

# Introduction

Scientific Programming in Python

# Organizational matters

# About us

Philipp

- 6th semester Bachelor
- 4 years of Python experience
- [pthoelke@uos.de](mailto:pthoelke@uos.de)

Chris

- some year in the Master
- 5 years of Python experience
- [cstenkamp@uos.de](mailto:cstenkamp@uos.de)

- If you have any questions, 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 the slides or the homework, please do so via a github-issue! The repository for homework is  
<https://github.com/scientificprogrammingUOS>

# Date and Time

## During Corona

- Lecture-Videos uploaded on Tuesday
  - Split into multiple parts available in Courseware
  - Including small exercises to test your attention
  - Videos should be 90 minutes in length
- We will try live-Meetings Thursday 14:00 - 16:00
  - These Meetings are for your questions
  - We will also answer any questions you ask via Mail/Forum/...
  - **Ask questions!** (see “*Infrastructure*” for where)

**This lecture is for you!**

Ask questions when you don't understand something,  
everything else is a waste of your time!

# Date and Time

Next Thursday

- 14:00 - 15:00: Live Stud.IP Meeting (non-mandatory)
- 15:00 - 18:00: Available for private conversations in Skype for individual (installation-) problems
  - Philipp: <https://join.skype.com/invite/q1r9ULtOTxGO>
  - Chris: <https://join.skype.com/invite/mT3yXuxQESAf>

# Date and Time

After Corona

- One **session** every *Tuesday 12:00 - 14:00*
- One **session** every *Thursday 14:00 - 16:00*
- No mandatory attendance
- Lecture will still be filmed

Approach:

- Lectures will be a mix of concepts, coding tutorial and interactive exercises

# Organizational matters

## Modulzuordnungen

Bachelor of Science Cognitive Science > CS-BWP-MCS - Methoden der Kognitionswissenschaft, gültig ab WS 2019/20 

Bachelor of Science Cognitive Science > KOGW-PWB - Profilbildender Wahlbereich 

- New Examination Regulations:
  - 4 Credits for the “Methods of Cognitive Science” Module
  - You need 12 Credits to fulfill this module, 8 of which need to have a grade
- Old Examination Regulations:
  - 4 Credits for “Profilbildender Wahlbereich” (your 30 free credits)
- **Complete enough homework to get a “pass”,  
Additionally write the exam to get a grade**
  - Exam: On the computer (hopefully in the same room), coding tasks
  - Open book exam, use any source you want, time will be your constraint
  - *Not automatically graded*

Questions regarding the new  
examination regulations to  
[mentoring@cogsci.uni-osnabrueck.de](mailto:mentoring@cogsci.uni-osnabrueck.de)

# Homework

- Weekly coding homework, done individually
- Homework corrected automatically, and you get the tests, too!
- 1-5 Tasks per sheet, maximally 25 points per sheet
  - No partial points per task!
- 12-13 Sheets →  $12 \cdot 25 = 300$  points maximally
  - **Reach 200 points ( $\frac{2}{3}$ ) to get the Schein**
  - Some homework will give less points, so there will be bonus exercises to make sure there are 300 points to reach

More info on how to work on the homework later!

# Intro

# Why scientific programming?

# Science

- Build and organize knowledge
- Test explanation about our world
- Communicate our results to others
- Systematically
- Objectively
- Transparently
- Reproducibly

Otherwise it's not science.

# Programming helps us

- Build and organize knowledge → by building databases of scientific results
- Test explanation about our world → by automating experiments
- Communicate our results to others → by sharing code on top of papers
- Systematically → by easily making analyses exhaustive
- Objectively → computers are not subject to human biases (computer programs still are)
- Transparently → by using open source tools and sharing our analyses
- Reproducibly → by codifying our analyses we make them reproducible

You need to write **clean code!**

# Why Python?

# Python

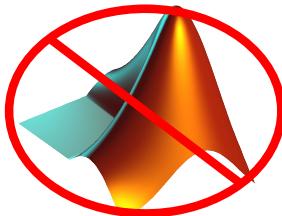
- Create by Guido van Rossum in the 90s
- Now open source project developed by the Python Software foundation
- High-level language (no hardware-knowledge necessary)
- Interpreted and dynamically typed language
- Consistent and minimal syntax
- → easy to learn and write
- Great ecosystem and great community!





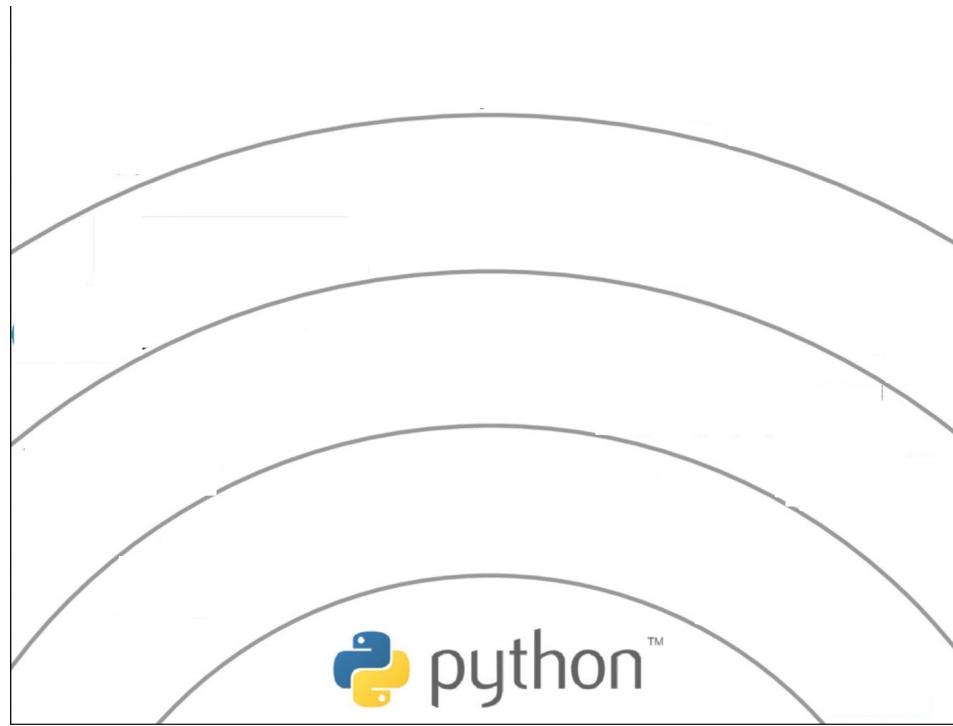
# Python is better

- Better than Matlab → As a free and open source project you can save money and actually share your results
- Better than Java → Get more done with less code and without overly complex object orientation.
- 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 R → As a general purpose programming language you can do anything with Python not just statistics

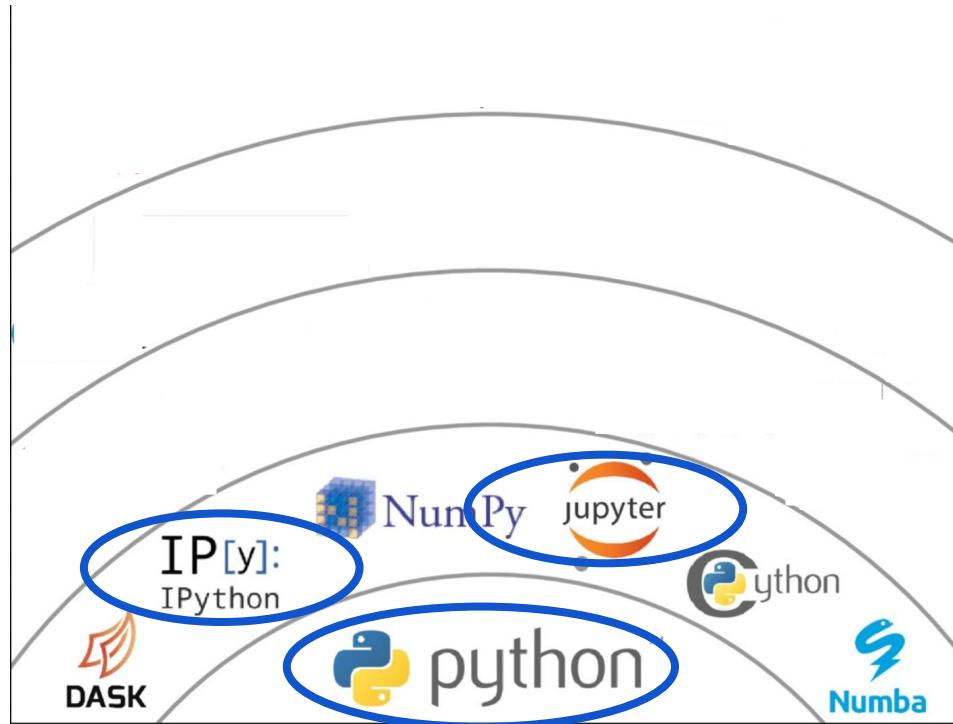


# Our path through scientific python

# Our path through scientific python



# Our path through scientific python



# 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 *

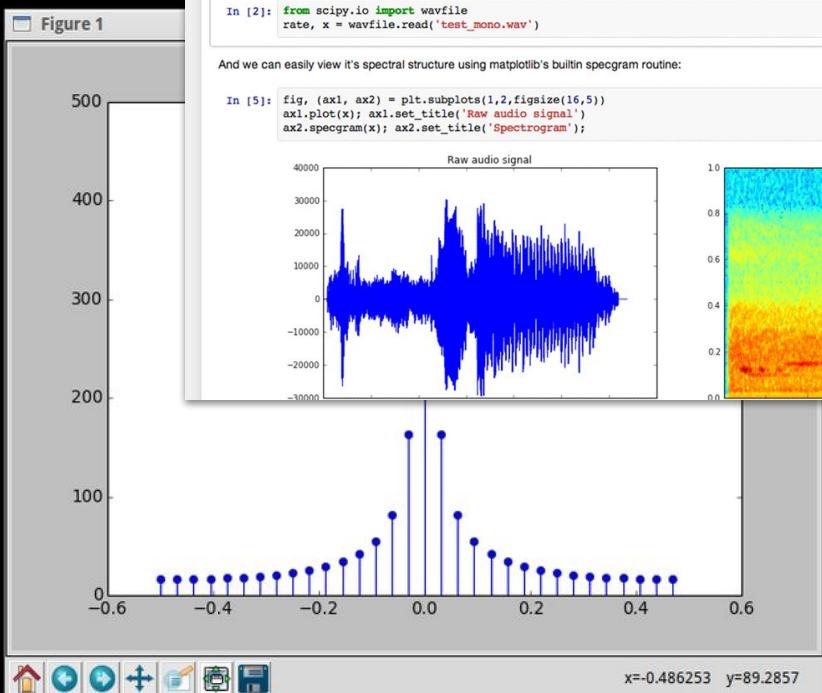
In [2]: a = arange(32)

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]: 
```



## Jupyter spectrogram (autosaved)

File Edit View Insert Cell Kernel Help

CellToolbar

### Simple spectral analysis

An illustration of the [Discrete Fourier Transform](#)

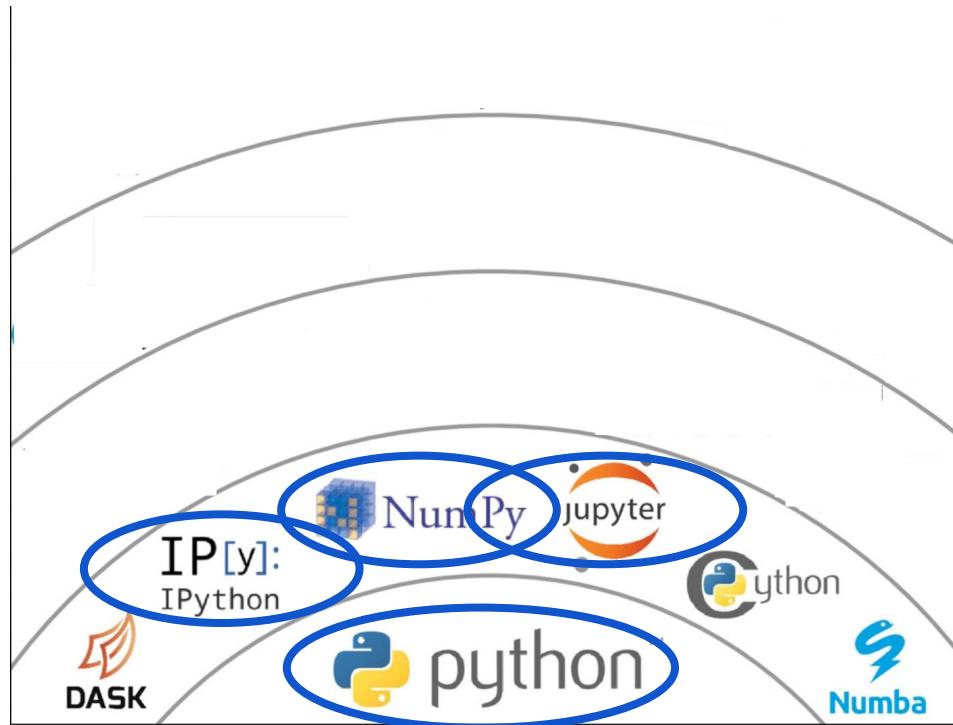
$$X_k = \sum_{n=0}^{N-1} x_n \exp^{-\frac{2\pi i}{N} kn} \quad k = 0, \dots, N-1$$

```
In [2]: from scipy.io import wavfile
rate, x = wavfile.read('test_mono.wav')
```

And we can easily view it's spectral structure using matplotlib's builtin specgram routine:

