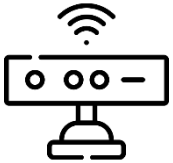
Institute of Automotive Technology
Faculty for Mechanical Engineering
Technical University of Munich

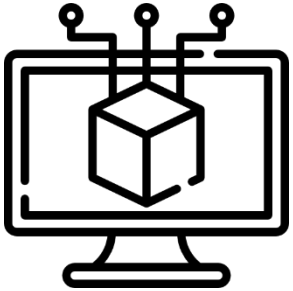
Autonomous Driving Software Engineering

Prof. Dr.-Ing. Markus Lienkamp

Dipl. Ing. Nico Uhlemann

Simon Sagmeister, M.Sc.





Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

1. Introduction
2. Homework
3. Programming Environment
4. System Setup
5. Working on your weekly homework

Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

1. **Introduction**
2. Homework
3. Programming Environment
4. System Setup
5. Working on your weekly homework

Practice – General Information

Idea

- The lecture's focus is on software. Software means coding and coding requires experience and practice.
- The lectures give the overview about the applied algorithm and software methods, the practice sessions show the application.

Procedure of the practice sessions

- Practical coding examples will be presented
- The idea is that you get a practical insight to specific algorithms and software modules
- The practice sessions are interactive, so you are invited to ask questions and discuss the presented software
- Additionally, there will be a weekly homework you can do on your own at home (1 week time)

Practice and Homework – Coding Skills

WARNING

We are doing coding tutorials and coding tests in this lecture which are based on Python3 programming.

Knowing how the Python3 syntax works is a prerequisite of this lecture.

If you do not have Python3 experience, please do this free tutorial, where you can learn the syntax:

Online Tutorial: https://www.learnpython.org/

Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

1. Introduction
2. **Homework**
3. Programming Environment
4. System Setup
5. Task submission

General Information

- The homework builds upon the practice sessions, so attendance to the practice sessions is helpful
- Each homework is only available for one week:
  - (this) Tuesday, 0:00 – (next) Tuesday, 23:59
- There is no time limit for each task of the homework within the week
- You can stop a task and restart any time within this week
- You can change your answers as long as the homework is still open
- You can submit your answers and evaluate it on your own to check if everything is correct
- If you do the homework regularly and achieve a score of **50% correct answers** (mean value calculated from each homework) by the end of the semester

  → You will get a bonus on your final grade of **+0.3**

  → Note: The bonus will only be given if you passed the exam regularly, i.e. your exam result has to be 4.0 or better

Framework Information

- The homework consists of small coding tests you need to solve
- You can access the homework on the lecture's website: https://adse.ftm.ed.tum.de/
- We are using CodeFREAK to provide the task and for you to submit your results
- To work on the task we recommend to use PyCharm + Anaconda
- We use **Python3.8** for all coding tasks
- The in- and output of each tasks are described within the homework. Please read the description properly

Honor Code Rules

Rule 1
Prior to each submission, it must be indicated whether you got assistance for developing the code and who assisted you in the development.

Rule 2
No current program code or solutions to individual problems should be shared with other students.

Rule 3
Solutions must not be uploaded to or shared via the internet.

Rule 4
All students must be able to explain the homework submitted at all times.

Evaluation

- After you submitted the homework, you must run the evaluation on your own
- For each task, there are multiple points to receive:
  - → Each task has a different number of points
  - → Each task will be checked by **Pytest**, an automated **Code Checker**
  - → For some homework, you will get points for your **Code Style**, checked via **Flake8**
  - → The next points are granted when completing the unittests successfully, done via **Pytest**
  - → Your code will be automatically evaluated through these tests, so no manual evaluation will be given. You can check if your code passes the tests a priori.
- After the submission you will see the evaluation of each task separately.

Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

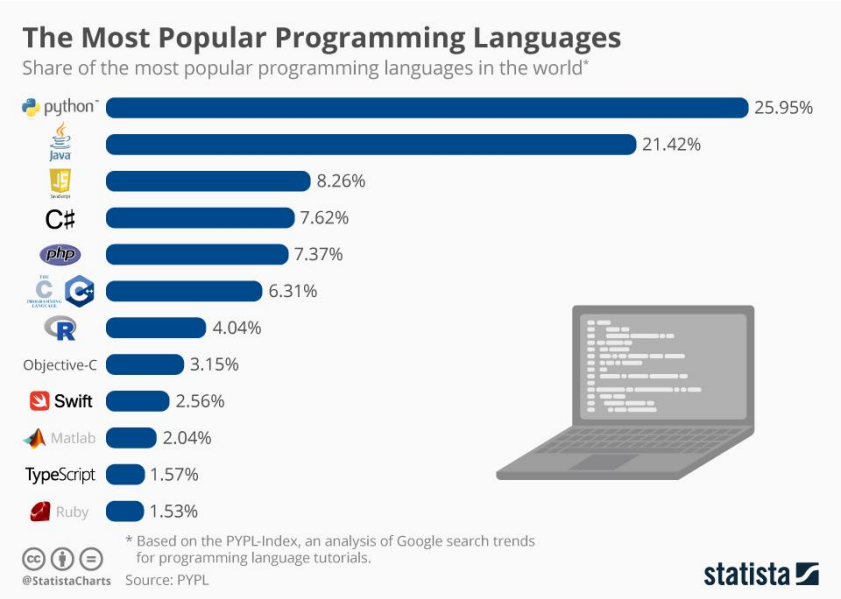
1. Introduction
2. Homework
3. **Programming Environment**
4. System Setup
5. Working on your weekly homework

Programming environment

- Homework and practices consist of programming tasks using Python 3.8.
- Being able to code in Python is a prerequisite of this lecture.
- If you don‘t have Python experience, have a look in the following tutorial: https://www.learnpython.org/
- The programming environment consists of: Anaconda, PyCharm, Jupyter Notebook and CodeFREAK.

Why Python?

- Python is an interpreted, high-level, general-purpose programming language.
- Most popular scientific programming language and is open-source
- Used for Deep Learning, Data Science, Rapid Development, etc.



Python Packages

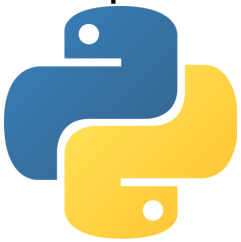
- Python scripts usually start with import commands. You don‘t program everything yourself. It is an open source community!

```python
import numpy as np

list_in = [1, 2, 3, 10.0, 23.3]

# Mean of a list ##
np.mean(list_in)
```

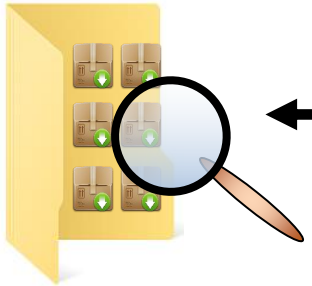
```python
# import packages
import numpy as np
from joblib import load, dump
from sklearn.model_selection import train_test_split, ParameterSampler
from tensorflow.keras.models import load_model
```

Code with imports

Interpreter

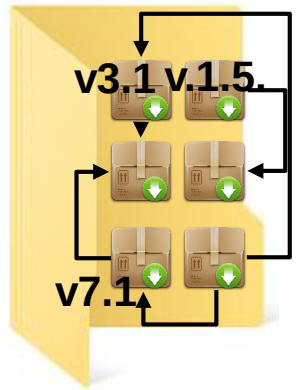


Package directories



Package Management

- Python packages experience constant change (multiple versions).
- Python packages may have interdependencies.
- Package managers (e.g. Anaconda) handle installation, versioning updates, mutual dependencies and even come with an actual interpreter.



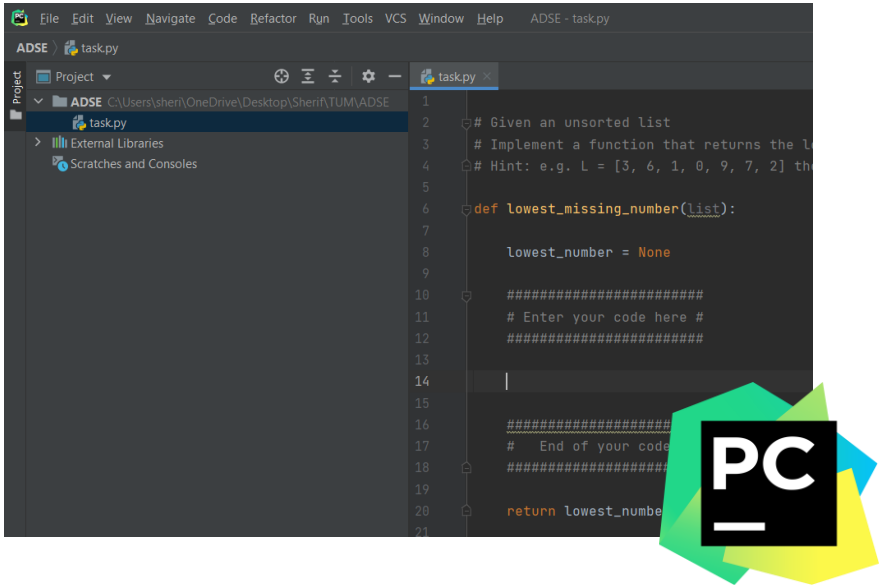
Package Directories

ANACONDA®

Integrated Development Environment (IDE)

- For actual coding in Python you need an editor or an integrated development environment (IDE).
- IDEs include additional functionalities compared to a editor, such as debugging, compiling and many more.

