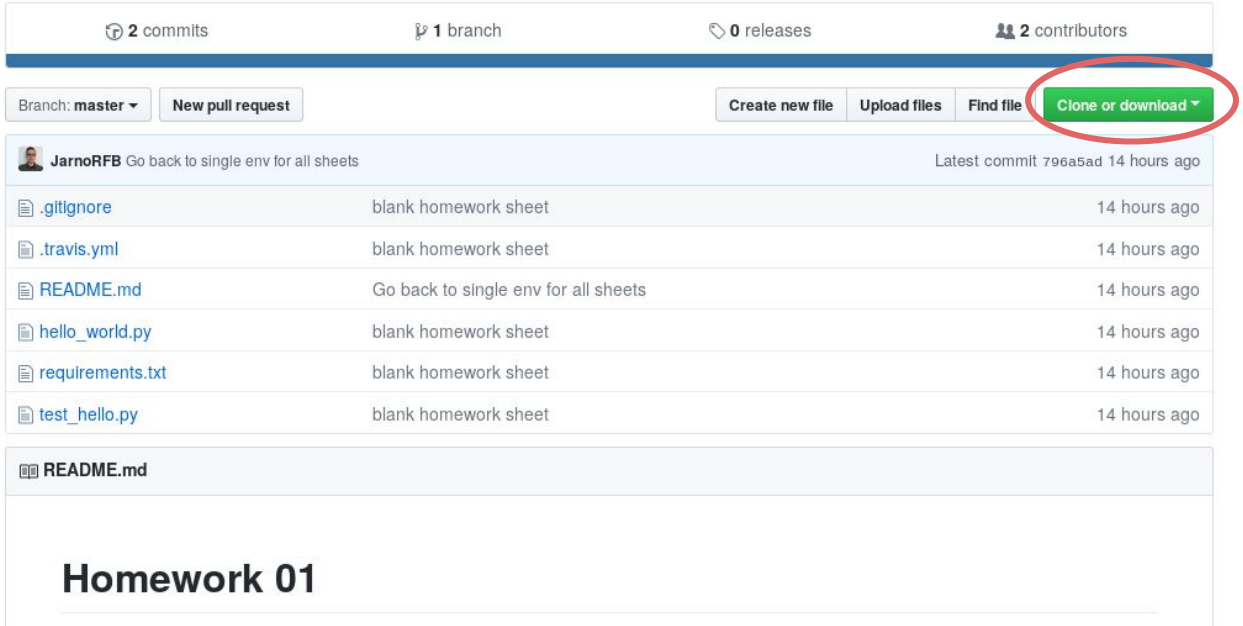


- If your RZ-login is not listed here or spelled incorrectly, please write us an email!
- When it says *"preparing your new repository, there is no need to keep this window open, we'll email you when the import is done"*, just hit F5 after 5 seconds

How to download and work on Homework

- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

homework-1-nelssner created by GitHub Classroom



- If your RZ-login is not listed here or spelled incorrectly, please write us an email!
- When it says *"preparing your new repository, there is no need to keep this window open, we'll email you when the import is done"*, just hit F5 after 5 seconds

How to download and work on Homework

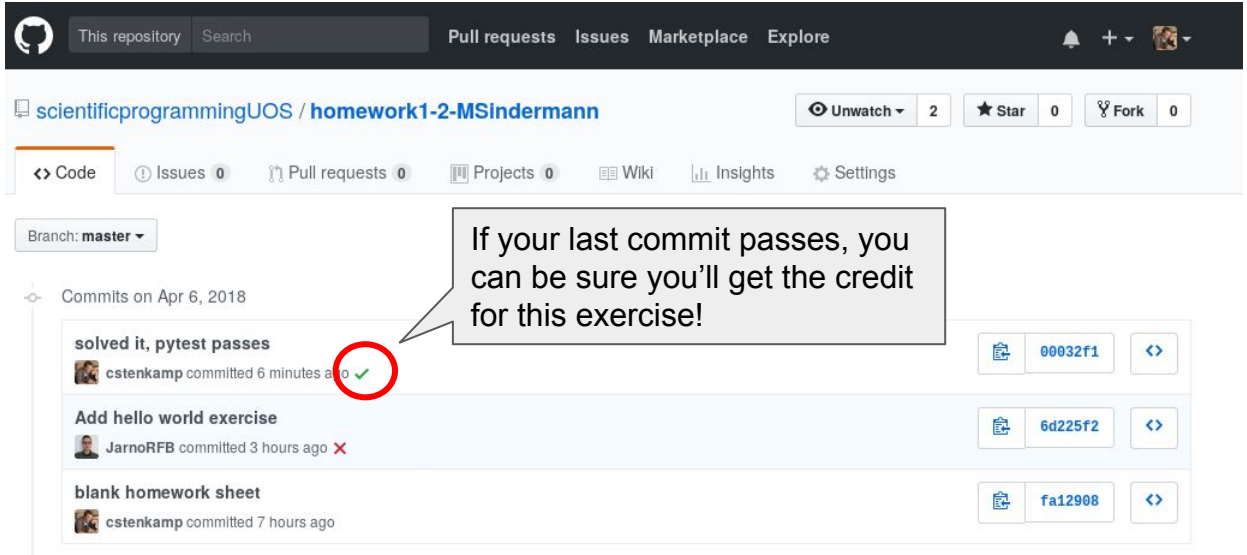
- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

```
git clone REPOSITORY_URL
cd YOUR_REPOSITORY_PATH
conda create -n scientificpython --file requirements.txt python=3.6
source activate scientificpython  (on linux)
...
pytest
...
git status
git add CHANGED_FILE
git commit -m "solved the exercise"
git push origin master
source deactivate  (on linux)
```

How to download and work on Homework

- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

- `git config --get remote.origin.url` tells you the domain of your remote repository



- Note that the first check will be performed 5-15 minutes after accepting the exercise, from then on ASAP

How to download and work on Homework

- Homework-links:
  - https://classroom.github.com/a/hl2V5_56 (1st regular)
  - https://classroom.github.com/a/tokG-j1Q (1st bonus)

- Note that the first check will be performed 5-15 minutes after accepting the exercise, from then on ASAP
- First deadline is **next monday, 14:00** (hard)
- Clone the repository >15 minutes before the deadline passes, make the last commit before the deadline passes
- Yes, you get the test-scripts, but don't change them, we check for that!
- You'll get an email telling you if you passed or failed, including your overall pass/fail-count right after the next lecture!

Thanks for your attention!

- We will have have a feedback-questionnaire after 4-5 sessions
- Any questions and remarks please via email!
- Content-suggestions are always welcome!

Thanks to Patrick Faion and Brian Lewis for the old *Scientific Programming in Python* lecture
Thanks to Lukas Kalbertodt for the git lecture