```
In [5]: fig, (ax1, ax2) = plt.subplots(1,2,figsize=(16,5))
ax1.plot(x); ax1.set_title('Raw audio signal')
ax2.specgram(x); ax2.set_title('Spectrogram');
```

# Our path through scientific python



# NumPy

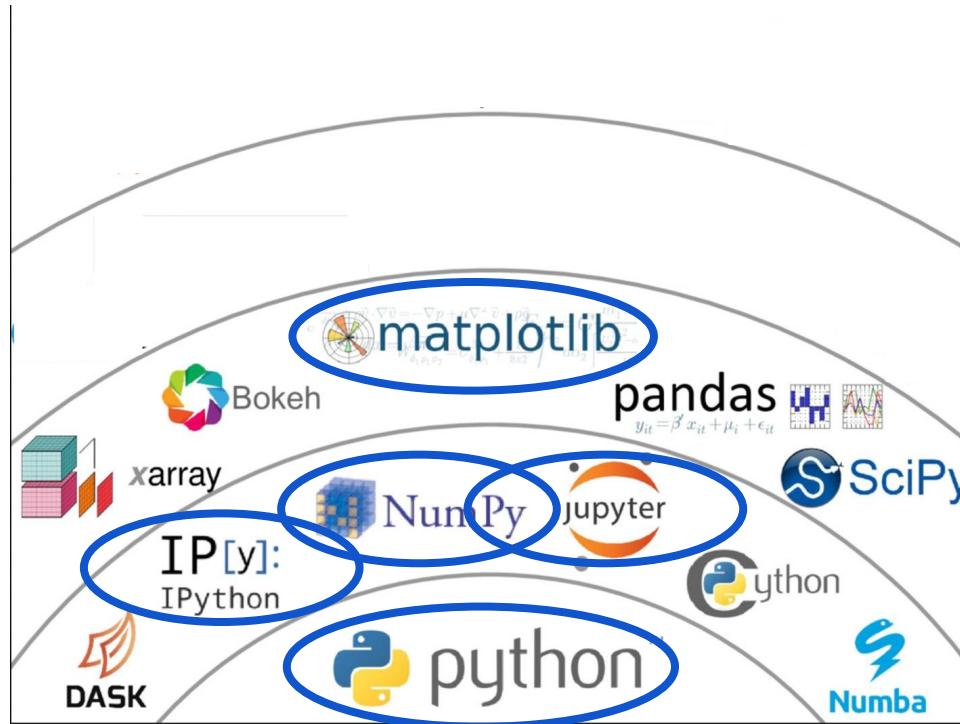
```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]  
array([ 1, 12, 23, 34, 45])
```

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

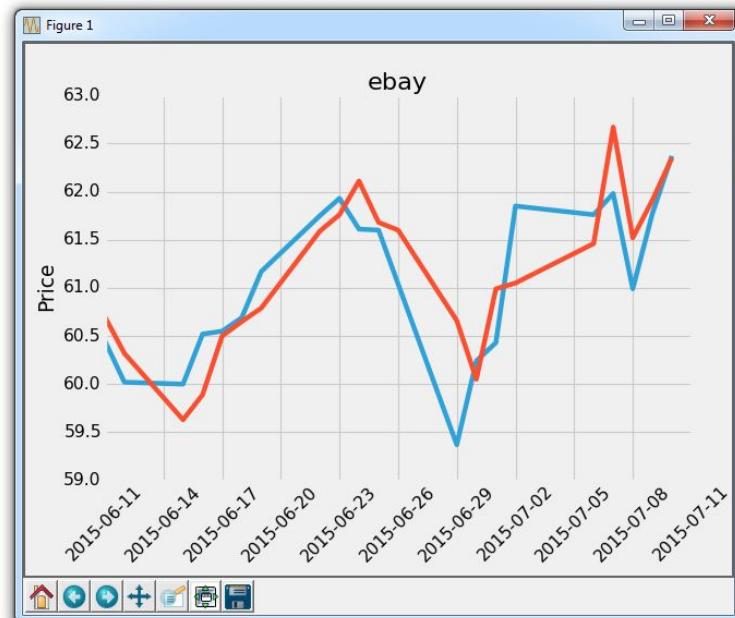
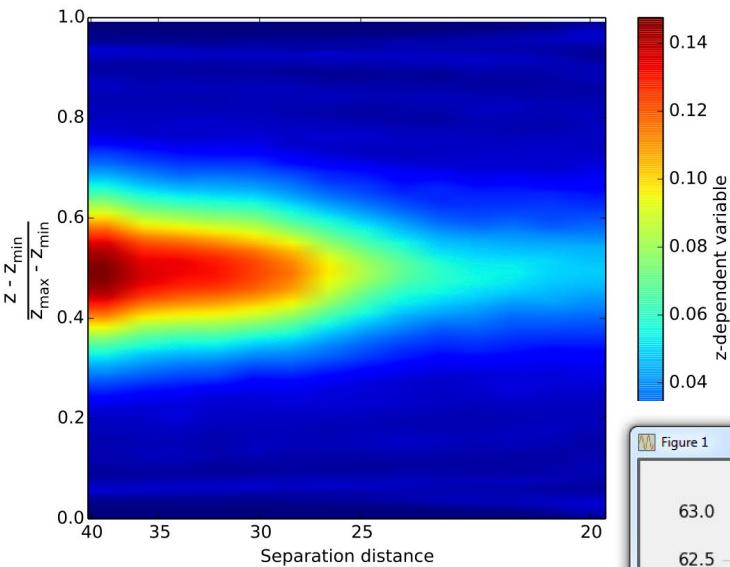
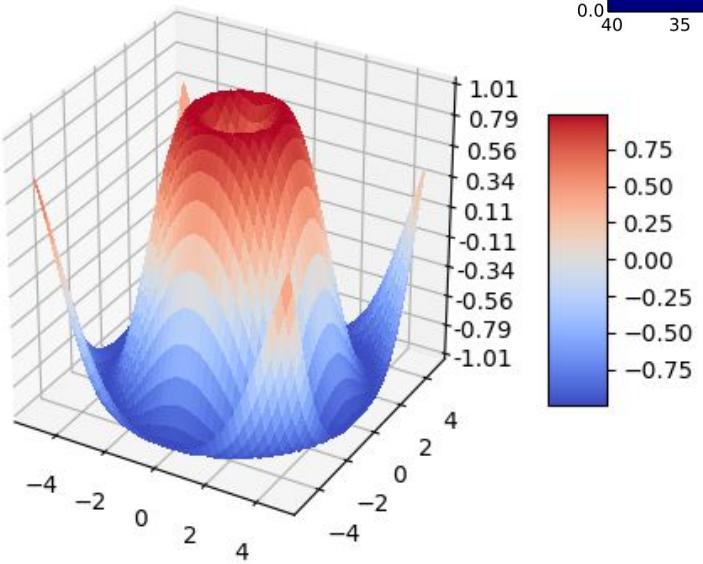
```
>>> mask = array([1,0,1,0,0,1],  
                  dtype=bool)  
>>> a[mask,2]  
array([2,22,52])
```

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 0  | 1  | 2  | 3  | 4  | 5  |
| 10 | 11 | 12 | 13 | 14 | 15 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 30 | 31 | 32 | 33 | 34 | 35 |
| 40 | 41 | 42 | 43 | 44 | 45 |
| 50 | 51 | 52 | 53 | 54 | 55 |

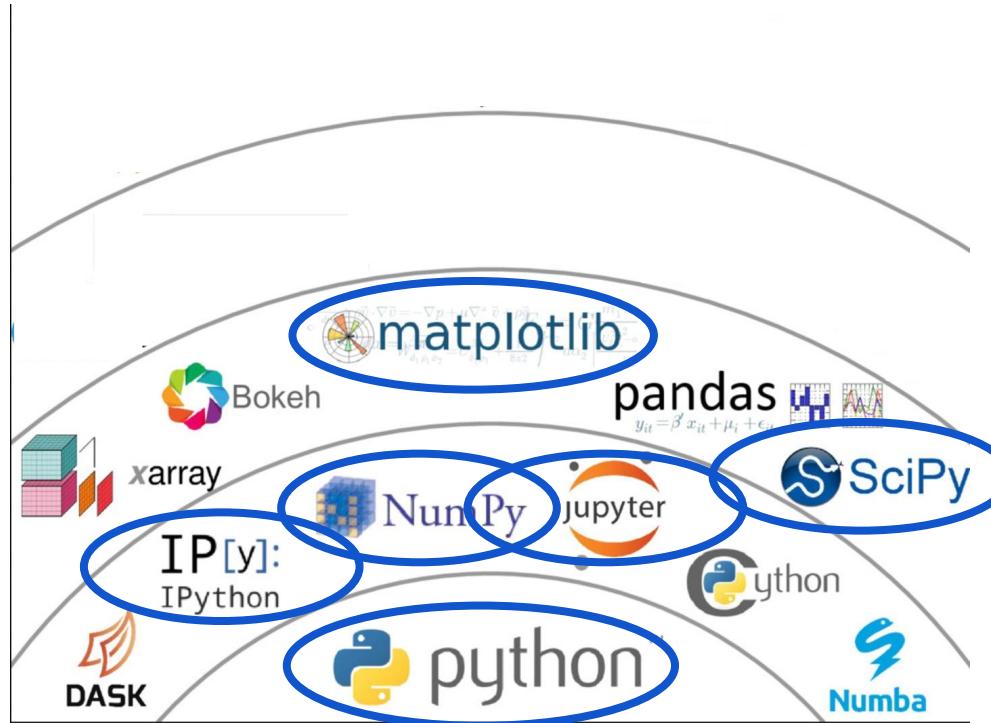
# Our path through scientific python



# Matplotlib



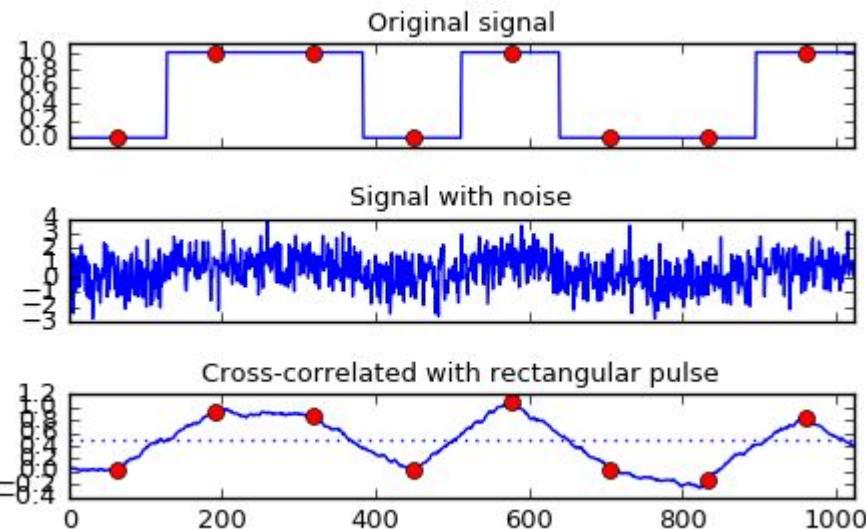
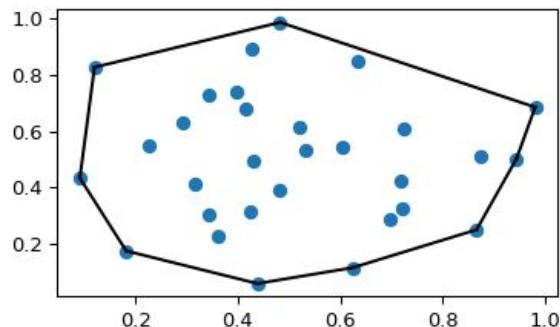
# Our path through scientific python



# SciPy

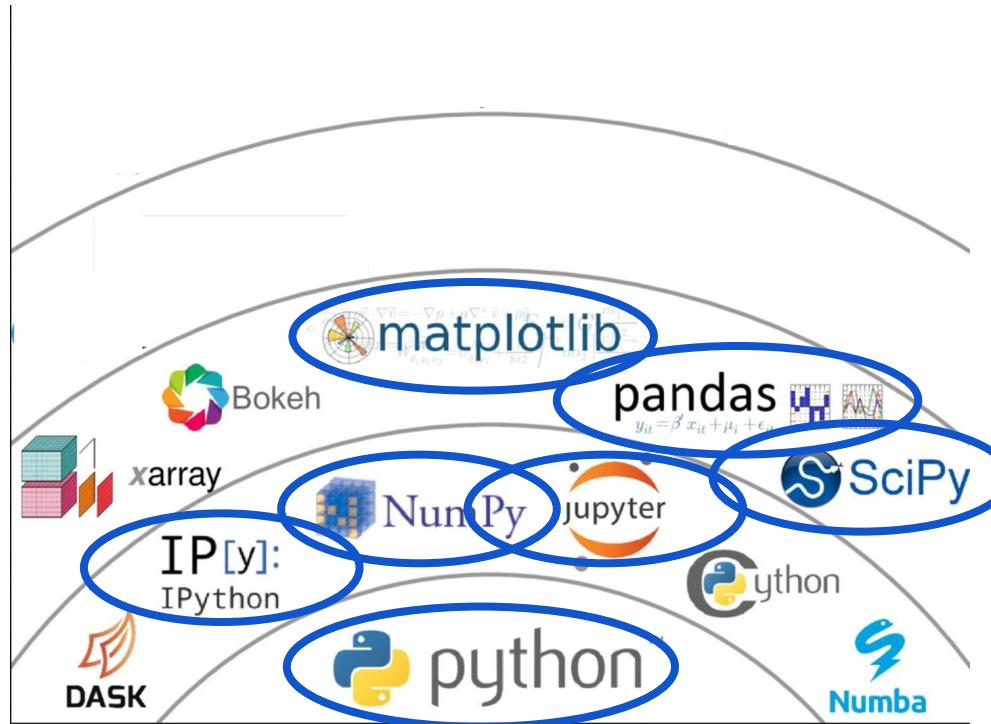
$$\int_0^{\infty} \int_1^{\infty} \frac{e^{-xt}}{t^n} dt dx$$

$$\begin{aligned}x + 3y + 5z &= 10 \\2x + 5y + z &= 8 \\2x + 3y + 8z &= 3\end{aligned}$$



```
>>> a = np.array([ 0.7972,  0.0767,  0.4383,  0.7866,  0.8091,
...                 0.1954,  0.6307,  0.6599,  0.1065,  0.0508])
>>> from scipy import stats
>>> stats.zscore(a)
array([ 1.1273, -1.247 , -0.0552,  1.0923,  1.1664, -0.8559,  0.5786,
       0.6748, -1.1488, -1.3324])
```

# Our path through scientific python



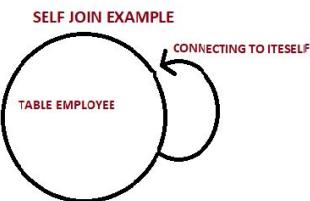
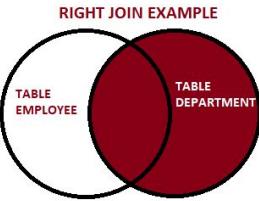
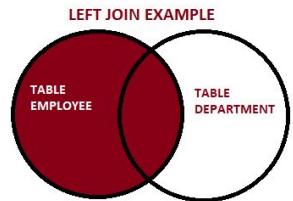
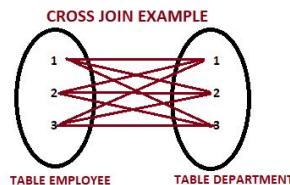
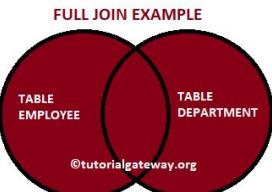
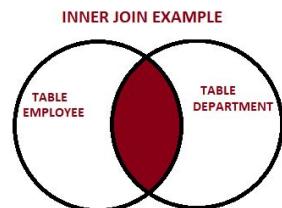
# Pandas