→ PyCharm is, where you code your functions and homework.

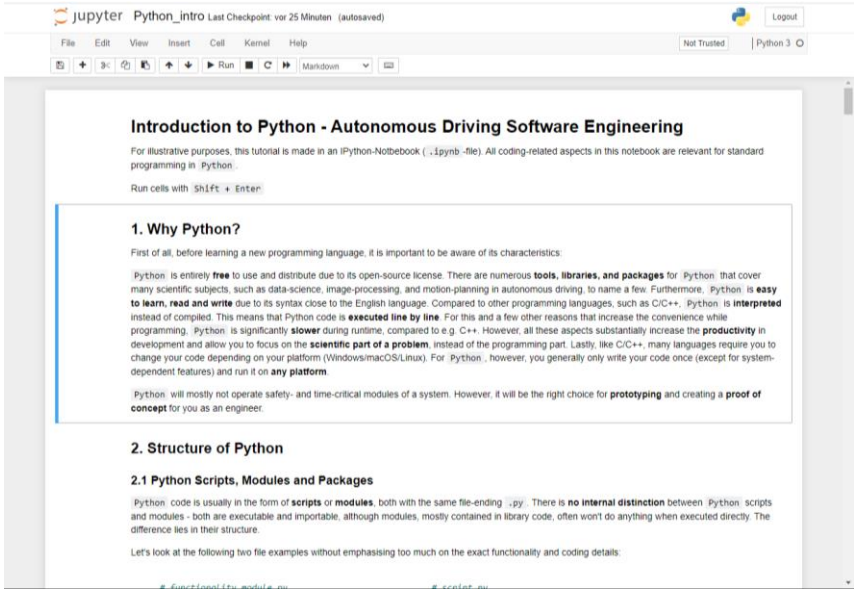


PyCharm IDE

Jupyter Notebook for Practice Sessions

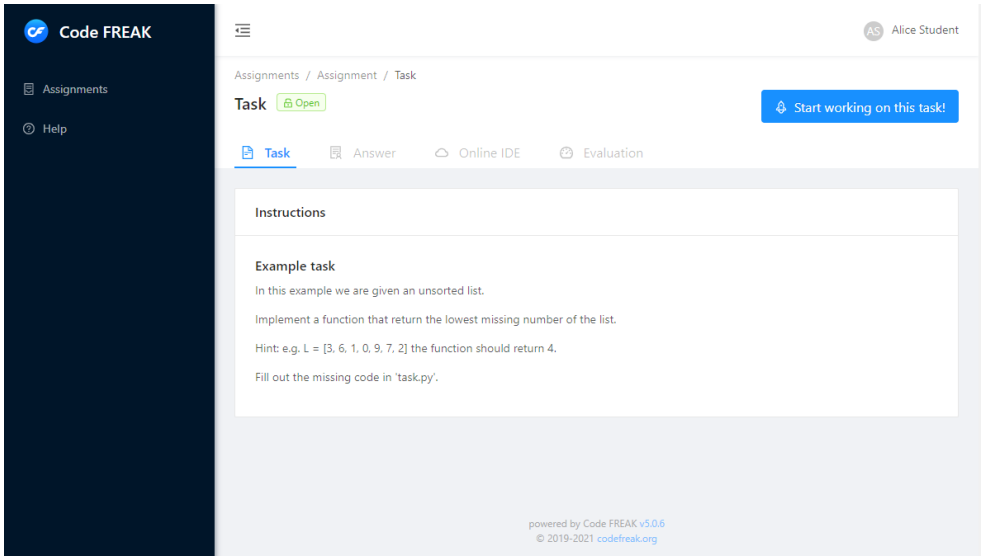
- During the lecture we will show coding examples in Jupyter Notebook

*The **Jupyter Notebook** is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. **Uses** include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more*



Homework

- Idea: Focus on Practice and Coding
- Automated Testing and Evaluation via Code FREAK
- You will get homework, which you can download and submit on a separate domain: https://adse.ftm.ed.tum.de/



Code FREAK task submission page

Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

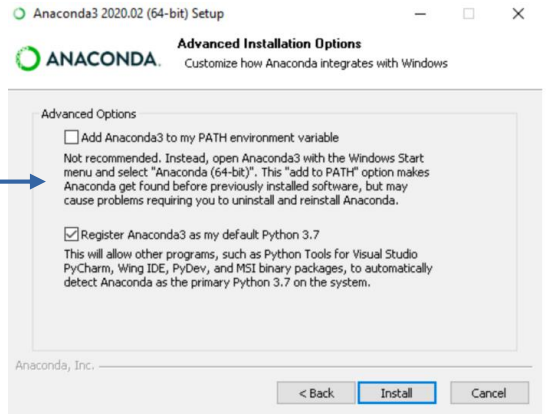
1. Introduction
2. Homework
3. Programming Environment
4. **System Setup**
5. Working on your weekly homework

Your task at home

Setting up your system – Anaconda Navigator

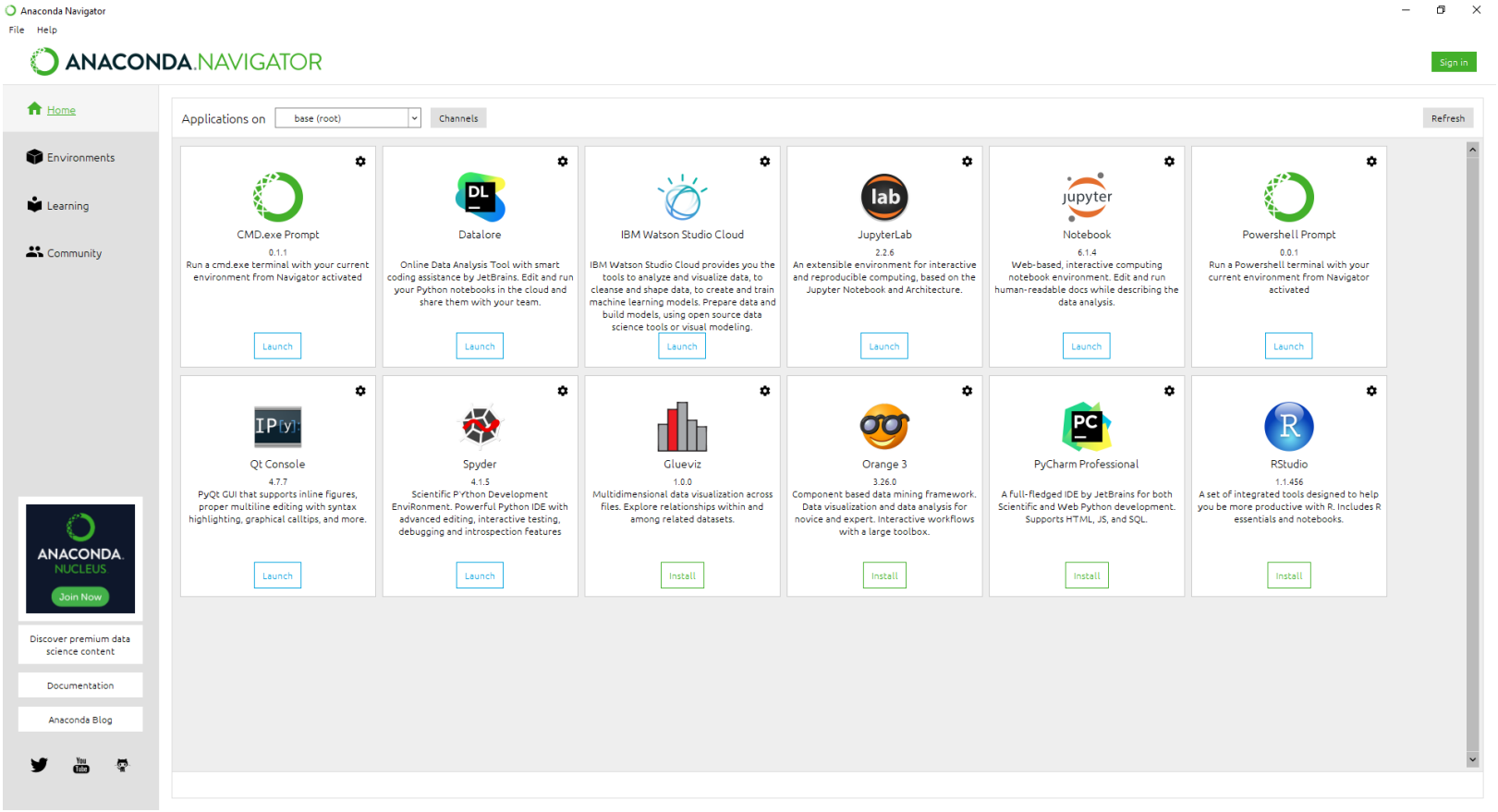
- Anaconda Navigator combines Python (Explicit installation of Python interpreter not needed) with a GUI to setup project environments, to install packages and to launch your IDE.
- Download the most recent version available on the Anaconda homepage: https://www.anaconda.com/products/individual
  - Go to **Download** and select the Python 3.8 version.
  - After the download, start the installation, for further information: https://docs.anaconda.com/anaconda/install/