```
# 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

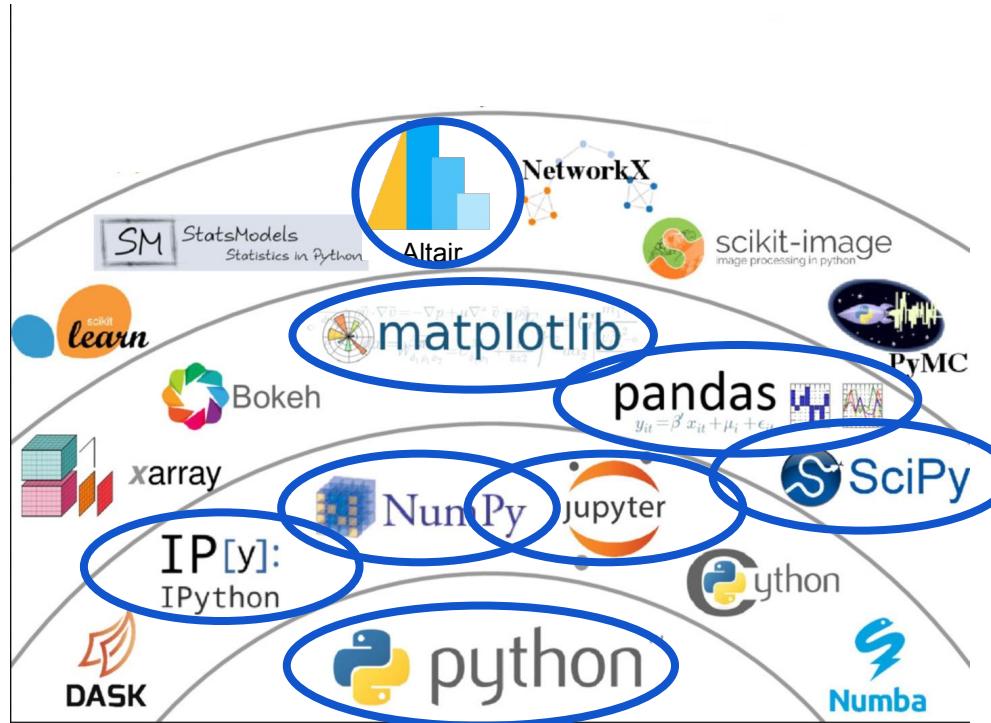
# Create a second dataframe
data2 = [10.0, 10.0, 9, 9, 8, 8, 7, 7, 6, 6]
idx2 = pd.date_range('1/1/2013', periods=10, freq='MS')
df2 = pd.DataFrame(data2, index=idx2, columns=['Revenue'])
df2['State'] = States
```

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

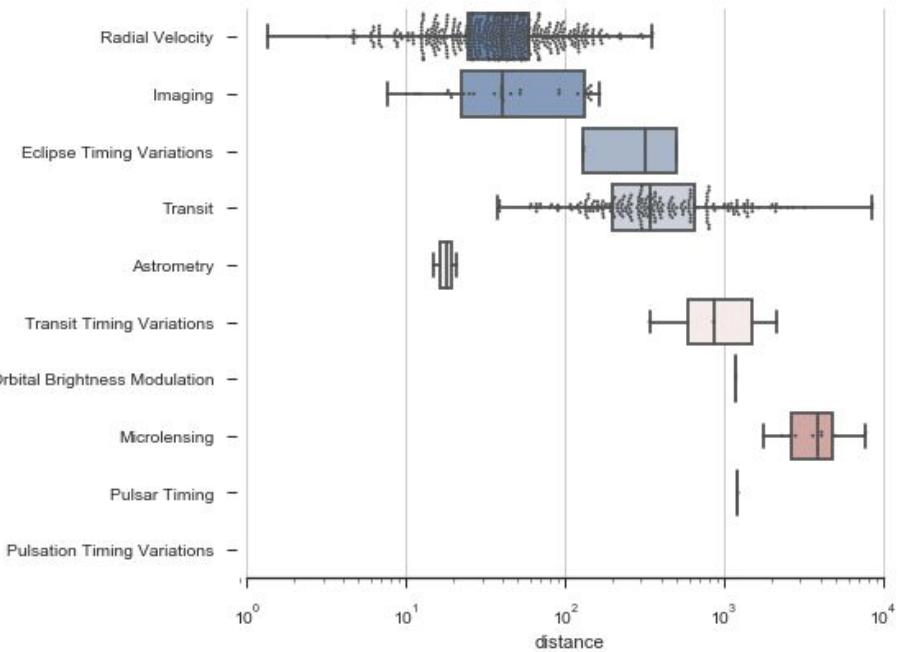
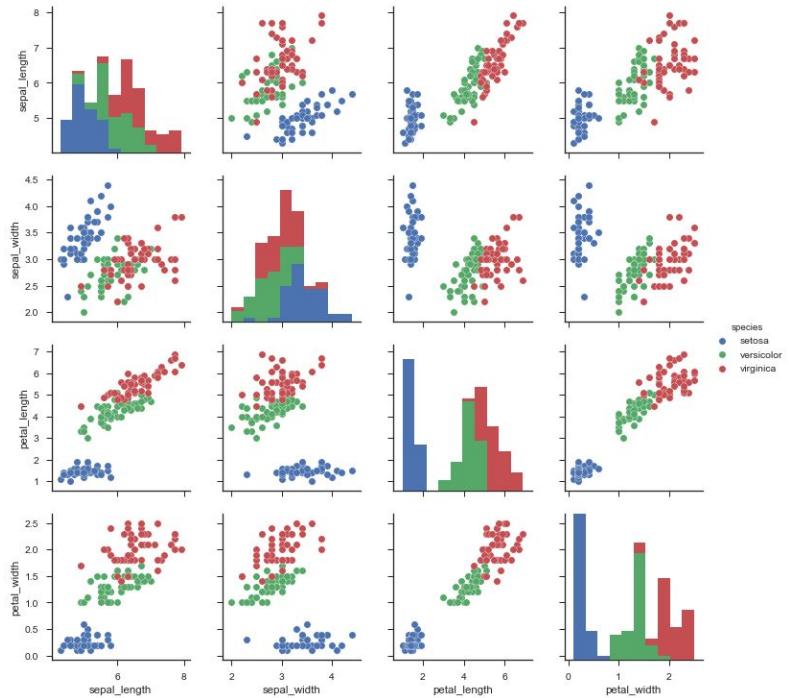
|            | Revenue | State |
|------------|---------|-------|
| 2012-01-01 | 1.0     | NY    |
| 2012-02-01 | 2.0     | NY    |
| 2012-03-01 | 3.0     | NY    |
| 2012-04-01 | 4.0     | NY    |
| 2012-05-01 | 5.0     | FL    |
| 2012-06-01 | 6.0     | FL    |
| 2012-07-01 | 7.0     | GA    |
| 2012-08-01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



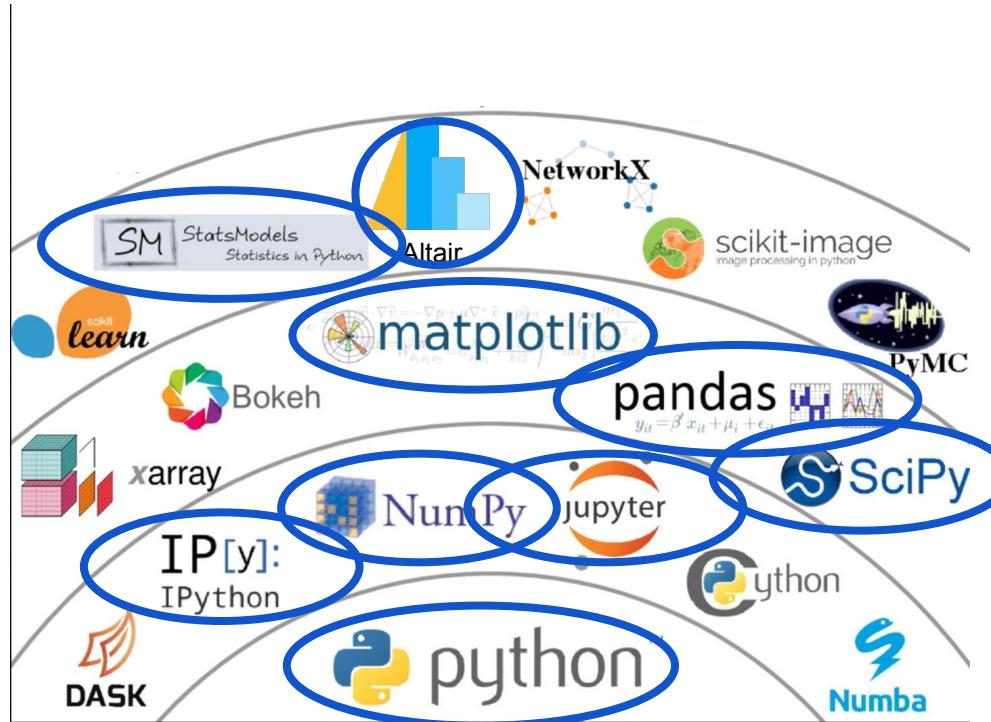
# Our path through scientific python



# Statistical visualization



# Our path through scientific python

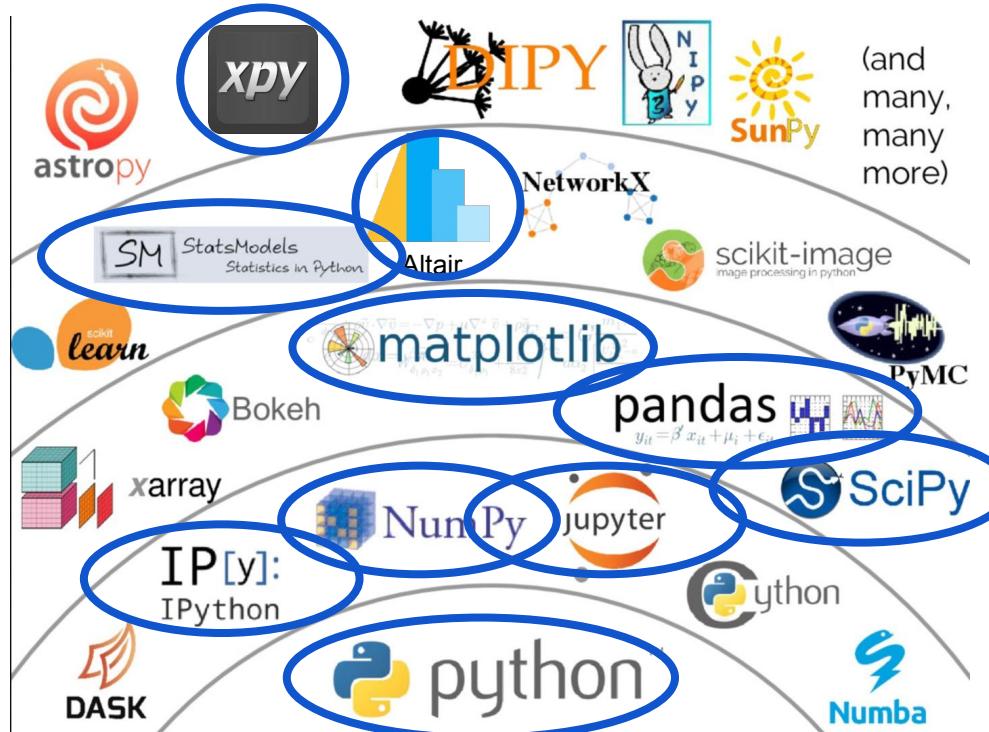


# Statsmodels

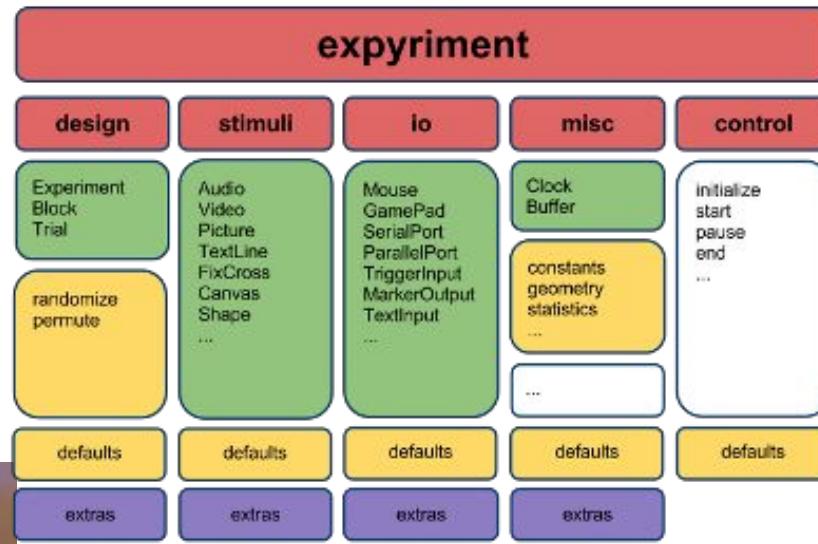
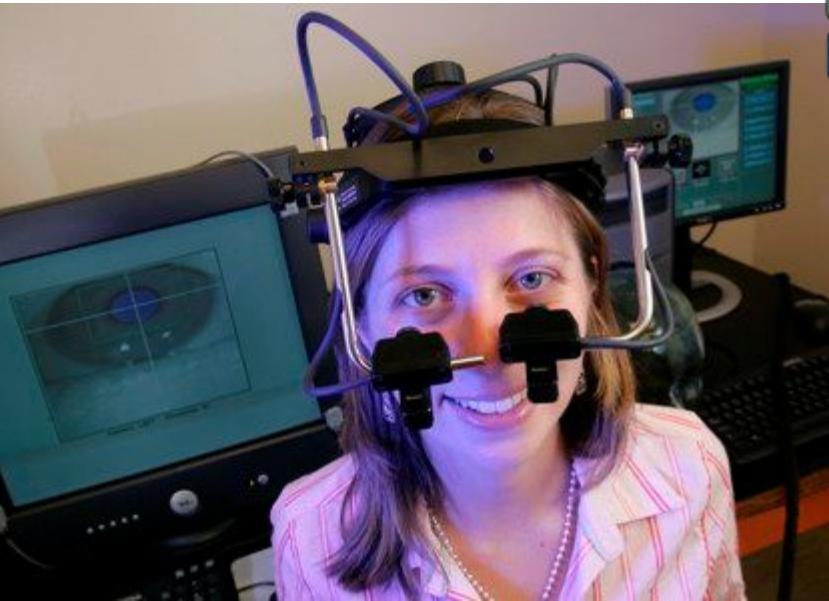
In [5]: results = smf.ols('Lottery ~ Literacy + np.log(Pop1831)', data=dat).fit()

| 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



# Expyriment



Package  
Module  
Class  
Function  
Plugin structure

# Your workflow with Python

## Extracting your data

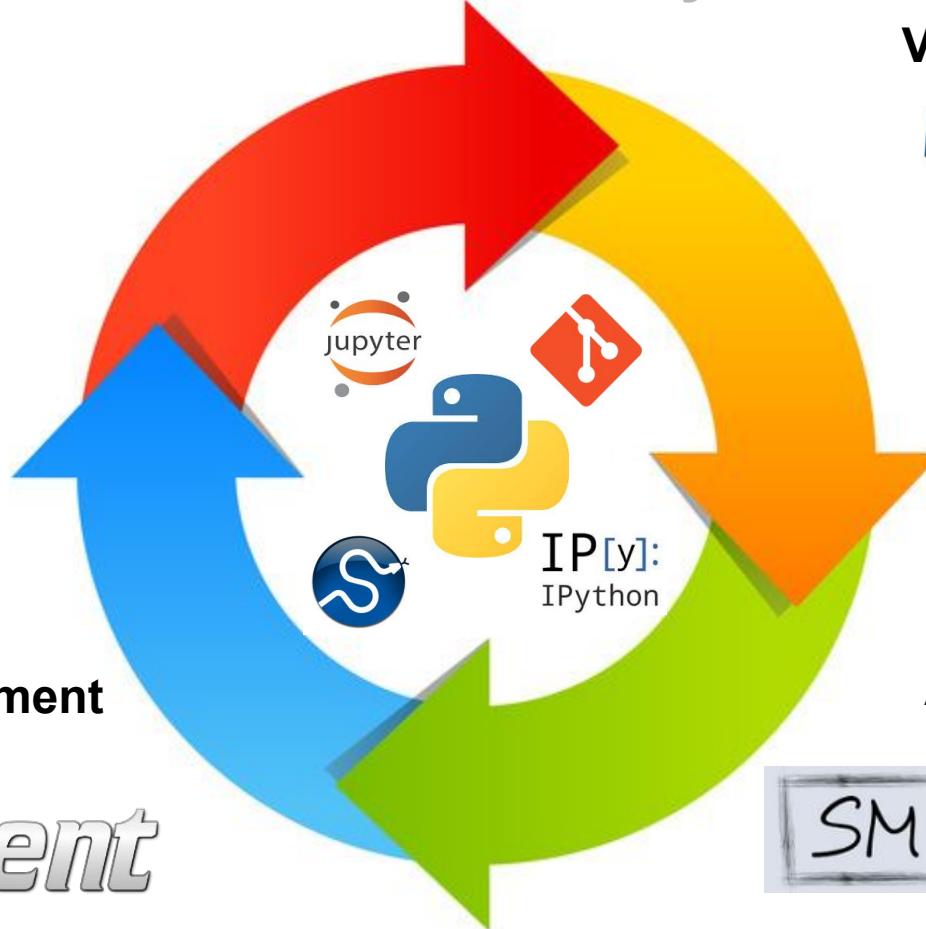


NumPy

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

## Making your experiment

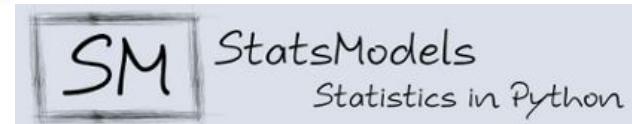


## Visualizing your data

matplotlib



## Analyzing your data



StatsModels  
Statistics in Python



SciPy

# Let's compute!

Go to <https://github.com/scientificprogrammingUOS/lectures> and click “launch binder”

|  |                                     |               |
|--|-------------------------------------|---------------|
| week12-Machine_Learning_and_Parallel_Programming | Clean up                            | 8 months ago  |
| .gitignore                                       | Add hidden checkpoints to gitignore | 11 months ago |
| CORRECTIONS.txt                                  | adde corrections.txt                | 11 months ago |
| README.md  | Fix typo                            | 2 months ago  |
| requirements.txt                                 | Prepare for binder deployment       | 7 months ago  |

README.md

 [launch binder](#)

## Lectures in Scientific Computing in Python

This repository contains all lectures from the course *Scientific programming in Python* that is part of the Cognitive Science program at the University Osnabrück. Each lecture is accompanied by a Jupyter notebook that explains each topic with a combination of code and text. You can view the notebooks directly on GitHub or run them locally and play with the code. If you do not want to install anything, click on the Binder logo above to run all the notebooks in a ready to use environment in the cloud.

# Outline

1. Intro & Organization
2. Basic Python
3. Advanced Python
4. Numerical Computing with NumPy
5. Debugging in Python
6. Visualizations with Matplotlib
7. Framing and Cleaning Data with Pandas
8. Exploring and Analyzing Data with Pandas
9. Statistical and Interactive Visualizations
10. SciPy and Statistical Computing
11. Statistical Modeling with statsmodels
12. Creating Experiments with Expyriment
13. Performance Optimization and Multithreading

## ***Basic Programming in Python:***

- *Hello World*
- *Variables & Assignments*
- *Control Structures*
- *Data Structures*
- *Strings & Formatting*
- *Input & Output*
- *Debugging & Good Practices*
- *Built-In Packages*
- *Object-Oriented Programming*

→ *External Packages*  
→ *Working on Projects*

# Course Infrastructure & Communication

# Communication

- Where to find Information?
  - Stud.IP
  - Lectures-repository
  - Dashboard
- Where to ask questions?
  - Private questions: Email
  - Casual questions: Blubber
  - Questions that can be answered by the community: Stackoverflow Team
  - More detailed questions: Stud.IP Forum

# Stud.IP

- Your first place to go for lecture-recordings

The screenshot shows the Stud.IP interface. At the top, there is a navigation bar with icons for Start, Courses, Files, Messages, Community, Profile, Planner, Search, Tools, and Notice board. Below the navigation bar, a secondary menu bar includes Overview, Forum, Participants, Schedule, Blubber, Courseware (which is highlighted in blue), and Meetings.

The main content area displays a course titled "Seminar: Scientific programming in Python (co-taught by Christoph Stenkamp and Philipp Thölke) - Courseware". On the left, there is a sidebar with a blue header labeled "Courseware" containing links for "Last changes" and "Progress". Below this, under "Content", there is a section titled "Introduction" with sub-links for "Organizational matters" and "Course overview".

The main content area features a large video player window. The video title is "Introduction" and the subtitle is "Scientific Programming in Python". The video player includes standard controls like play/pause, volume, and a progress bar. At the bottom of the video player, there is a button labeled "Video in neuem Fenster anzeigen".

At the top right of the main content area, there is a breadcrumb navigation path: "Introduction > Organizational matters". Below the breadcrumb are several small navigation icons: back, forward, search, and others.

# Stud.IP

- Your first place to go for lecture-recordings

Seminar: Scientific programming in Python (co-taught by Christoph Stenkamp and Philipp Thölke)

The screenshot shows the Stud.IP interface for a course titled "Seminar: Scientific programming in Python". The top navigation bar includes links for Start, Courses, Files, Messages, Community, Profile, Planner, Search, Tools, and Notice board. Below this, a secondary navigation bar shows tabs for Overview, Forum, Participants, Schedule, Blubber, Courseware (which is selected), and Meetings.

**Brief information:** Includes a thumbnail image of a blue circular logo with a yellow 'S' and a red 'P', and a link to "Details".

**Actions:** Includes a "Sign out of the course" button.

**Share:** Includes a "Copy link to this course" button.

**Basic details:** Shows the course title and co-teachers.

**Time / Course location:** Lists weekly sessions on Tuesdays from 12:00-14:00 in room 35/E16, and Thursdays from 14:00-16:00 in room 35/E16.

**Next date:** Lists the next session on Thursday, 23.04.2020, from 14:00-16:00 in room 35/E16.

**Lecturers:** Lists Prof. Dr. phil. Kai-Uwe Kühnberger.

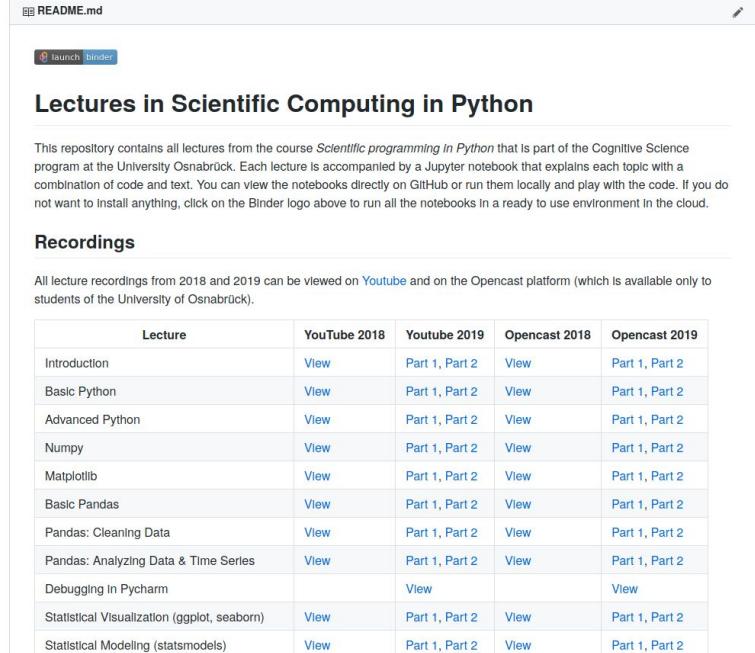
**Announcements:** Lists "First Steps of this course".

**Dates in the period from 21. April 2020 to 05. May 2020:** Lists the dates for the seminar sessions.

**Video in neuem Fenster anzeigen:** A button to view video content in a new window.

# Lectures-repository

- Your first place to go for lecture-notebooks
- Binder explained in a previous video
- See following video for how to install the necessary libraries
- If you want to see content not this year, look at last year's recordings



The screenshot shows the GitHub README.md page for the repository. At the top, there are 'README.md' and 'binder' buttons. Below the title, there is a section titled 'Recordings' with a table showing links to YouTube, YouTube 2018, YouTube 2019, Opencast 2018, and Opencast 2019 for various lectures.

| Lecture                                     | YouTube              | YouTube 2018         | YouTube 2019                   | Opencast 2018        | Opencast 2019                  |
|---|----------------------|----------------------|--------------------------------|----------------------|--------------------------------|
| Introduction                                | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Basic Python                                | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Advanced Python                             | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Numpy                                       | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Matplotlib                                  | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Basic Pandas                                | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Pandas: Cleaning Data                       | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Pandas: Analyzing Data & Time Series        | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Debugging in Pycharm                        |                      | <a href="#">View</a> |                                |                      | <a href="#">View</a>           |
| Statistical Visualization (ggplot, seaborn) | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |
| Statistical Modeling (statsmodels)          | <a href="#">View</a> |                      | <a href="#">Part 1, Part 2</a> | <a href="#">View</a> | <a href="#">Part 1, Part 2</a> |

# Dashboard

- Your first place to go for the homework
- Work-in-Progress, far from perfect
- Login via GitHub - only possible after accepting the first homework
  - See following video for how to do that
- Your go-to place to see how many points You got so far
- Gets live information from GitHub and Stud.IP
  - Homework information
  - Announcements
  - Git-Commits
  - Forum-Posts

## Current Homework

### homework04

Deadline: 2020-04-28 12:00 (6 days 21:20 to go)

Points: 0/10 ([check why](#))

There are 0 commits to the homework-branch since you accepted the homework.

There are 0 posts regarding this homework in the [Stud.IP Forum](#).

Stackoverflow tags: ["homework01"]

Additional information:

None

Write an [Email](#) to the admins regarding this homework

## Past Homework

### homework03

Deadline: 2020-04-20 12:00

Points: 3/3 ([check why](#))

# Dashboard

- Your first place to go for the homework
- Work-in-Progress, far from perfect
  
- Links to all other necessary places
- Link to the JupyterHub-instance

### Announcements:

#### First Steps of this course

Dear Pythonists,

Welcome to this course! We spend the last days eagerly preparing everything, and so far everything is looking really well and we are happy to spend this semester with you, most likely online for the most part. As a few of you have already asked a few things about how the course will proceed, here some quick information:

- The Tuesday-session will not be held live. Instead, we will use Courseware to provide you with Videos with varying degrees of interactivity.
- The first session will be split into multiple (non-interactive) Videos that we will upload within the next three hours on Stud.IP Courseware/OpenCast
- Any other Information, also regarding how we proceed with the practice on Thursday, will be explained in the video, so please make sure to watch all of these Videos before you ask any further questions. I will update this announcement as soon as all Videos are online
- In the videos, we will explain the course infrastructure. Most of that is home-coded and completely new and largely untested, so please be patient if it doesn't work terribly smoothly at first
- If, after having watched all the videos, you have questions left that are relevant for everybody or that can also be answered by other participants of the course, please ask them in the Stud.IP Forum or on Blubber
- The course is held and organized by Philipp Thölke and Christoph Stenkamp. Besides his name on your Scheine, Prof. Kühnberger has nothing to do with this course. That means as long as it's not about that, don't ask him any questions, but ask us instead (with an email if it's supposed to be private, and via the mentioned means otherwise).

That's all for now, everything else will be explained in a video!

Here's to a good semester!

[Open Jupyter at hub.sciprog.de](#)

[Open lectures-repository](#)

[Open Stackoverflow Team](#)

[Open Stud.IP: Blubber | CourseWare | Meetings](#)

[Stud.IP Forums: | Allgemeine Diskussion | Installation Problems | Homework 1 | Homework 2 | Homework 3 | Technical Problems with the course Infrastructure](#)

[Open lectures-repository](#)

# Stud.IP Forum

Start Courses Files Messages Community Profile Planner Search Tools Notice board

STUD.IP

Overview Forum Participants Schedule Blubber Courseware Meetings

## Seminar: Scientific programming in Python (co-taught by Christoph Stenkamp and Philipp Thölke) - Forum [?](#)



### Übersicht

#### Allgemein

| Name of the area:  | Postings | last answer   |
|--|----------|---|
| Allgemeine Diskussion<br>Hier ist Raum für allgemeine Diskussionen | 2        | from Christoph Stenkamp<br>on 14.04.2020, 21:20 <a href="#">?</a> |
| Installation Problems  | 0        |   |
| Homework 1   | 0        |   |
| Homework 2   | 0        |   |
| Homework 3   | 0        |   |
| Technical Problems with the course infrastructure                  | 0        |   |

**Overview**

- New postings
- Latest posting
- Bookmarked postings

**Search**

Search posts

Title  Content  Author

**Actions**

[Subscribe to the entire forum.](#)

**Export**

[Export postings as PDF](#)

# Stackoverflow Team

Products Scientific Programming 2020 Search... 1 2 ? 🔍

## Scientific Programming 2020 Questions

Ask Private Question

3 questions Newest Active Bountied Unanswered More Filter

1 vote 1 answer 10 views

Do we have standards for commit messages? Do we have standards for commit messages? modified Apr 15 at 16:04 Chris Stenkamp 36 3

version-control

1 vote 1 answer 3 views

What code repository do we use and how do I request access? 12345 answered Apr 15 at 14:04 Chris Stenkamp 36 3

version-control

1 vote 1 answer 6 views

How do I setup my local environment? How do I setup my local environment? answered Apr 15 at 14:03 Chris Stenkamp 36 3

architecture

Custom Filters Create a custom filter

Watched Tags on Scientific Programming 2020

Watch tags to curate your list of questions and receive optional notifications on that tag's activity.

Watch a tag

Ignored Tags Add an ignored tag

Integrations Slack Connect

Scientific Programming 2020 Private Team Basic

Questions Tags Users

Stack Overflow

TEAMS +

Scientific Program...

# Stackoverflow Team

To access the **Scientific Programming 2020** Team, you need to have a Stack Overflow account.

I have a Stack Overflow account

I don't have a Stack Overflow account

**Welcome!**

Stack Overflow for Teams makes it easy to search and find answers in a private and secure environment.

**S** **Scientific Programming 2020** is a private knowledge base powered by Stack Overflow for Teams

**Let's set up your account**

Enter your **@uos.de** email to get started

youremail@uos.de

**Continue**

Have a question? Contact your admin

# Setup for the course

# Which IDE?

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

- There are as many IDEs as there are opinions about them
- Pycharm, vscode, atom, notepad++, vi, emacs...
- For this course we will only use **JupyterLab**
- Contains all you need and is great for interactive development as usually encountered in scientific contexts

# Which IDEs are there?

- Atom <https://atom.io/> (or for linux *apt-get install atom*)
  - Useful Packages: **Hydrogen** by *nteract* & **hydrogen-launcher** by *lgeiger*, providing an interactive Kernel
  - 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 <sup>supposedly</sup> much faster once you did
- Jupyter Lab
  - Can not only work with pure code-files, but also *Notebook-Files* that contain code, formatted text and results of running your code
    - These notebooks are nicely rendered on Github and can be exported to HTML or PDF, or simply to a .py-script
  - Can also edit standard-python-files and use them with an interactive Kernel

# Working without installing anything

# http://hub.sciprogram.de

hub.sciprogram.de/user/cstenkamp/lab?redirects=1

File Edit View Run Kernel Hub Tabs Settings Help

Name Last Modified

- lost+found 13 hours ago
- Untitled Folder 2 hours ago
- Untitled Folder 1 2 hours ago
- Untitled.ipynb 12 hours ago
- asdf.py 2 hours ago
- untitled.txt a minute ago

Untitled.ipynb    untitled.txt    asdf.py

a = 1+1

Create Console for Editor

untitled.txt

```
Python 3.7.3 | packaged by conda-forge | (default, Mar 27 2019, 23:01:00)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.6.1 -- An enhanced Interactive Python. Type '?' for help.
```

```
[1]: a = 1+1
[2]: a
[2]: 2
```

# <http://hub.sciprog.de>

- The hub is a nice way to work on the tasks without having to install python on your system, all you need is a web browser
- Only use it if you need it, and close the sessions after you're, because the more users are online the more we pay
- It may also be a bit buggy at first, so please bear with us!

# Working with installing everything

# Virtual environments

- Virtual environments are sandboxes for your python and its packages - allowing 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 sheets:
  - <http://know.continuum.io/rs/387-XNW-688/images/conda-cheatsheet.pdf>
  - [https://conda.io/projects/conda/en/latest/\\_downloads/1f5ecf5a87b1c1a8aaf5a7ab8a7a0ff7/conda-cheatsheet.pdf](https://conda.io/projects/conda/en/latest/_downloads/1f5ecf5a87b1c1a8aaf5a7ab8a7a0ff7/conda-cheatsheet.pdf)

# Install Anaconda or Miniconda

- **Anaconda** is a *Python distribution* made for scientific computing, packed with its own package manager (*conda*).
  - Anaconda contains >720 pre-installed packages at ~3GB
  - Download: <https://www.anaconda.com/distribution/>
- **Miniconda** is the same as Anaconda, just without all the pre-installed packages (besides the package manager)
  - Thus its only 66MB, and every package you need can be installed via *conda*
  - Download: <https://docs.conda.io/en/latest/miniconda.html>

We strongly recommend you to install Miniconda, as Anaconda simply installs Miniconda plus >160 packages into your base-environment (most of which you don't need, and certainly not in your base-environment)

# Installation instructions: Miniconda & Linux

- Download your version from <https://docs.conda.io/en/latest/miniconda.html>
- `bash Miniconda3-latest-Linux-x86_64.sh` (saying yes when it asks you to add it to the terminal)
  - Make sure to `conda update --all` afterwards.
  - Once you have installed it and opened a new shell, conda will automatically enable the base-environment
    - It will indicate that with a `(base)` in front of each line.
    - Further, which python should now answer .../anaconda3/bin/python or .../miniconda3/bin/python
    - If you don't want conda to automatically activate the conda-base-environment, you can turn it off using`conda config --set auto_activate_base false`
- If you don't have git already (try by running `which git`), install it using `sudo apt-get install git`

To have jupyterlab globally on your system:

- `conda install jupyter`
- `conda install jupyterlab`
- `conda install nodejs=10.13`
- `jupyter lab build`

To create the environment for the class:

- `git clone https://github.com/scientificprogrammingUOS/lectures.git`
- `conda env create -f lectures/environment.yml`
- `conda activate scientific_programming`
- `jupyter lab build`

# Installation instructions: Miniconda & Windows

- Download your version from <https://docs.conda.io/en/latest/miniconda.html>
- Run the graphical installer.
  - **Make sure to add conda to your PATH<sup>1</sup>**, such that you can use it from your standard terminal.
- Afterwards, open the command-prompt as Administrator (hit Win-Key, type “cmd”, right-click “Command Prompt”, select “as Admin”)
  - Test if your installation was correct by running `where python`  
→ it should return a path containing .../anaconda3/... or ../miniconda3/...
  - Update your Conda installation by running `conda update --all`
  - Note that if you want to use the conda-base-environment without side effects, you have to run `conda activate` every time you open a shell
- Install git from <https://git-scm.com/downloads>
  - Make sure that git **can** be used from your command-line<sup>2</sup>
  - Make sure that you use one of the two options **committing unix-style<sup>3</sup>**
  - Leave everything else the way it is
- Afterwards you should have the commands `python`, `conda` and `git` as registered commands<sup>4</sup>

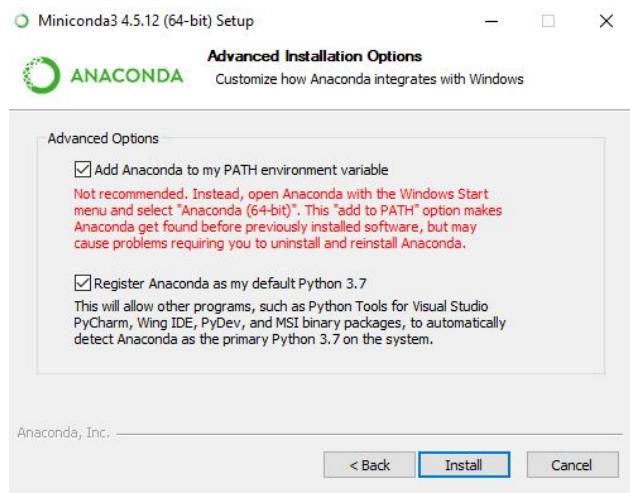
To have jupyterlab globally on your system:

- `conda install jupyter`
- `conda install jupyterlab`
- `conda install nodejs=10.13`
- `jupyter lab build5`

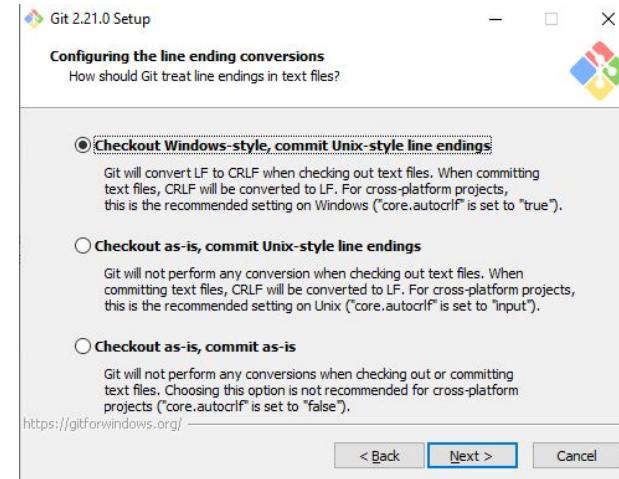
To create the environment for the class:

- `git clone https://github.com/scientificprogrammingUOS/lectures.git`
- `conda env create -f lectures/environment.yml`
- `conda activate scientific_programming`
- `jupyter lab build5`

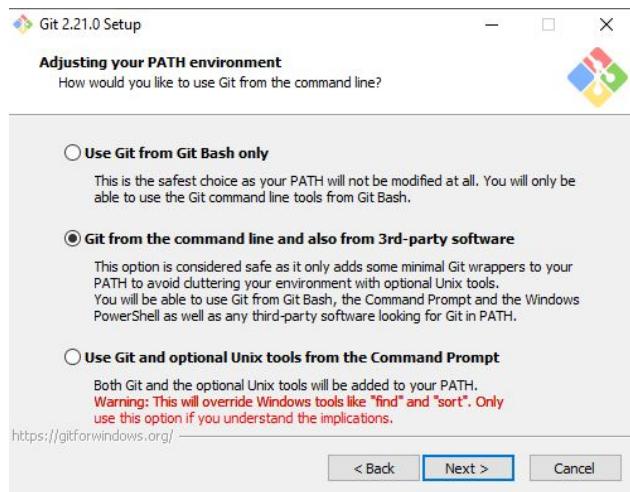
1)



3)



2)



4)

```

C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>where python
C:\Users\User\Miniconda3\python.exe

C:\Users\User>where conda
C:\Users\User\Miniconda3\Library\bin\conda.bat
C:\Users\User\Miniconda3\Scripts\conda.exe

C:\Users\User>where git
C:\Program Files\Git\cmd\git.exe

C:\Users\User>

```

# Windows: If the previous didn't work

- If you are sure you followed the previous steps as described, and you still get an error when trying to set up the environment, you can instead not install the complete environment via conda, but instead via pip:

```
conda create -n scientific_programming python=3.7
conda activate scientific_programming
pip install jupyter nodejs jupyterlab ipywidgets numpy matplotlib pandas statsmodels
seaborn plotnine pivottablejs pandas-profiling altair vega_datasets numba pytest cython
tqdm markdown expyriment pygame jupyterlab-solutions jupyterlab_code_formatter nb_black
ipympy scikit-misc pySerial tqdm cleanipynbconda install nodejs -y
conda install mpl_sample_data
```

- Afterwards, you can enable this environment just as if you installed it using the environment.yml file.
- As conda is however better for installing packages (due to better storage use and dependency resolution), this should only be an emergency solution.

# After Installation

- Once you activate your environment using `conda activate scientific_python`, your shell should indicate that you're inside this environment
- Note that you have to **activate your this environment every time you work on the exercises!**
- To test if all packages are installed successfully, run `conda list` and check if all demanded packages are indeed listed.
- To start working inside jupyter lab, navigate to the correct directory using `cd`, and then start jupyterlab by executing `jupyter lab .` (yes, including the dot)<sup>5</sup>

5) If the commands involving jupyter don't work on Windows, try using a hyphen instead of a space:

→ `jupyter-labextension install @lckr/jupyterlab_variableinspector` (*exemplary, do this for all the labextensions*)  
→ `jupyter-lab .`

# Get the lectures-repository to run

To nicely render some stuff in our repository (see image), you need additional libraries

The screenshot shows a Jupyter Notebook cell with the following content:

```
Plot the distribution of males and females in a barplot. Rotate the xticks of the resulting plot by 25° like in the MPL-Homework
```

```
[*]: count_down(3)
```

A progress bar indicates the cell is 5% complete, having run 9/180 cells in 00:09 (02:51 total, 1.00s/it).

At the bottom, there are two buttons: "Solution available" and "REVEAL SOLUTION".

To install, open a terminal/cmd, `cd` to the lectures-repo, activate the environment and run the following:

On Windows:

- `jupyter lab build5`
- *if needed:* `jupyter labextension uninstall jupyterlab-jupytext && jupyter labextension install jupyterlab-jupytext`
- `postbuild_windows.bat`

On Linux/Mac:

- `jupyter lab build`
- *if needed:* `jupyter labextension uninstall jupyterlab-jupytext && jupyter labextension install jupyterlab-jupytext`
- `chmod +x postbuild_linux`
- `./postbuild_linux`

# Installation instructions: Miniconda & Linux

You can also try to just go for one single command on Linux, replacing all the previous steps:

```
cd ~ && wget  
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh && chmod +x  
Miniconda3-latest-Linux-x86_64.sh && ./Miniconda3-latest-Linux-x86_64.sh -b -p  
$PWD/conda && eval "$($PWD/conda/bin/conda shell.bash hook)" && echo $0 | awk  
-F"/" '{print $NF}' | read myshell && conda init myshell && cd ~ && git clone  
https://github.com/scientificprogrammingUOS/lectures.git && conda env create -f  
lectures/environment.yml && conda activate scientific_programming && jupyter lab  
build && chmod +x lectures/postbuild_linux && lectures/postbuild_linux
```

# Howto: Jupyterlab

# Getting used to Jupyter lab

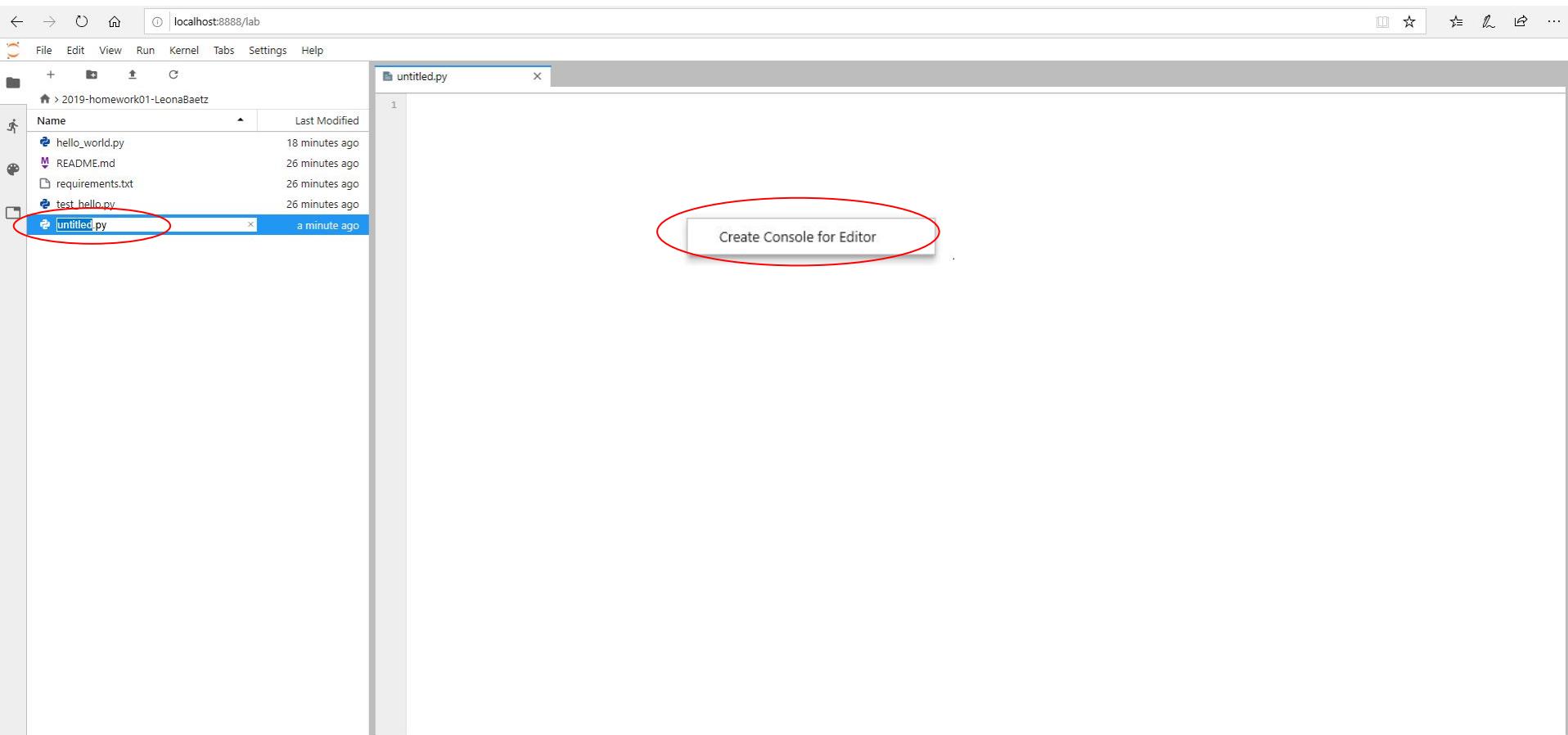
- Jupyter Lab Cheat-sheet:  
[https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf\\_bw/](https://www.cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf_bw/)
- Markdown-Commands Cheat-Sheet:  
<https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>

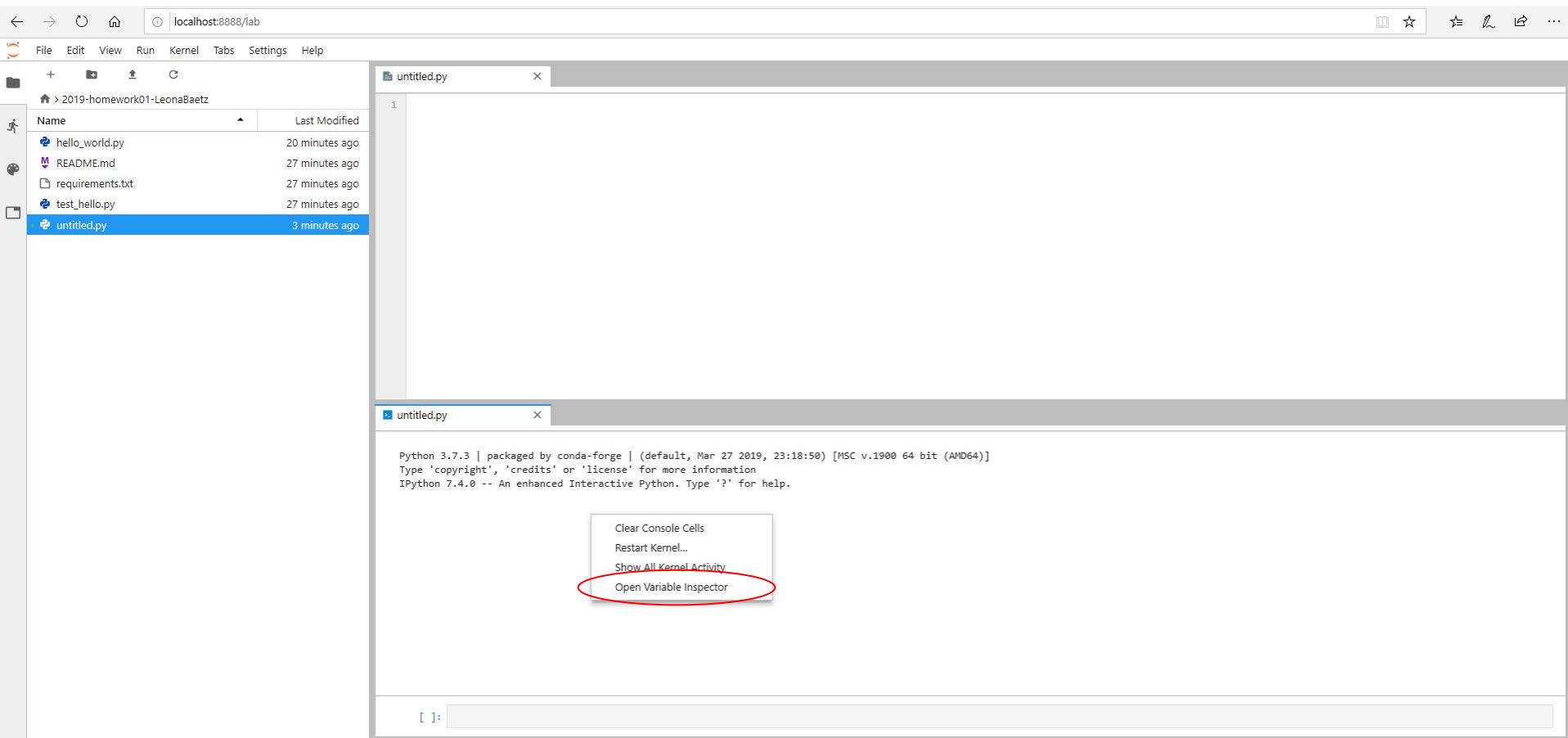
# Howto: Interactive Kernel

A screenshot of a Jupyter Notebook interface. On the left, there is a file browser pane titled "Launcher". It shows a list of files in the current directory:

| Name             | Last Modified |
|------------------|---------------|
| hello_world.py   | seconds ago   |
| README.md        | 8 minutes ago |
| requirements.txt | 8 minutes ago |
| test_hello.py    | 8 minutes ago |

The file "hello\_world.py" is currently selected and highlighted in blue. To the right of the file browser is the main workspace area, which contains a "Launcher" sidebar. The "Launcher" sidebar includes sections for "Notebook", "Console", and "Other". The "Notebook" section shows a "Python 3" icon. The "Console" section also shows a "Python 3" icon. The "Other" section contains two items: "Terminal" (represented by a dollar sign icon) and "Text File" (represented by a document icon). A red circle highlights the "Text File" icon.





The screenshot shows a Jupyter Notebook interface running on localhost:8888/lab. The top navigation bar includes back, forward, search, and other standard browser controls. The main menu bar has File, Edit, View, Run, Kernel, Tabs, Settings, and Help.

The left sidebar displays a file tree for the directory 2019-homework01-LeonaBaetz. The files listed are:

- hello\_world.py (modified 22 minutes ago)
- README.md (modified 29 minutes ago)
- requirements.txt (modified 29 minutes ago)
- test\_hello.py (modified 29 minutes ago)
- untitled.py (modified seconds ago)

The central workspace contains two code cells:

- A top cell titled "untitled.py" with the code: `a = 1+1`.
- A bottom cell titled "untitled.py" showing the Python environment details:

```
Python 3.7.3 | packaged by conda-forge | (default, Mar 27 2019, 23:18:50) [MSC v.1900 64 bit (AMD64)]  
Type 'copyright', 'credits' or 'license' for more information  
IPython 7.4.0 -- An enhanced Interactive Python. Type '?' for help.
```

To the right of the workspace is a "Variable Inspector" panel titled "Inspecting python-kernel 'python3'". It shows a table with one row:

| NAME | TYPE | SIZE | SHAPE | CONTENT |
|------|------|------|-------|---------|
| a    | int  | 28   |       | 2       |

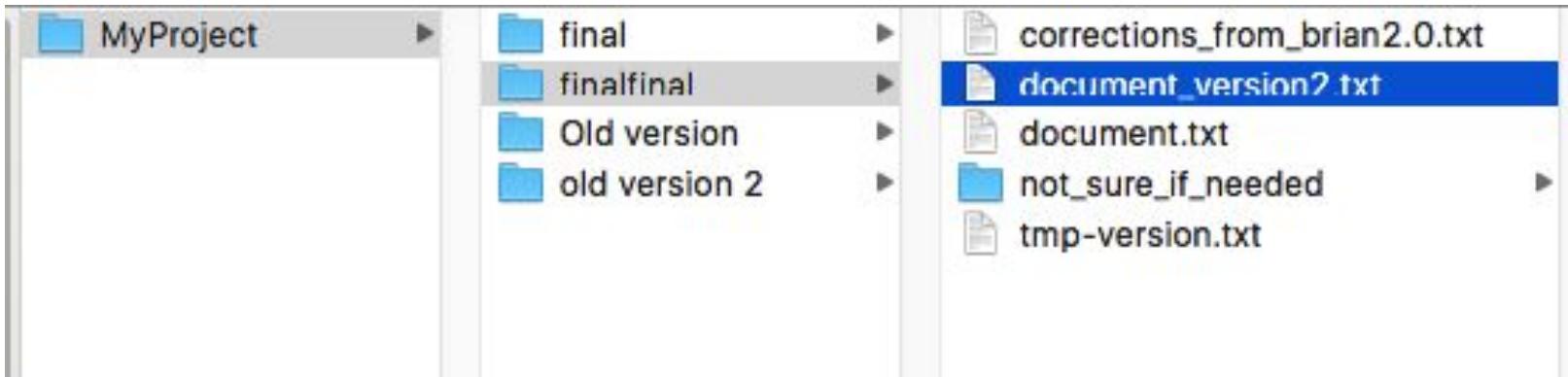
# Intro: git



# 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, compare new versions to old versions, have multiple versions in parallel, and work simultaneously on projects
- 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...

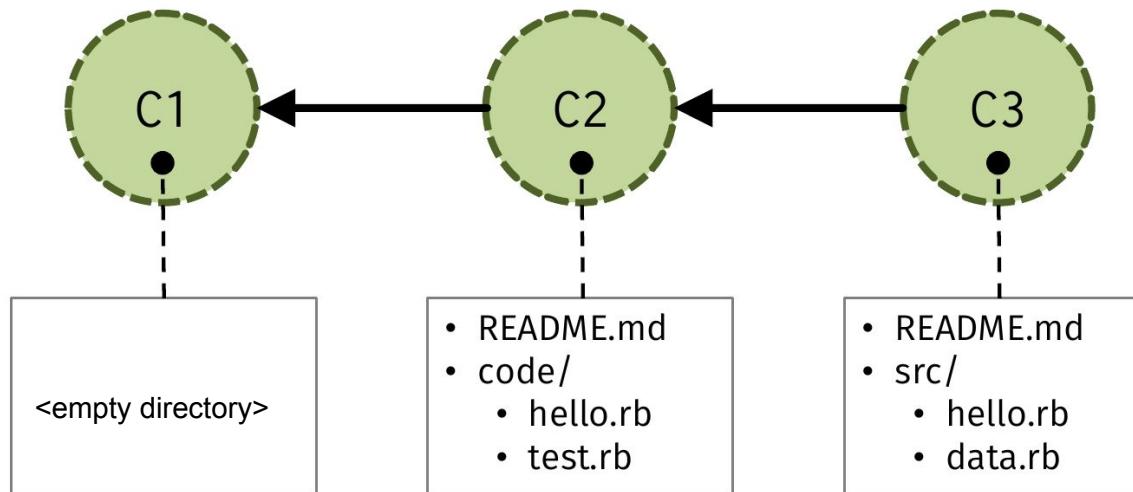
- Author
- Timestamp
- Description

Allows...

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

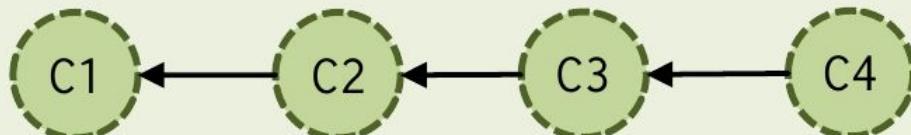
# A commit

- 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

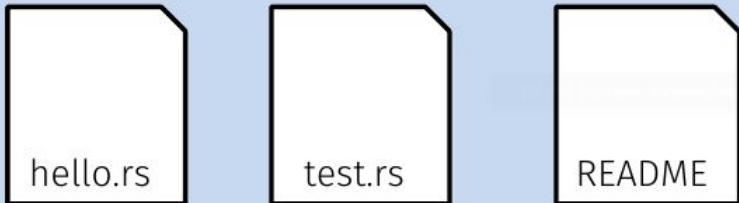
Version Database:



Puts all **staged** files into a new commit

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

Working Directory:



git commit



# 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

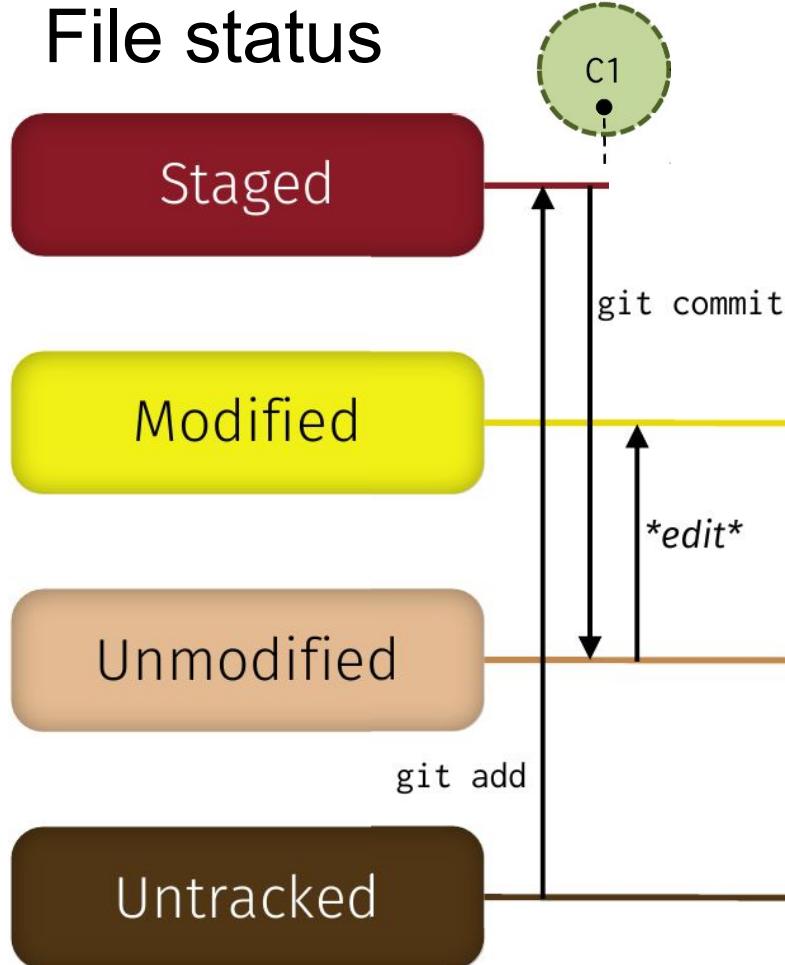
Staged

Modified

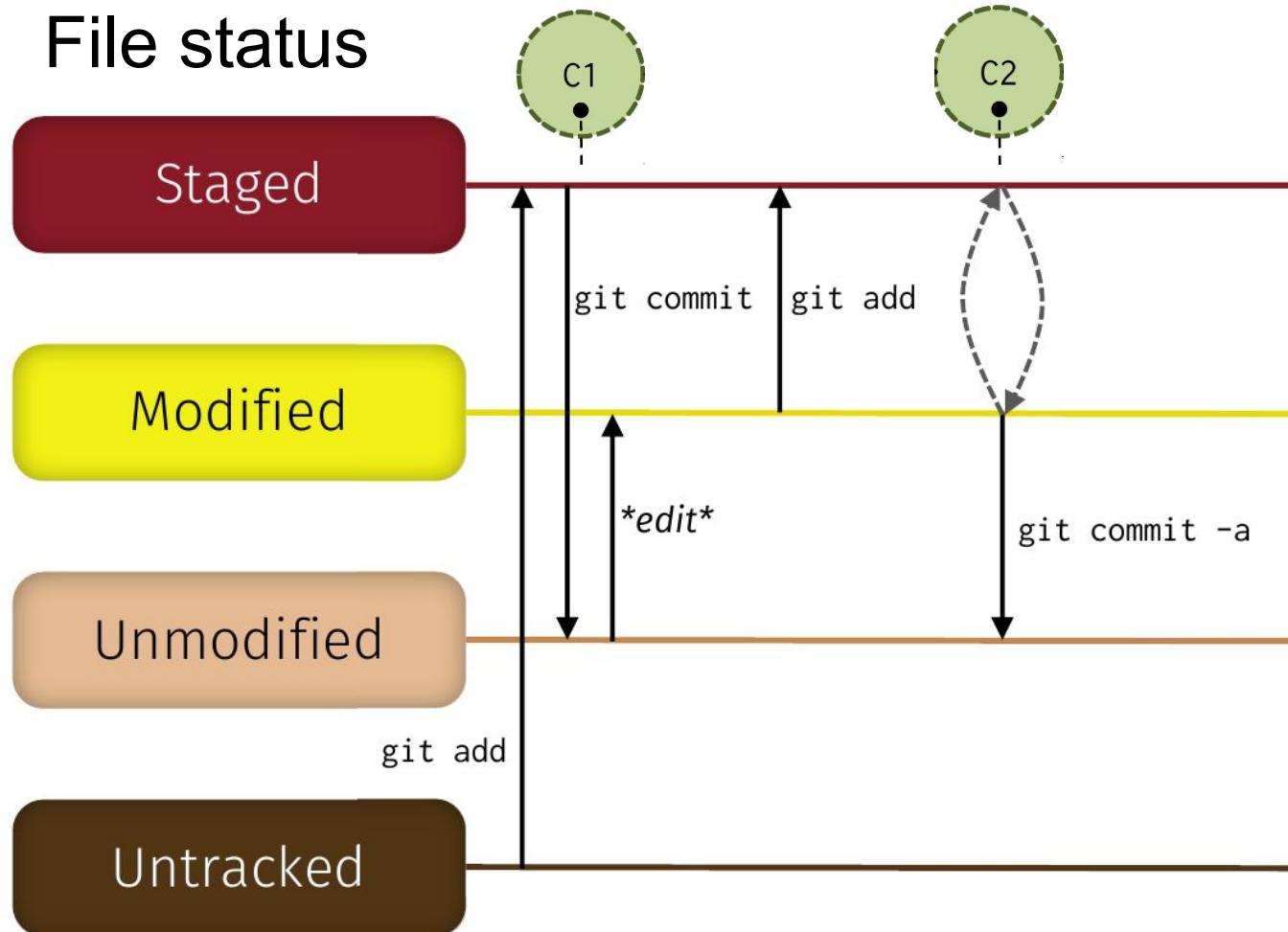
Unmodified

Untracked

# File status



# File status



# git status

Staged

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

Modified