Warning: When installing on Windows, do **not** add Anaconda to the PATH environment.



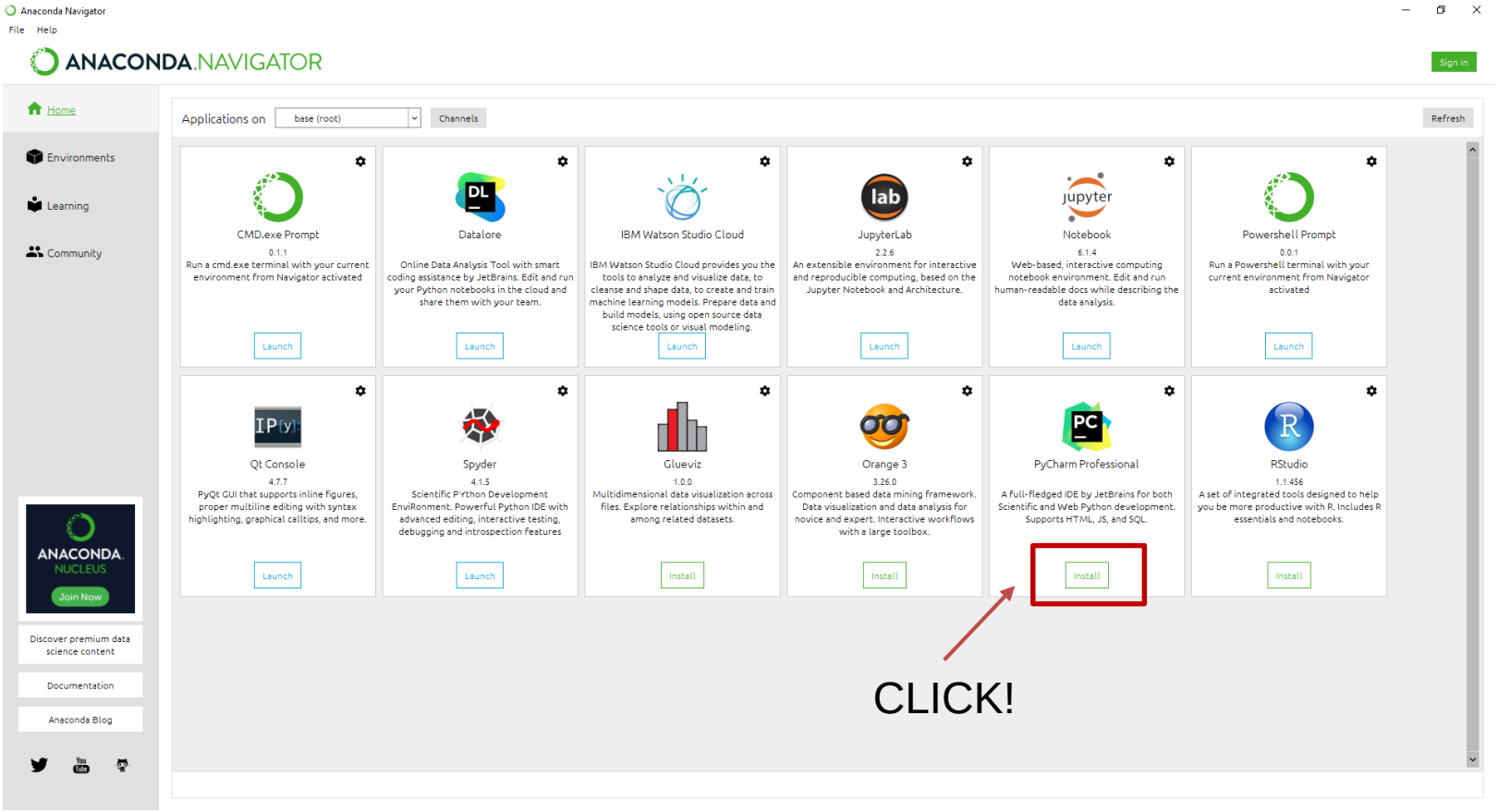
Setting up your system – Anaconda Navigator

- Launch the Anaconda Navigator.



Setting up your system – PyCharm

- Install PyCharm Professional.



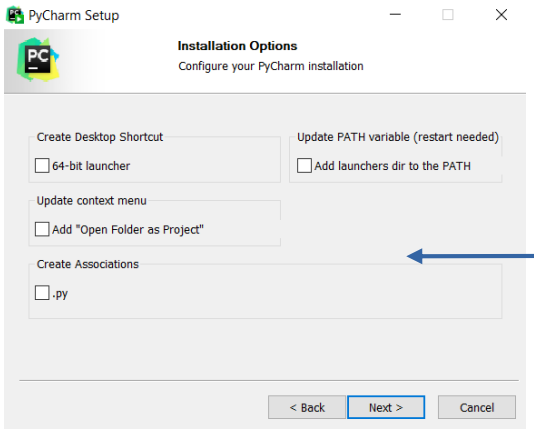
Setting up your system – PyCharm

- Install PyCharm from the Anaconda Navigator.
- Default download is **pro**-version of PyCharm.
  - Free for students but requires registration with your <TUMID>@mytum.de email-address under: https://www.jetbrains.com/shop/eform/students
  - Your mail address is then registered as JB-Account.
- **Alternative**: Chose the community edition which is free from registration at: https://www.jetbrains.com/pycharm/download/

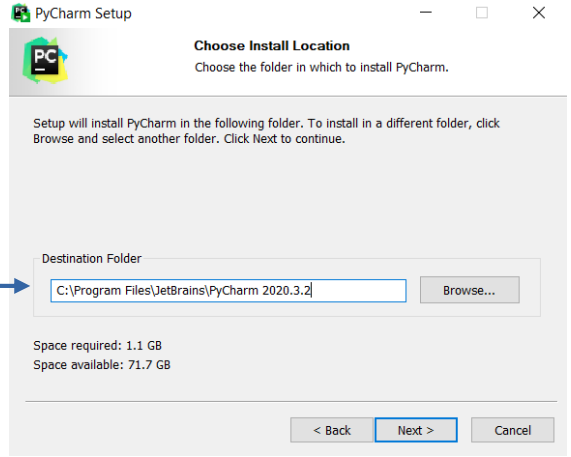
Refresh the Anaconda Navigator. Updates your available applications.

Setting up your system – PyCharm

Pro / Community edition steps:

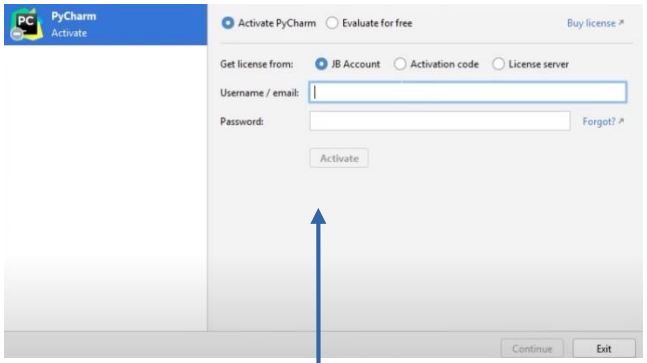


Nothing to select, as we will always launch PyCharm from the Anaconda Navigator.



Don’t change path. Otherwise Anaconda might have trouble finding PyCharm.

Additional step for PyCharm pro:



Activate PyCharm with your TUM mail adress after registration.

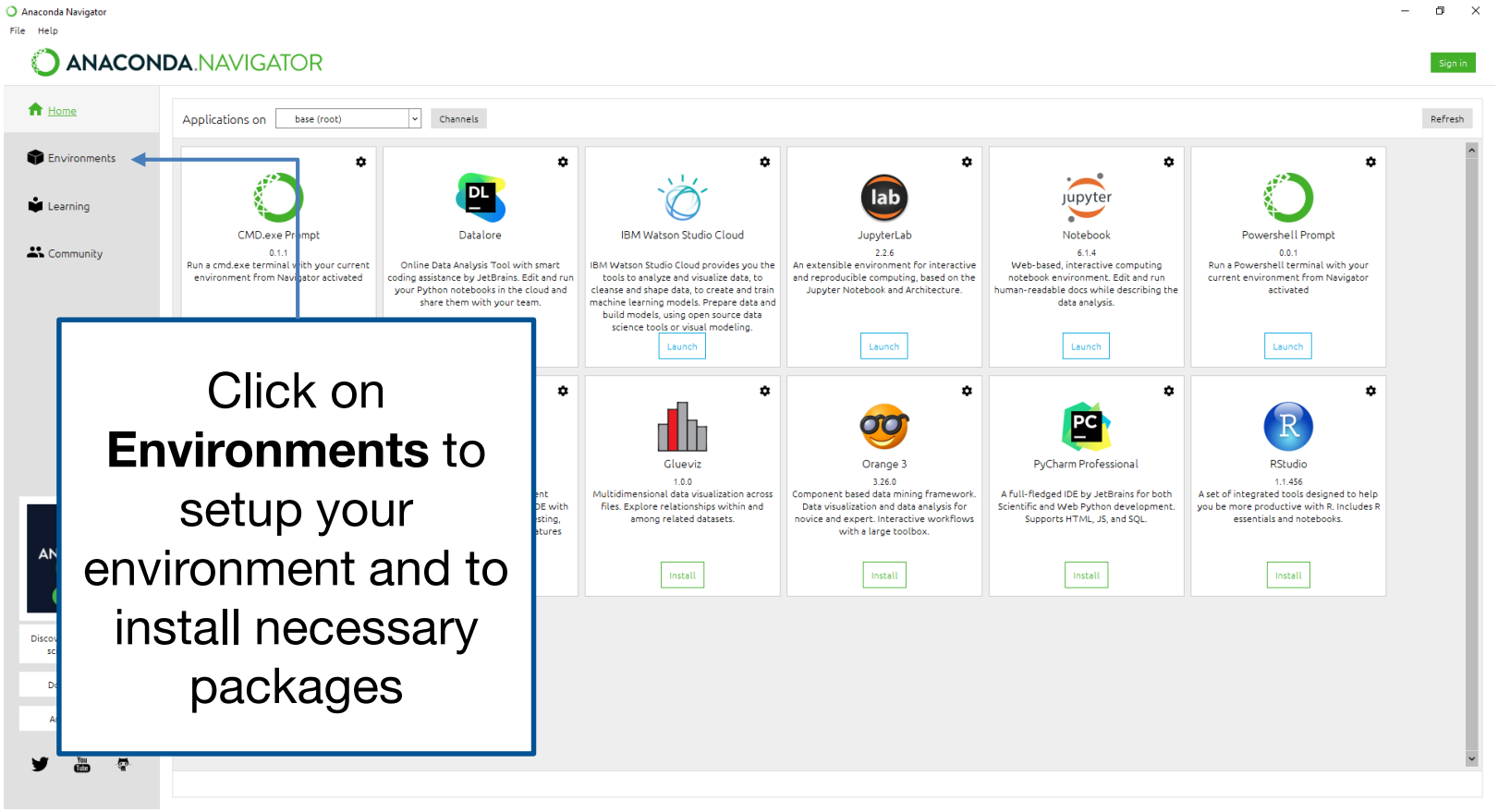
Setting up your system – Folder structure

To keep the overview about your weekly homework, we recommend a folder structure, which looks like this:

```
C:\Users\<username>\source\ADSE_2023
    python_intro
        python_intro.ipynb
        <Here your can place the Repetitorium>
    homework_01
        <Here you can create your weekly PyCharm Projects>
        task1
            task1.py

            …
        task2
            …
    homework_02
        task1
            …
```

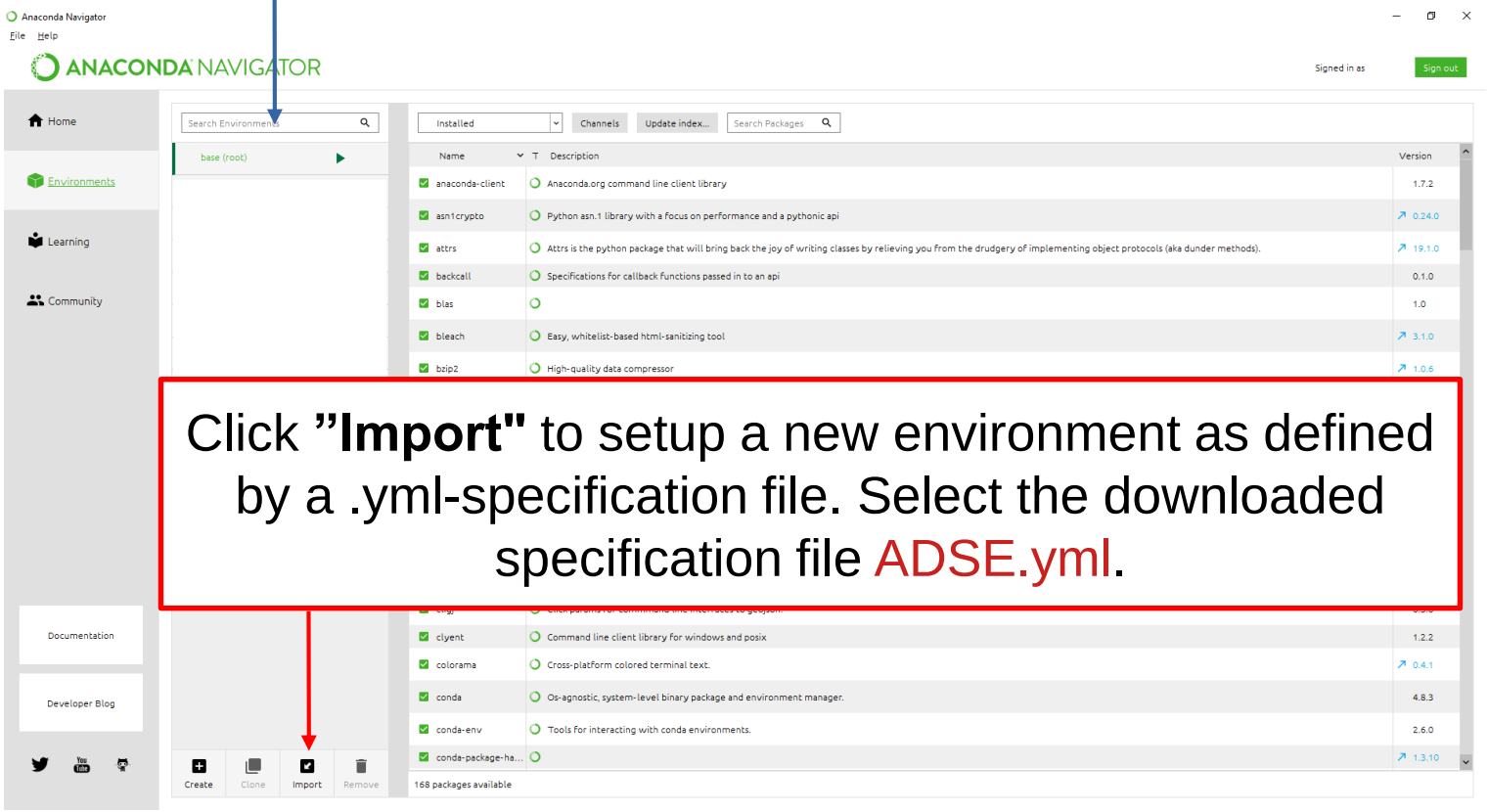
Setting up your system – Environment

- Return to the Anaconda navigator and navigate to the environments section.