```
Changes to be committed:  
(use "git reset HEAD <file>..." to unstage)  
  
  new file: bye.rs  
  new file: hello.rs
```

Unmodified

```
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
```

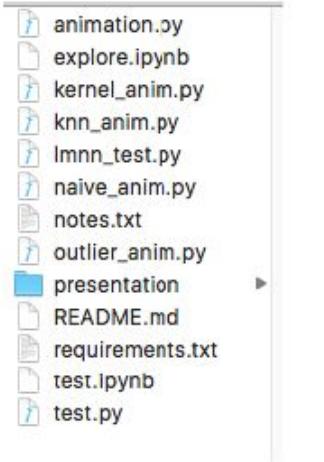
Untracked

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

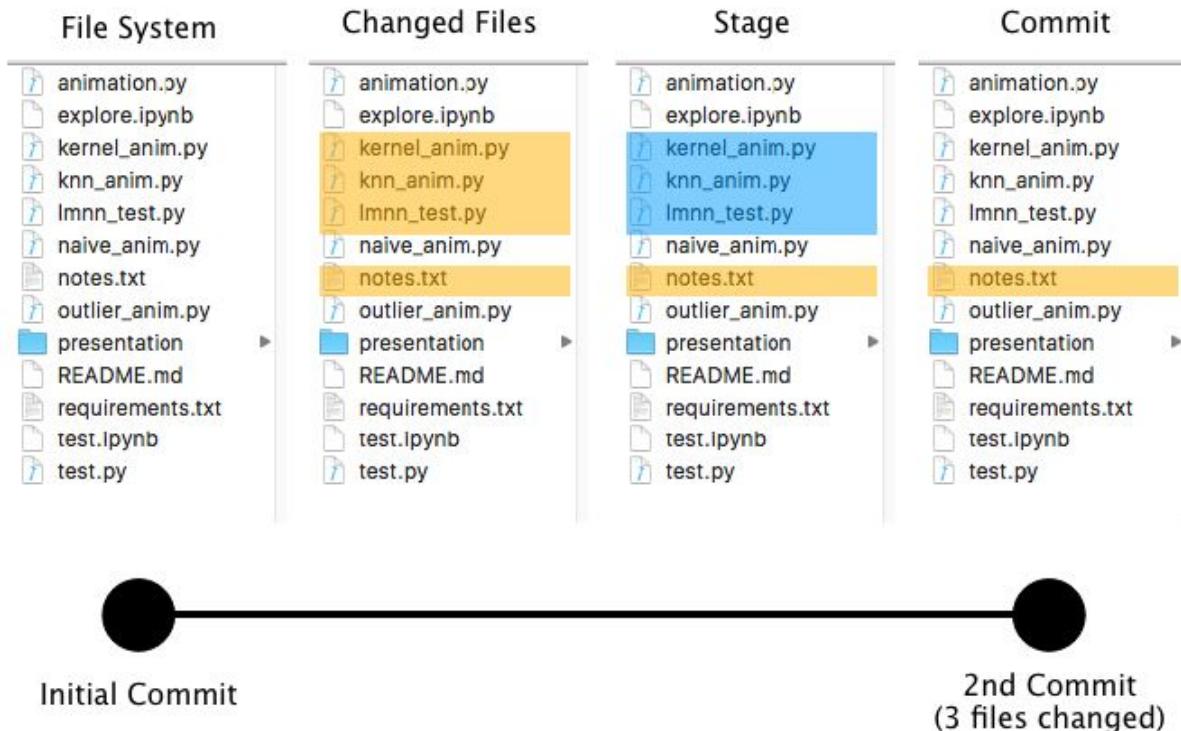
Run *git status* before any other command to know what's going on!

# Git workflow

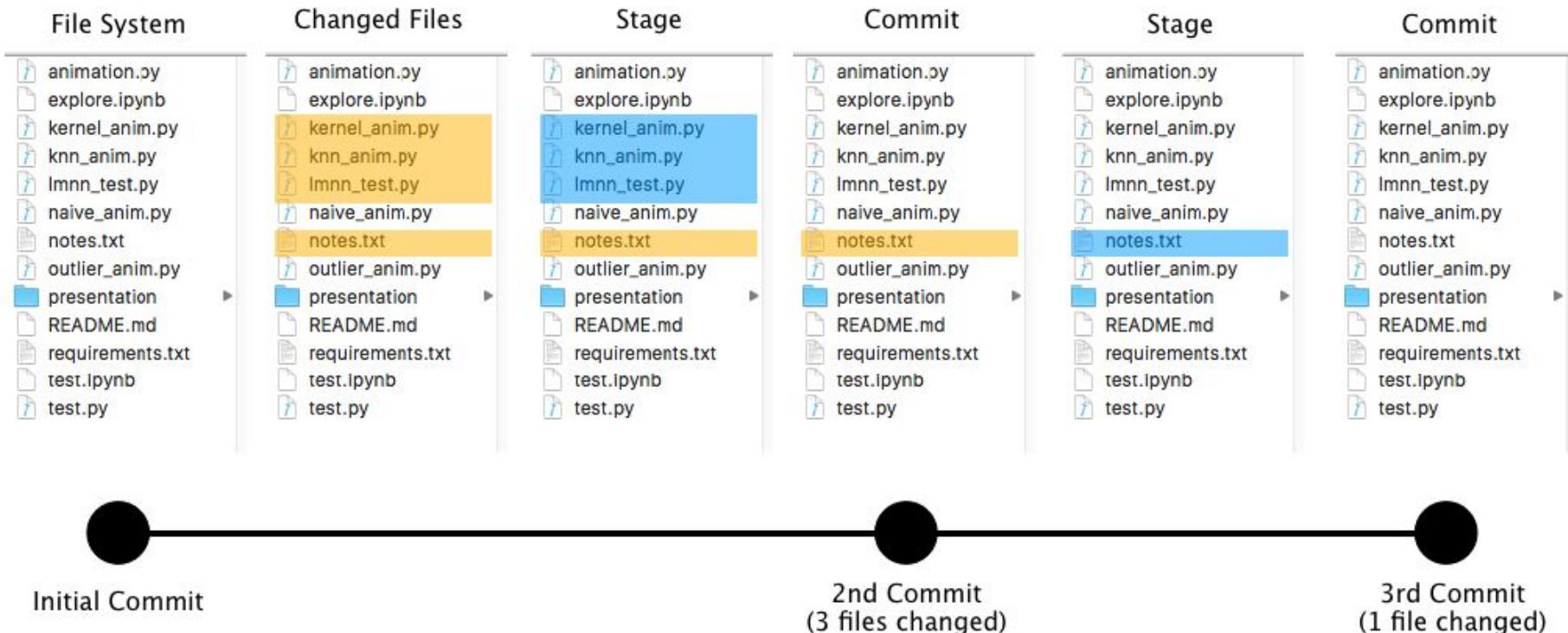
## File System



# Git workflow



# Git workflow



Initial Commit

2nd Commit  
(3 files changed)

3rd Commit  
(1 file changed)



# Getting started with GitHub

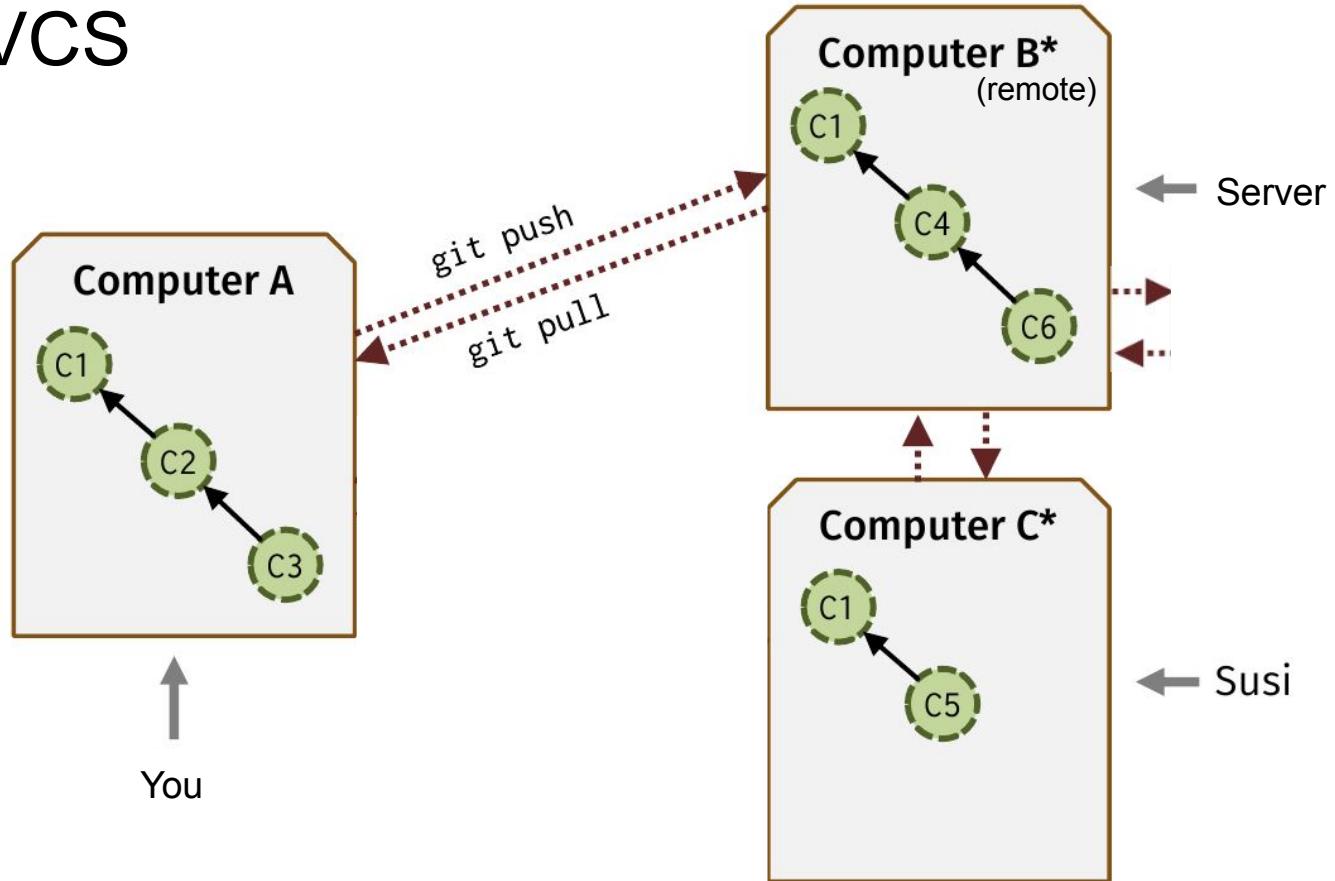
- Github is a website, providing a server to store your projects and also a web-interface to easily access them and to collaborate with others
- GitHub adds many additional features, like Pull-Requests with Code review, Issues, Wikis, ...

- 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!



[7,8]

# Distributed VCS



# Demo: git

# Demo: shells

# What to do

## 1. Create a new repository

- `git init PROJECT1`
- Creates a new folder PROJECT1
- `cd PROJECT1`

## 2. Add some files

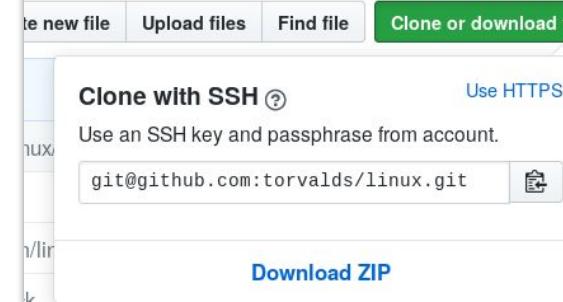
- `echo "hallo" > file.txt`
  - `ls/dir`
  - `git status`
  - `git add file.txt`
  - `git status`
  - `git commit -m "added a file"`
  - `git status`
- Folder contains `file.txt`
- Untracked files: "file.txt"
- Changes to be committed: "file.txt"
- nothing to commit, working tree clean

## 1. Clone a repository

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

## 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>

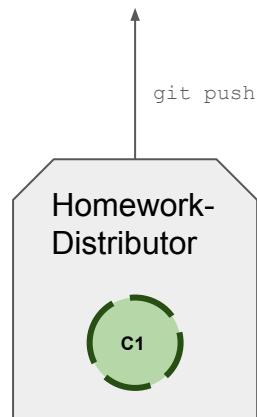
# Homework

# Homework

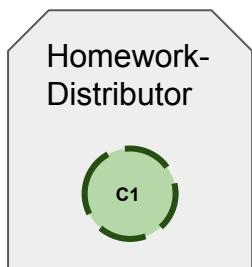
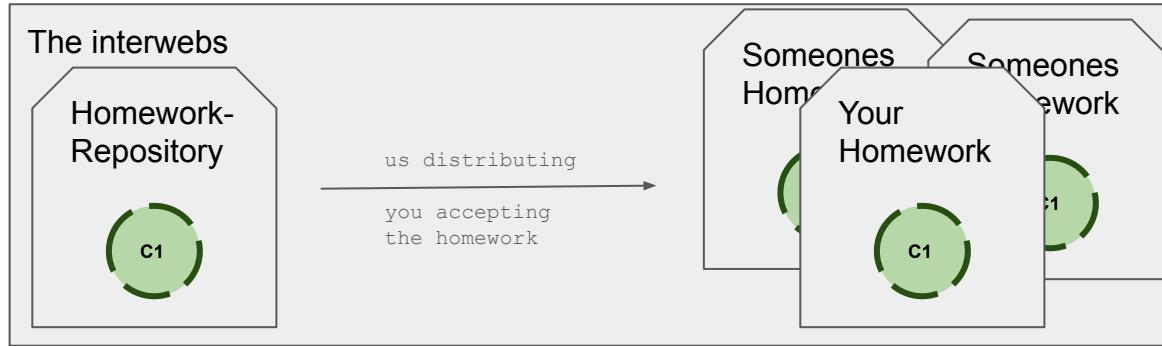
- Homework is distributed via *Github classroom*
- You need a GitHub-account to work on the homework!
- We will submit sample-solutions for the homework, and to see them, you need to at least accept the homework!

# Working with Github Classroom

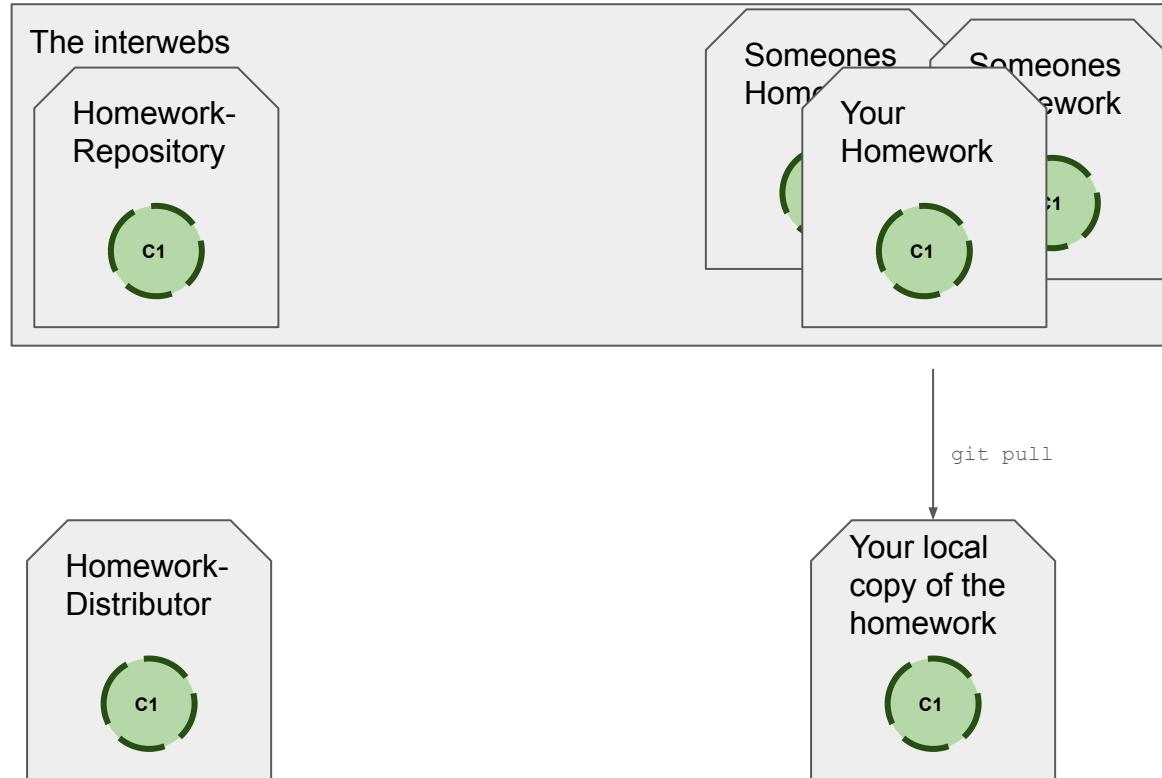
The interwebs



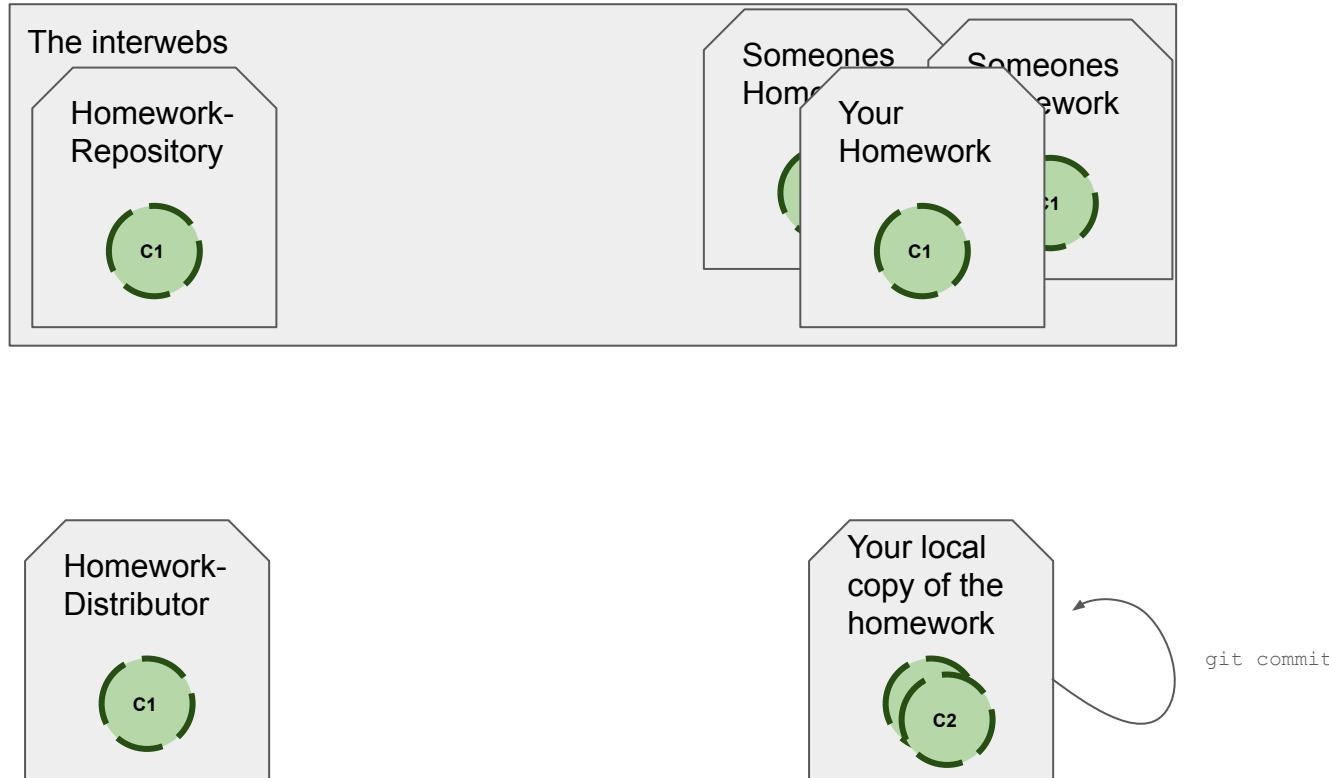
# Working with Github Classroom



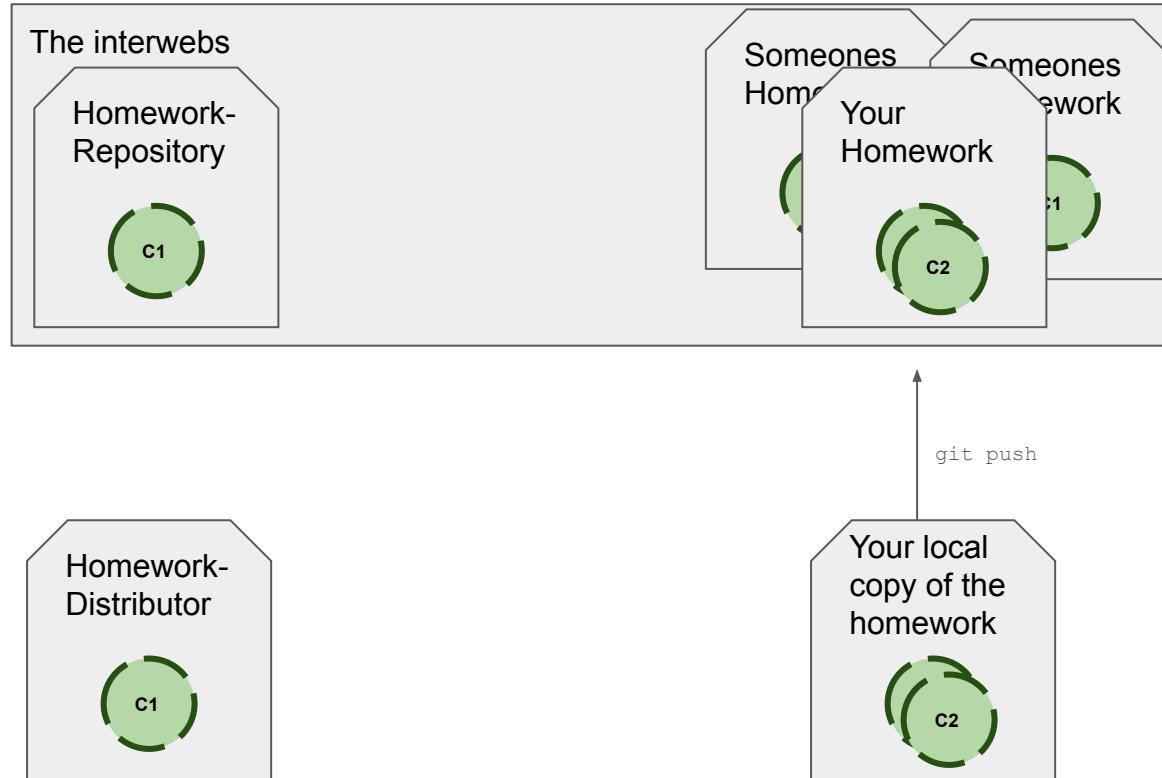
# Working with Github Classroom



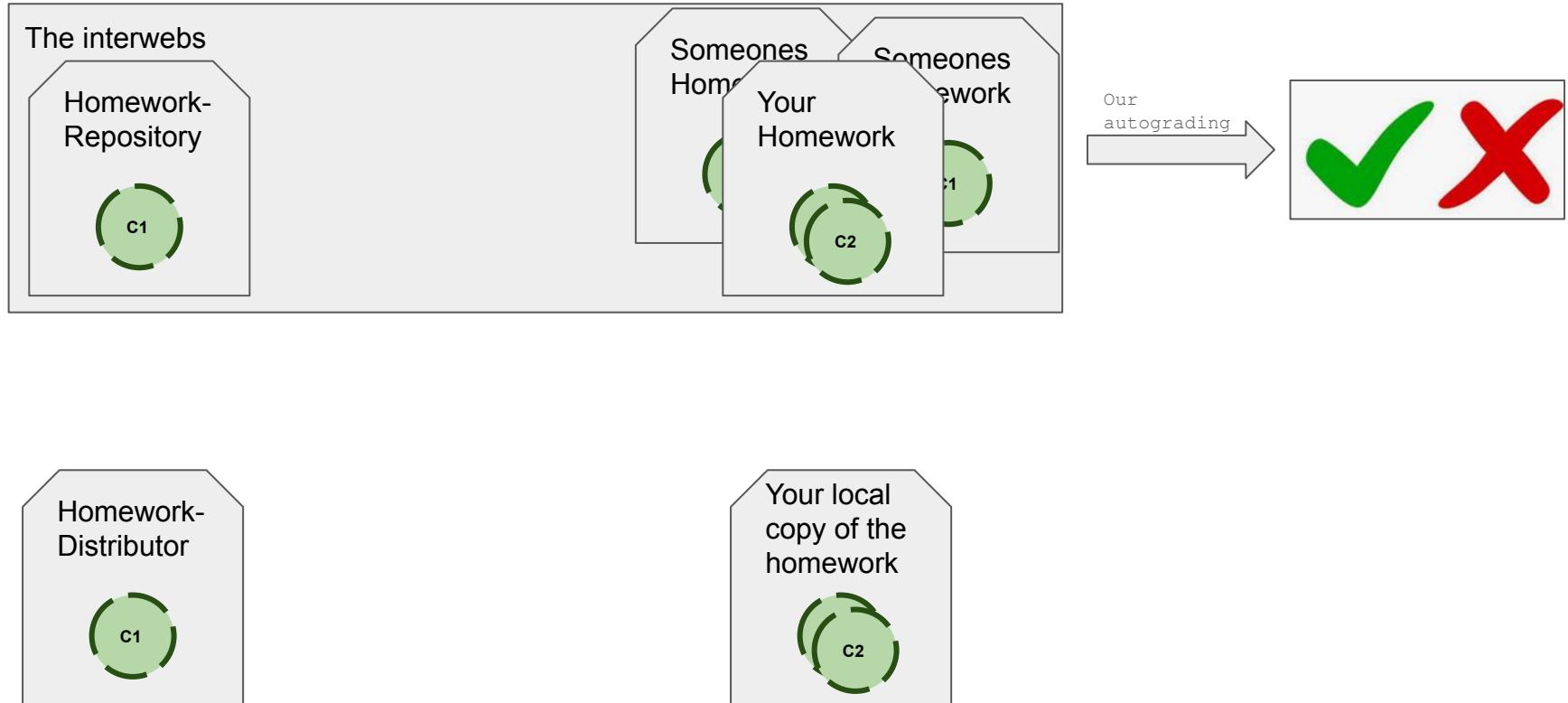
# Working with Github Classroom



# Working with Github Classroom

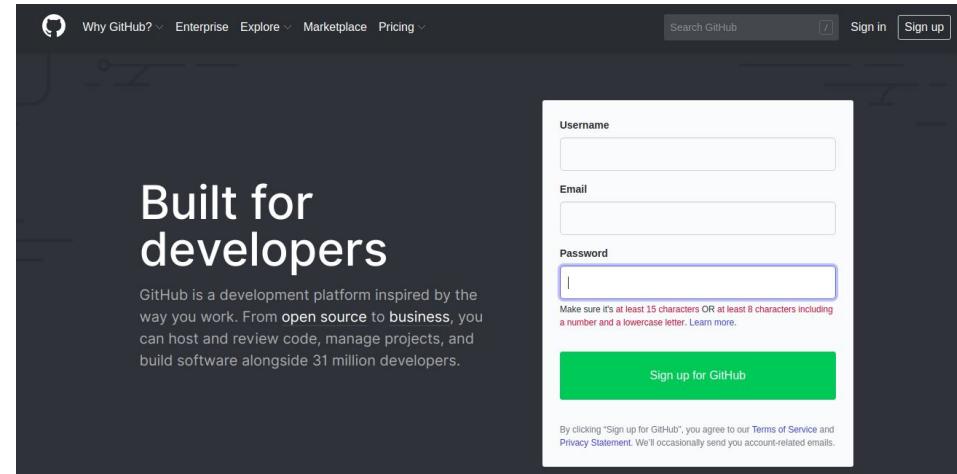


# Working with Github Classroom



# Get your homework

- Create your GitHub-Account
  - Go to <https://github.com/>
  - Your GitHub account is like your portfolio for programming, so use something you can show to others instead of xXOmqltzPotatoXx
  - Use your @uos-mail to get unlimited free repositories! <https://education.github.com/pack/offers>



# Get your homework

- Create your GitHub-Account
  - Go to <https://github.com/>
  - Your GitHub account is like your portfolio for programming, so use something you can show to others instead of xXOmqltzPotatoXx
  - Use your @uos-mail to get unlimited free repositories! <https://education.github.com/pack/offers>
- Accept the first homework for 2020 at this link:
  - <https://classroom.github.com/a/tEzM8Vux>

[Code](#)[Issues 0](#)[Pull requests 0](#)[Actions](#)[Wiki](#)[Security 0](#)[Insights](#)

2020-homework01-llloft created by GitHub Classroom

-o 3 commits

1 branch

0 packages

0 releases

2 contributors

Missing token!

Branch: master ▾

[New pull request](#)[Create new file](#)[Upload files](#)[Find file](#)[Clone or download ▾](#)

cstenkamp GitHub Classroom Autograding Workflow

Latest commit 35bafe6 8 minutes ago



Necessary for the automated grading

8 minutes ago



Makes sure you (and we) don't commit unnecessary stuff

8 minutes ago



Contains the task description (seen below)

8 minutes ago



Contains the barebone structure of the task you'll solve

8 minutes ago



List of packages you'll need in your environment for this task

8 minutes ago



The testing-files you and we use to see if your program works as intended

8 ago



# Homework 01

# 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

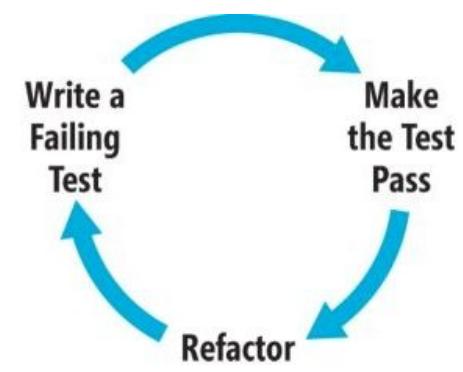
test_prime.py .F [100%]

=====
FAILURES =====
test_find_prime_method _[100%]

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

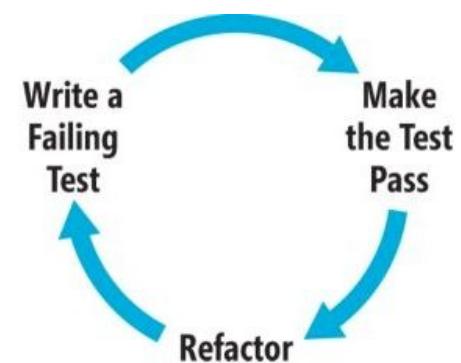
```
chris@debian:~/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01$ pytest
=====
test session starts =====
platform linux -- Python 3.6.5rc1, pytest-3.5.0, py-1.5.3, pluggy-0.6.0
rootdir: /home/chris/Documents/UNI/sem_10/Scientific_Programming_Python/homework/bonus01, inifile:
collected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

test_prime.py .. [100%]

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

```
pytest test_file_1           to run only one test-file
pytest test_file_2::test_func_1a to run only one test-function
pytest -vv                   to get longer error messages
pytest --tb=no               to only check pass/fail

pytest test_file_1 --vv --tb=0 you can combine these arguments!
```



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

# How to download and work on Homework

- Homework-link:
  - <https://classroom.github.com/a/tEzM8Vux>
  - → click on “clone or download”, copy the url

```
git clone REPOSITORY_URL
cd YOUR_REPOSITORY_PATH
conda activate scientific_programming
jupyter lab .
...
pytest
...
git status
git add CHANGED_FILE
git commit -m "solved the exercise"
git push
conda deactivate
```

# Scientific Programming in Python 2020 - Dashboard

You are logged in as sneuhoff (GitHub: illoft - #46219090)

Overall Points: 10/10

## Announcements:

### First Steps of this course

Dear Pythonists,

Welcome to this course! We spend the last days eagerly preparing everything, and so far everything is looking really well and we are happy to spend this semester with you, most likely online for the most part. As a few of you have already asked a few things about how the course will proceed, here some quick information:

- The Tuesday-session will not be held live. Instead, we will use Courseware to provide you with Videos with varying degrees of interactivity.
- The first session will be split into multiple (non-interactive) Videos that we will upload within the next three hours on Stud.IP Courseware/OpenCast
- Any other Information, also regarding how we proceed with the practice on Thursday, will be explained in the video, so please make sure to watch all of these Videos before you ask any further questions. I will update this announcement as soon as all Videos are online
- In the videos, we will explain the course infrastructure. Most of that is home-coded and completely new and largely untested, so please be patient if it doesn't work terribly smoothly at first
- If, after having watched all the videos, you have questions left that are relevant for everybody or that can also be answered by other participants of the course, please ask them in the Stud.IP Forum or on Blubber
- The course is held and organized by Philipp Thölke and Christoph Stenkamp. Besides his name on your Scheine, Prof. Kühnberger has nothing to do with this course. That means as long as it's not about that, don't ask him any questions, but ask us instead (with an email if it's supposed to be private, and via the mentioned means otherwise).

That's all for now, everything else will be explained in a video!

Here's to a good semester!

[Open Jupyter at hub.sciprog.de](#)

[Open lectures-repository](#)

[Open Stackoverflow Team](#)

[Open Stud.IP: Blubber | CourseWare | Meetings](#)

[Stud.IP Forums: | Allgemeine Diskussion | Installation Problems | Homework 1 | Homework 2 | Homework 3 | Technical Problems with the course infrastructure](#)

[Open lectures-repository](#)

## Current Homework

### homework01

Deadline: 2020-04-28 12:00 (5 days 22:11 to go)

Points: 10/10 ([check why](#))

There are 0 commits to the homework-branch since you accepted the homework.