Setting up your system – Environment

The "base" environment is available after the installation of Anaconda containing commonly used python packages.



Click **"Import"** to setup a new environment as defined by a .yml-specification file. Select the downloaded specification file ADSE.yml.

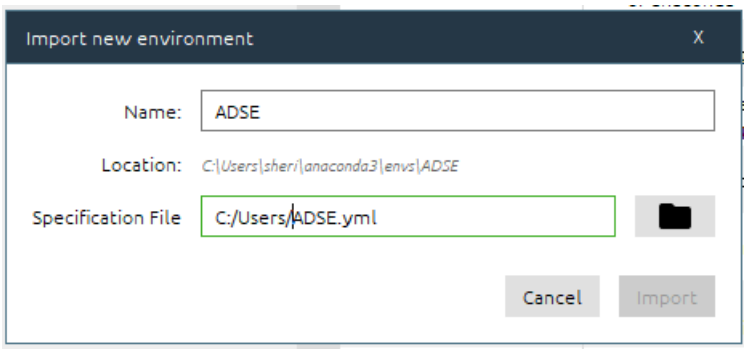
Setting up your system – Environment

- The necessary specification file can be downloaded from our Moodle course and looks as follows:

ADSE.yml

```yaml
name: ADSE
channels:
  - conda-forge
  - defaults
dependencies:
  - python=3.8
  - numpy=1.18.1
  - matplotlib=3.2.1
  - opencv=4.2.0
  - moviepy=1.0.1
  - scikit-learn=0.22.2
  - scikit-image=0.16.2
  - ipywidgets=7.5.1
  - folium=0.10.1
  - pandas=1.0.3
  - shapely=1.7.0
  - geopy=1.21.0
  - haversine=2.2.0
  - tensorflow=1.13.1
```

Required dependencies are listed with their package versions

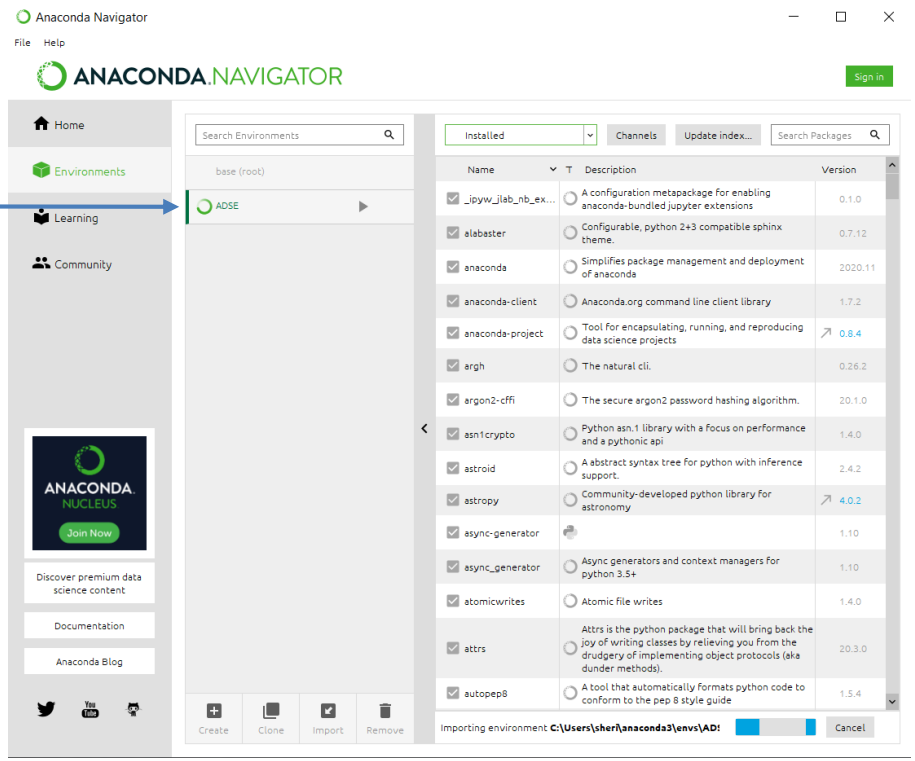


Select the downloaded specification file named ADSE.yml

Setting up your system – Environment

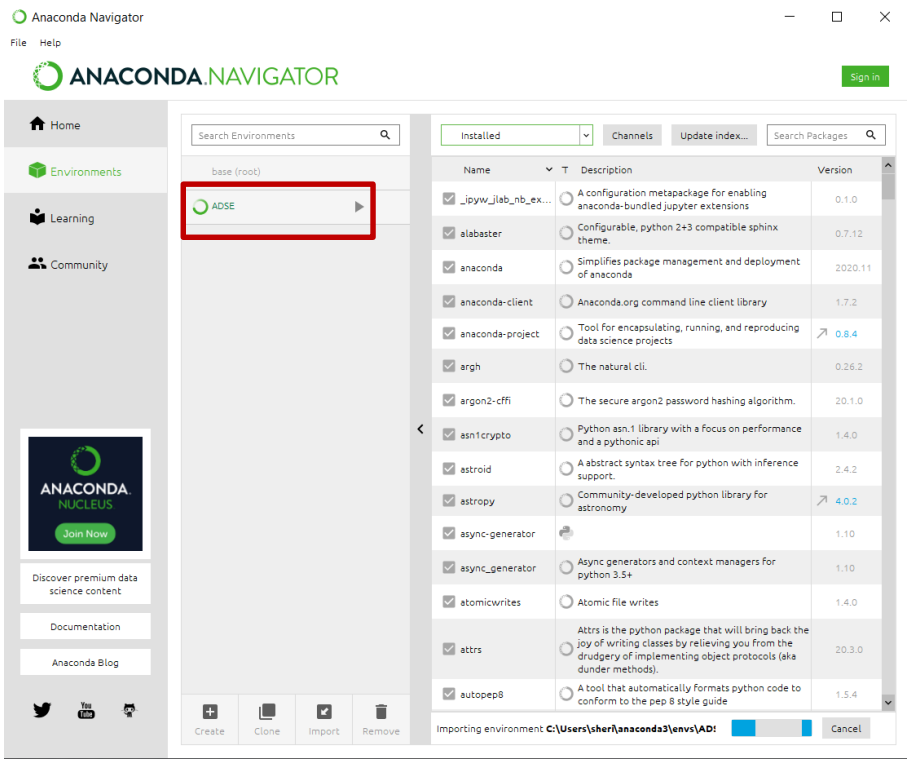
- The installation process takes some time because several packages need to be downloaded and installed.

There is a new button available after the setup of a new environment.



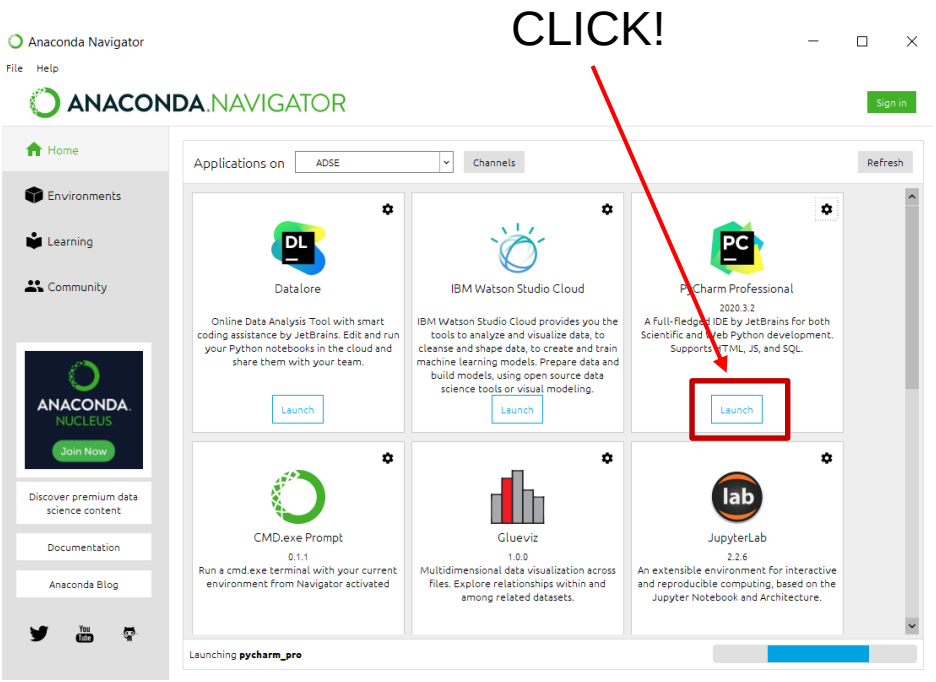
Setting up your system – Environment

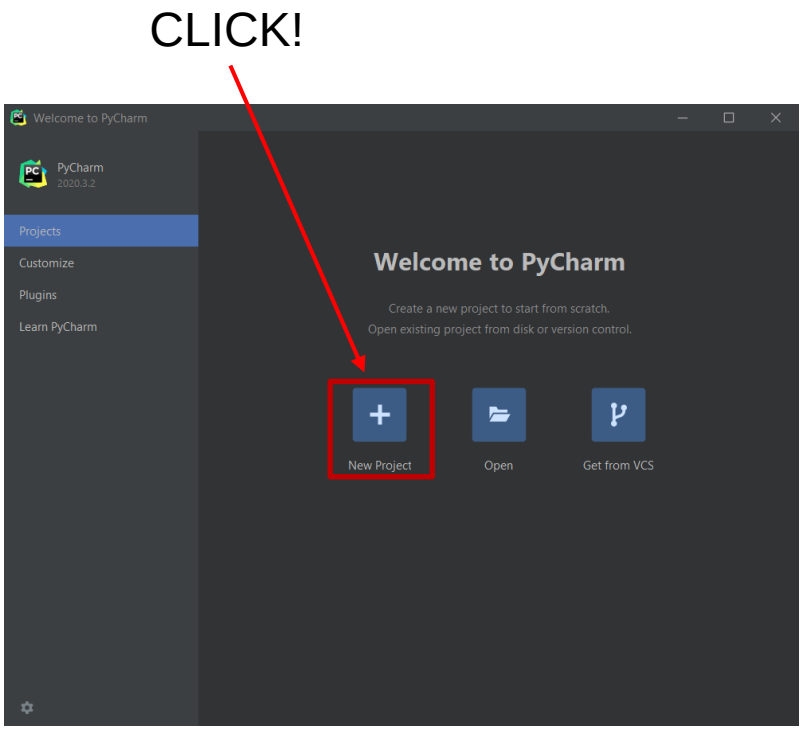
- Click on the button “ADSE” (name of the environment) to activate this course’s environment.



Setting up your system – Environment

- Launch PyCharm and begin with a new project.

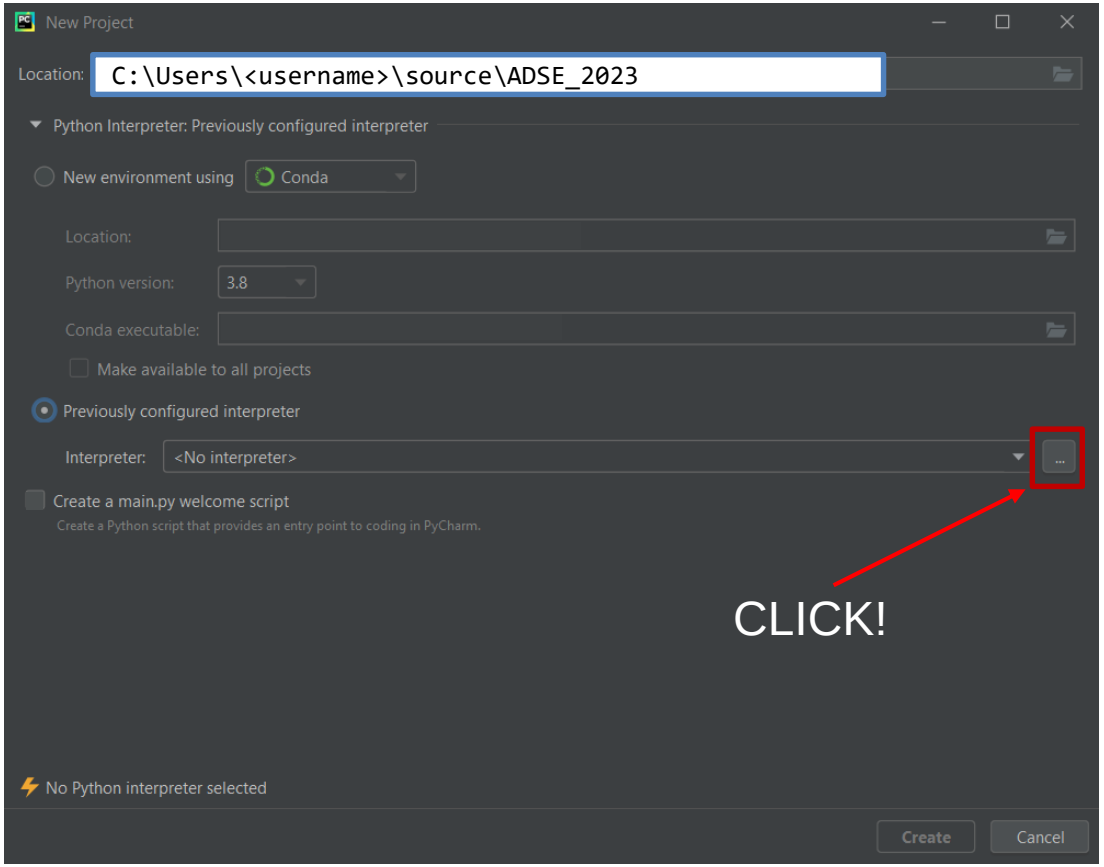




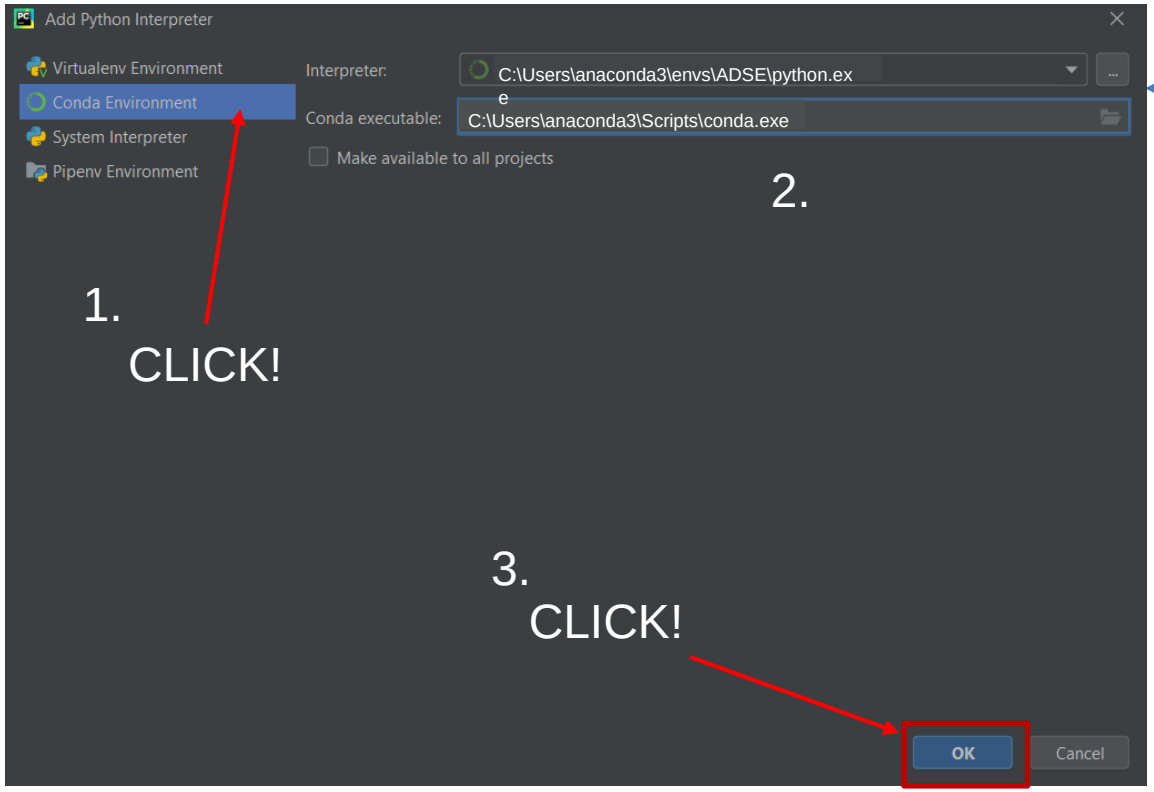
Setting up your system – Environment

The new project should be at the location of the weekly homework, **see Slide 1-25**.

Important: Instead of generating a new environment, we want to use the previously configured environment from Anaconda!