There are 0 posts regarding this homework in the [Stud.IP Forum](#).

Stackoverflow tags: ["homework01"]

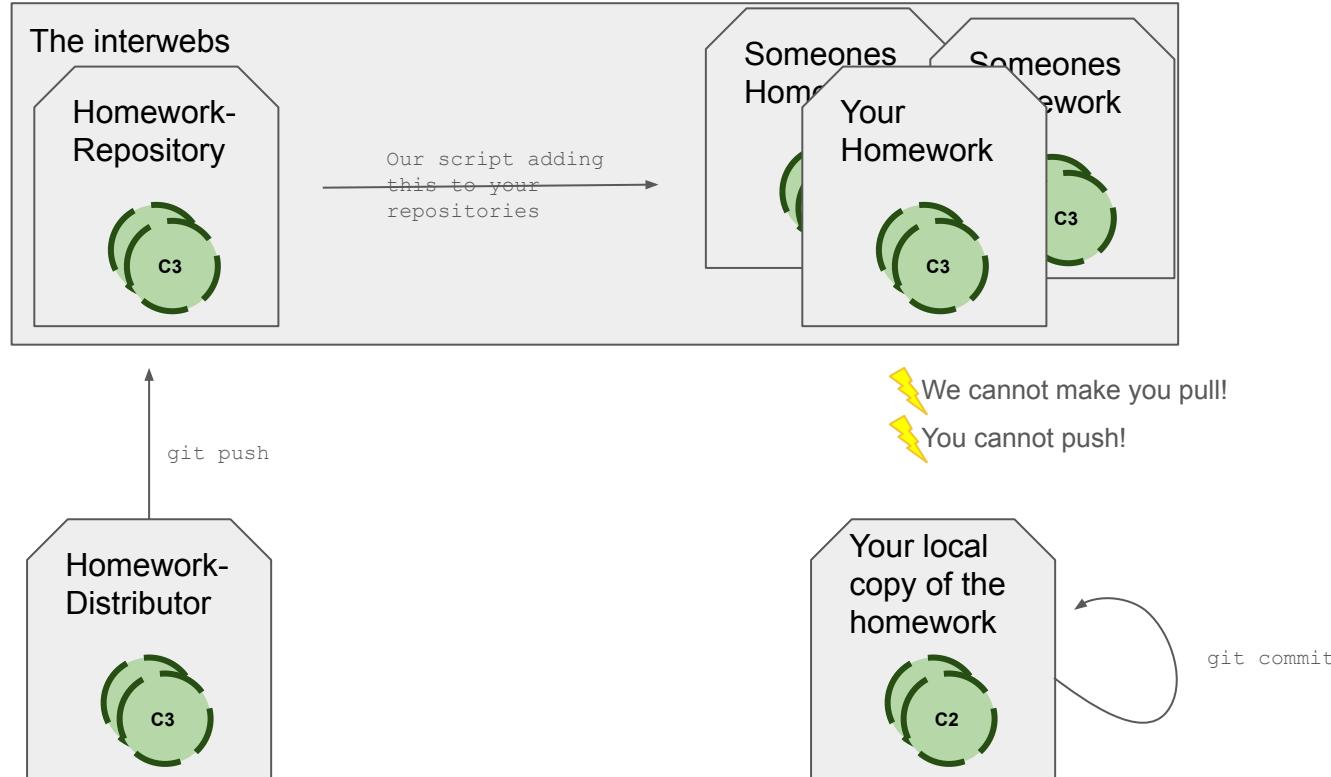
Additional information:

None

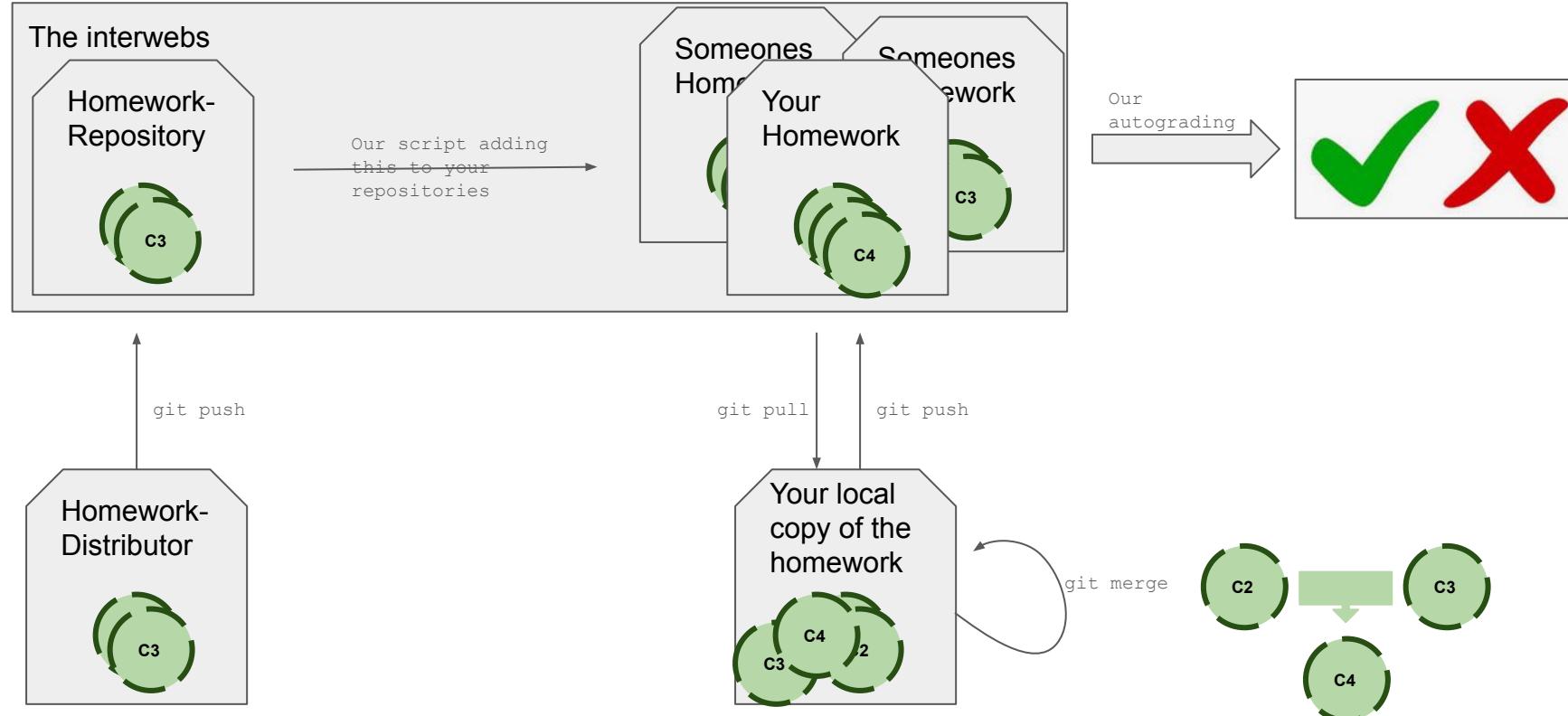
[Write an Email](#) to the admins regarding this homework

<demo: homework>

# When there are errors in the homework



# When there are errors in the homework



<Screenshots of Presentation>



Sign in to **GitHub**  
to continue to **GitHub Classroom**

Username or email address

Password

[Forgot password?](#)

**Sign in**

New to GitHub? [Create an account.](#)



## GitHub Classroom is requesting additional permissions



**GitHub Classroom by github**

would like additional permissions to



**Workflow**

Update GitHub Action Workflow files.



### Existing access

- ✓ Full control of orgs and teams, read and write org projects
- ✓ Access repository invitations
- ✓ Access user email addresses (read-only)

### Organization access



NBP-ACC ✓



scientificprogrammingUOS ✓

**Authorize github**

Authorizing will redirect to  
<https://classroom.github.com>



Owned & operated  
by GitHub

Created 5 years ago

More than 1K  
GitHub users



## GitHub Classroom is requesting additional permissions



**GitHub Classroom by github**

would like additional permissions to



**Organization webhooks**

Admin access



**Delete repositories**

Ability to delete any administrable repository



**Repositories**

Public and private



### Existing access

- ✓ Full control of orgs and teams, read and write org projects
- ✓ Access repository invitations
- ✓ Access user email addresses (read-only)
- ✓ Update github action workflows

### Organization access



NBP-ACC ✓



scientificprogrammingUOS ✓

**Authorize github**

# Join the classroom: scientificprogrammingUOS

To join the GitHub Classroom for this course, please select yourself from the list below to associate your GitHub account with your school's identifier (i.e., your name, ID, or email).

Can't find your name? [Skip to the next step →](#)

| Identifiers  |   |
|--------------|---|
| ahabdelfatta | > |
| ajaques      | > |
| annrichter   | > |
| aohnesorge   | > |
| arajendranna | > |
| ascherm      | > |
| atolkmitt    | > |
| bmajumder    | > |

If your RZ-login is not listed here, please write us an email!

Your account is linked to sneuhoff on the roster. If this is wrong, please reach out to your instructor.

X

scientificprogrammingUOS

## Accept the assignment — 2020-homework01

Once you accept this assignment, you will be granted access to the  
`2020-homework01-11loft` repository in the  
[scientificprogrammingUOS](#) organization on GitHub.

---

[Accept this assignment](#)

Code

Issues 0

Pull requests 0

Actions

Wiki

Security 0

Insights

2020-homework01-llloft created by GitHub Classroom

-o 3 commits

1 branch

0 packages

0 releases

2 contributors

Missing token!

Branch: master ▾

New pull request

Create new file

Upload files

Find file

Clone or download ▾



cstenkamp GitHub Classroom Autograding Workflow

Latest commit 35bafe5 8 minutes ago



.github GitHub Classroom Autograding Workflow

8 minutes ago



.gitignore Initial commit

8 minutes ago



README.md Initial commit

8 minutes ago



conda\_screenshot.png Initial commit

8 minutes ago



hello\_world.py Initial commit

8 minutes ago



requirements.txt Initial commit

8 minutes ago



test\_hello.py Initial commit

8 minutes ago

README.md



## Homework 01

```
Command Prompt
Microsoft Windows [Version 10.0.17763.348]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>mkdir python_homework && cd python_homework

C:\Users\User\python_homework>git clone https://github.com/scientificprogrammingUOS/2019-homework01-LeonaBaetz.git
Cloning into '2019-homework01-LeonaBaetz'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 25 (delta 10), reused 25 (delta 10), pack-reused 0
Unpacking objects: 100% (25/25), done.

C:\Users\User\python_homework>cd 2019-homework01-LeonaBaetz
```

```
C:\Users\User\python_homework\2019-homework01-LeonaBaetz>dir
Volume in drive C has no label.
Volume Serial Number is 045F-2CF3

Directory of C:\Users\User\python_homework\2019-homework01-LeonaBaetz

04/03/2019  11:42 AM    <DIR>        .
04/03/2019  11:42 AM    <DIR>        ..
04/03/2019  11:42 AM            1,339 .gitignore
04/03/2019  11:42 AM            138 .travis.yml
04/03/2019  11:42 AM            346 hello_world.py
04/03/2019  11:42 AM            6,937 README.md
04/03/2019  11:42 AM            15 requirements.txt
04/03/2019  11:42 AM            382 test_hello.py
                           6 File(s)       9,157 bytes
                           2 Dir(s)  92,013,391,872 bytes free

C:\Users\User\python_homework\2019-homework01-LeonaBaetz>conda activate scientific_programming

(scientific_programming) C:\Users\User\python_homework\2019-homework01-LeonaBaetz>pytest
=====
platform win32 -- Python 3.7.3, pytest-4.4.0, py-1.8.0, pluggy-0.9.0
rootdir: C:\Users\User\python_homework\2019-homework01-LeonaBaetz
collected 1 item

test_hello.py F
[100%]

=====
FAILURES =====
test_say_hello
-----
def test_say_hello():
    assert hasattr(hello_world, 'say_hello'), "Your Script must have an 'say_hello'-function!"

    mystdout = StringIO()
    saved_stdout = sys.stdout
    sys.stdout = mystdout
>     hello_world.say_hello()

test_hello.py:11:
-----
    def say_hello():
        """Use the print function to say 'hello world' """
        # once you created your function, you can delete the line below this one.
>     raise NotImplementedError
E     NotImplementedError

hello_world.py:4: NotImplementedError
=====
1 failed in 0.11 seconds
=====

(scientific_programming) C:\Users\User\python_homework\2019-homework01-LeonaBaetz>
```

The screenshot shows a Jupyter Notebook interface with the URL `localhost:8888/lab` in the address bar. The left sidebar displays a file tree with the following contents:

- Name: `hello_world.py` (highlighted with a red circle)
- M README.md
- requirements.txt
- test\_hello.py

The right panel is titled "Launcher" and contains three sections:

- Notebook**: A button labeled "Notebook" with a Python logo icon and "Python 3" below it.
- Console**: A button labeled "Console" with a Python logo icon and "Python 3" below it.
- Other**: A section containing two buttons:
  - Terminal**: A button with a terminal icon (\$-) and "Terminal" text.
  - Text File**: A button with a document icon and "Text File" text.

A red circle highlights the "Terminal" button in the "Other" section.

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

[scientificprogrammingUOS / 2020-homework01-llloft](#) Private Watch ▾ 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Wiki Security 0 Insights

2020-homework01-llloft created by GitHub Classroom

-o 3 commits 1 branch 0 packages 0 releases 2 contributors Missing token!

Branch: [master](#) [New pull request](#) [Create new file](#) [Upload files](#) [Find file](#) [Clone or download ▾](#)

| cstenkamp GitHub Classroom Autograding Workflow |                      |                                       | Latest commit 35baaf65 8 minutes ago |
|---|----------------------|---------------------------------------|--------------------------------------|
|   | .github              | GitHub Classroom Autograding Workflow | 8 minutes ago                        |
|   | .gitignore           | Initial commit                        | 8 minutes ago                        |
|   | README.md            | Initial commit                        | 8 minutes ago                        |
|   | conda_screenshot.png | Initial commit                        | 8 minutes ago                        |

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

 [scientificprogrammingUOS / 2020-homework01-Illoft](#) Private

generated from [scientificprogrammingUOS/homework01](#)

[Unwatch](#) 2 [Star](#) 0 [Fork](#) 0

[Code](#) [Issues 0](#) [Pull requests 0](#) [Actions](#) [Wiki](#) [Security 0](#) [Insights](#) [Settings](#)

Branch: [master](#) ▾

All checks have failed  
1 failing check

Commits on Apr 22, 2020

[GitHub Classroom Autograding Workflow](#) X GitHub Classroom Workflow / Autograding (push)… Details  [35baf65](#) 

 [cstenkamp](#) committed 15 minutes ago ×

[GitHub Classroom Autograding](#) Details  [b27b0fc](#) 

 [cstenkamp](#) committed 15 minutes ago

[Initial commit](#) Details  [4b4930b](#) 

 [JarnoRFB](#) committed 15 minutes ago

Newer Older

[scientificprogrammingUOS / 2020-homework01-Illoft](#) Private

Code Issues Pull requests Actions Wiki Security Insights

**GitHub Classroom Autograding Workflow**

master · 35ba65

**GitHub Classroom Workflow / Autograding**  
on: push

**Autograding**

**GitHub Classroom Workflow / Autograding**  
failed 16 minutes ago in 9s

**Set up job**

**Run actions/checkout@v2**

**Run education/autograding@v1**

```
42     def test_say_hello():
43         assert hasattr(hello_world, 'say_hello'), "Your Script must have an 'say_hello'-function!"
44
45         mystdout = StringIO()
46         saved_stdout = sys.stdout
47         sys.stdout = mystdout
48         >     hello_world.say_hello()
49
50     test_hello.py:11:
51     -----
52
53     def say_hello():
54         """Use the print function to say 'hello world'."""
55         # once you created your function, you can delete the line below this one.
56         >     raise NotImplementedError
57     E     NotImplementedError
58
59     hello_world.py:4: NotImplementedError
60     ===== short test summary info =====
61     FAILED test_hello.py::test_say_hello - NotImplementedError
62     ===== 1 failed in 0.03s =====
63
64     X testing: hello world
65     ::error::Error: Exit with code: 1 and signal: null
66
67     ::775a9c39-66af-48bd-bbf3-5debb7885186::
68
69 Points 0/10
```

generated from scientificprogrammingUOS/homework01

Code

Issues 0

Pull requests 0

Actions

Wiki

Security 0

Insights

Settings

Branch: master ▾

Commits on Apr 22, 2020

print helloworld, pytest passes

cstenkamp committed 32 seconds ago ✓

If your last commit passes, you  
can be sure you'll get full credit  
for this exercise!

GitHub Classroom Autograding Workflow

cstenkamp committed 20 minutes ago ✗

GitHub Classroom Autograding

cstenkamp committed 20 minutes ago

Initial commit

JarnoRFB committed 20 minutes ago

Newer

Older

# Thanks for your attention!

- Any questions and remarks please via the mentioned means!
- Content-suggestions are always welcome!

# Sources

1. <https://commons.wikimedia.org/wiki/File:Python.svg>
2. <https://pixabay.com/vectors/swiss-army-knife-pocket-knife-blade-154314/>
3. [https://en.wikipedia.org/wiki/R\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/R_(programming_language))
4. [https://commons.wikimedia.org/wiki/File:Matlab\\_Logo.png](https://commons.wikimedia.org/wiki/File:Matlab_Logo.png)
5. [https://commons.wikimedia.org/wiki/File:Images\\_200px-ISO\\_C%2B%2B\\_Logo\\_svg.png](https://commons.wikimedia.org/wiki/File:Images_200px-ISO_C%2B%2B_Logo_svg.png)
6. [https://pt.wikipedia.org/wiki/Julia\\_\(linguagem\\_de\\_programa%C3%A7%C3%A3o\)](https://pt.wikipedia.org/wiki/Julia_(linguagem_de_programa%C3%A7%C3%A3o))
7. [https://commons.wikimedia.org/wiki/File:Git\\_icon.svg](https://commons.wikimedia.org/wiki/File:Git_icon.svg)
8. [https://farm2.staticflickr.com/1482/24588096069\\_59a0513790\\_z.jpg](https://farm2.staticflickr.com/1482/24588096069_59a0513790_z.jpg)