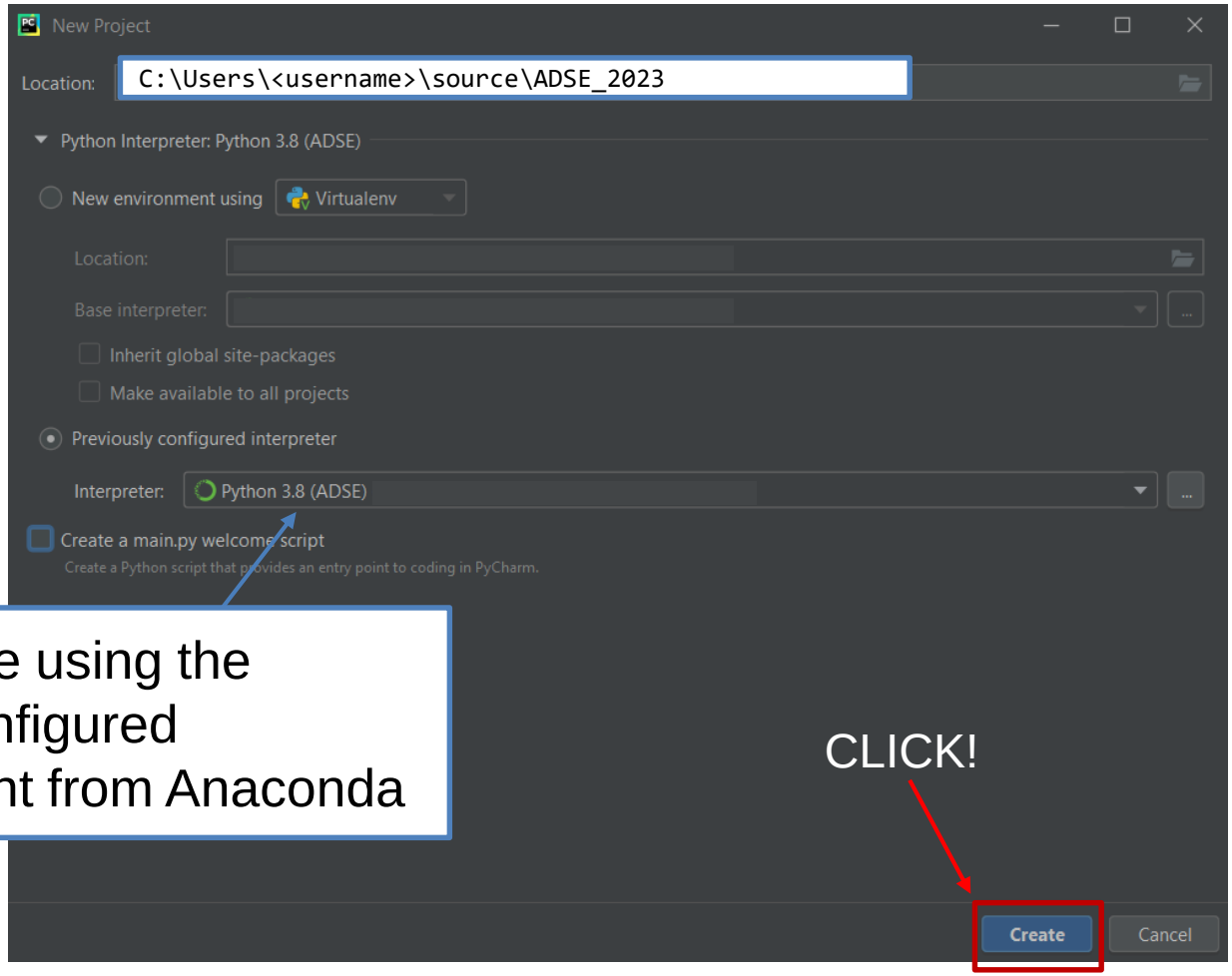


Setting up your system – Environment



Pycharm should automatically choose the correct Conda executable and the interpreter. If not navigate to the according folders as shown. Choose the Choose the python.exe-file in the ADSE Folder.

Setting up your system – Environment

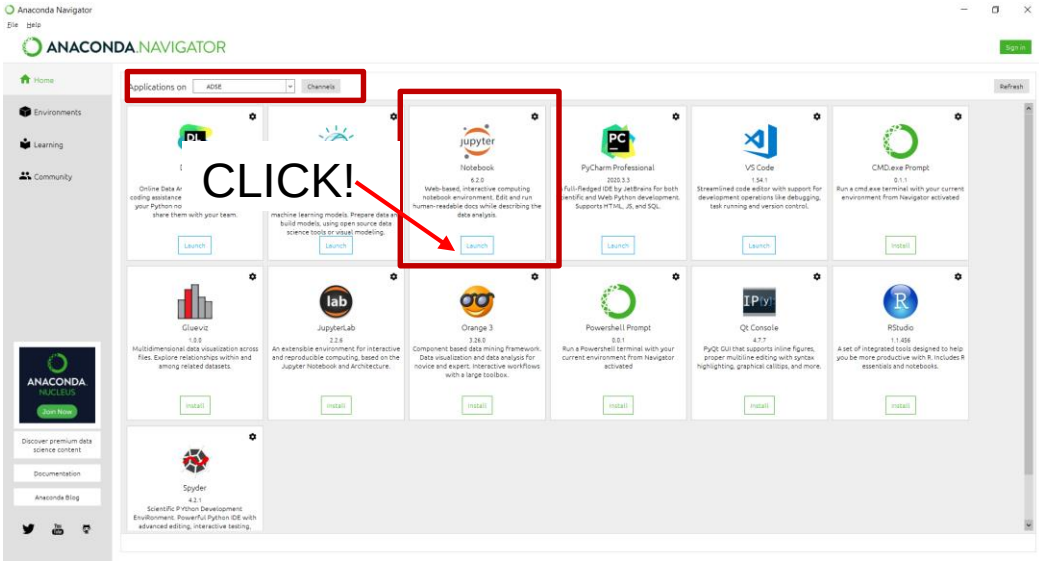


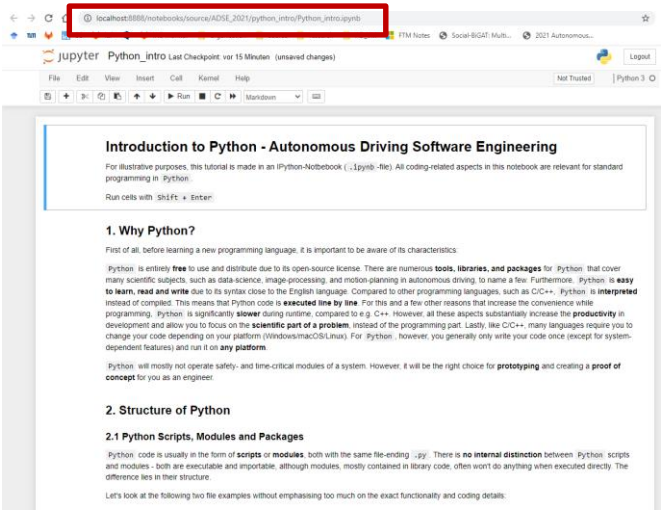
Now we are using the already configured environment from Anaconda

CLICK!

Setting up your system – Jupyter Notebook

- During the practice, we will show Code Examples in Jupyter Notebook. If you want to repeat the practice session on your own at home, you need to start Jupyter Notebook.
- To do so just (*install* &) *launch* Jupyter Notebook within your *ADSE-Environment.*
- Then navigate to the desired directory and run the .ipynb-files





Autonomous Driving Software Engineering
Prof. Dr. Markus Lienkamp

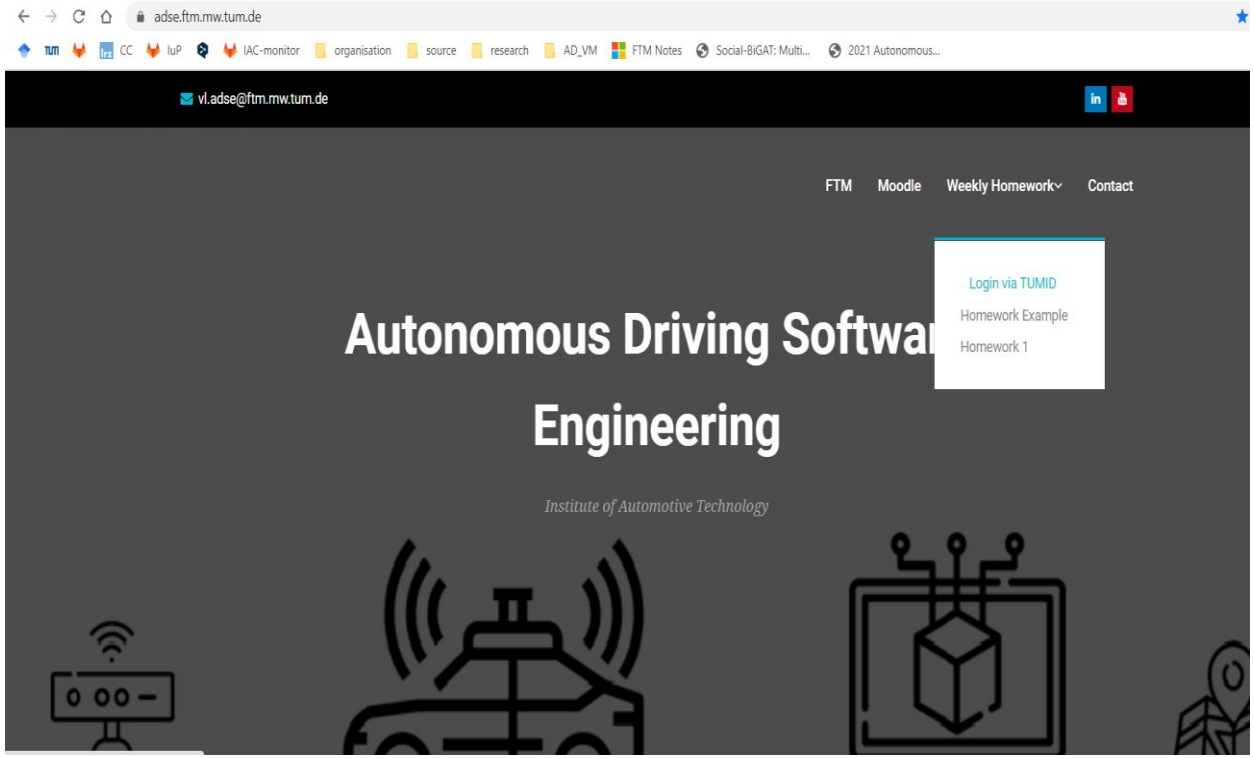
Dipl. Ing. Nico Uhlemann & Simon Sagmeister, M. Sc.

Agenda

1. Introduction
2. Homework
3. Programming Environment
4. System Setup
5. **Working on your weekly homework**

Working on your weekly homework – Code FREAK

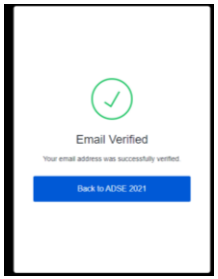
- The first step is to access your programming tasks submission website under: https://adse.ftm.ed.tum.de/



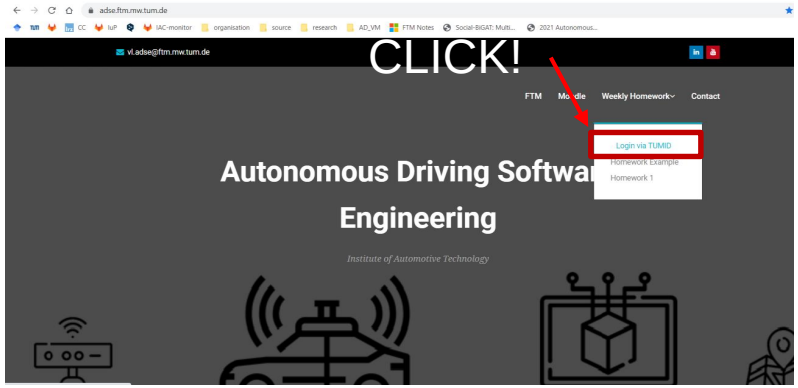
Working on your weekly homework – Code FREAK

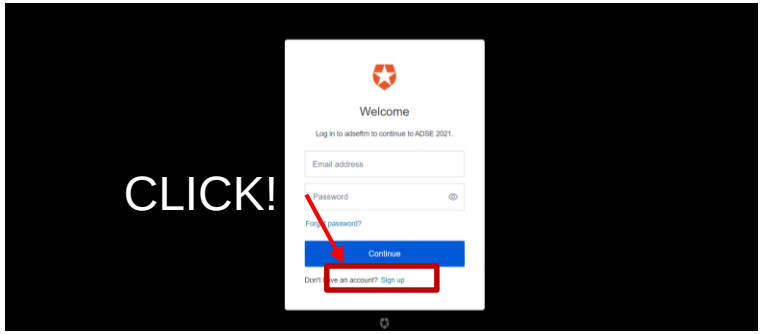
At the beginning of the semester:

- An initial Sign Up with your <TUMID>@mytum.de email-address is necessary.
- So click on *Weekly Homework* → *Login via TUMID* → *Redirecting to Student's Login* → *Sign up* (Don't forget to Verify you email)



PLEASE USE ONLY YOUR TUMID@mytum.de eMail-address, otherwise your results will not be evaluated





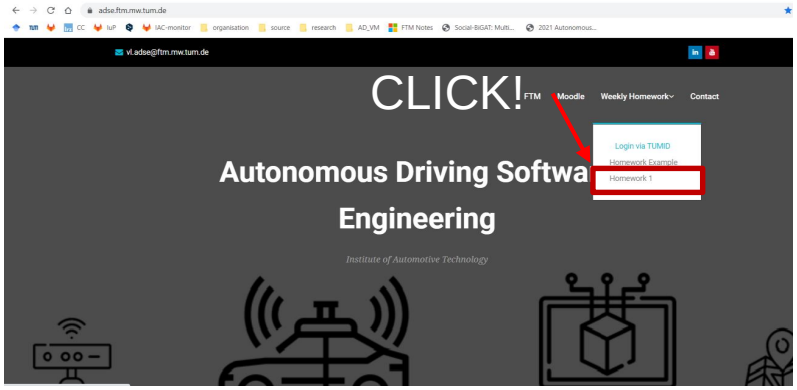
Working on your weekly homework – Code FREAK

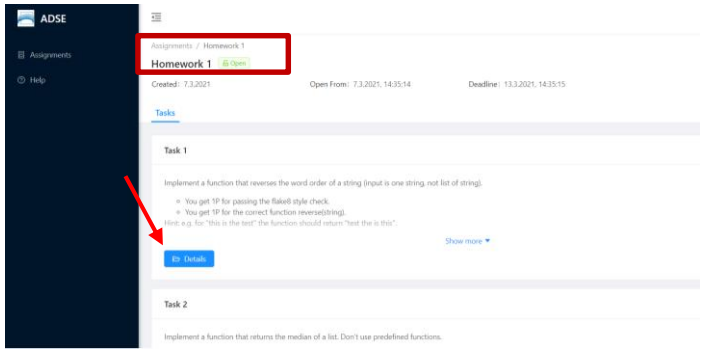
To start your weekly homework:

1. Login with <TUMID>@mytum.de to get to https://adse.ftm.ed.tum.de/submissions/. If you click on *Assignments*, the homework won't be visible yet.

2. To subscribe to the current homework, you have to go back to https://adse.ftm.ed.tum.de/ and click on *Weekly Homework → Homework <XYZ>.* Now you should get redirect to Assignments / Homework <XYZ>.

3. To add the homework to your assignments, just do *Task 1 → Details → Start working on this task1.* Now, the homework is stored in your assignment pool





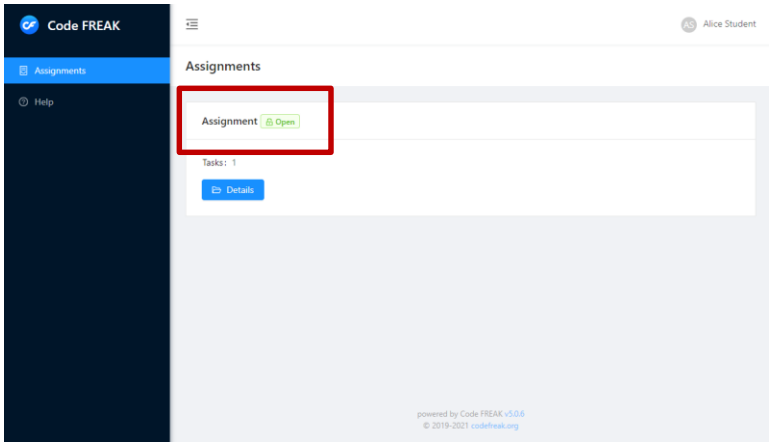
Working on your weekly homework – Code FREAK

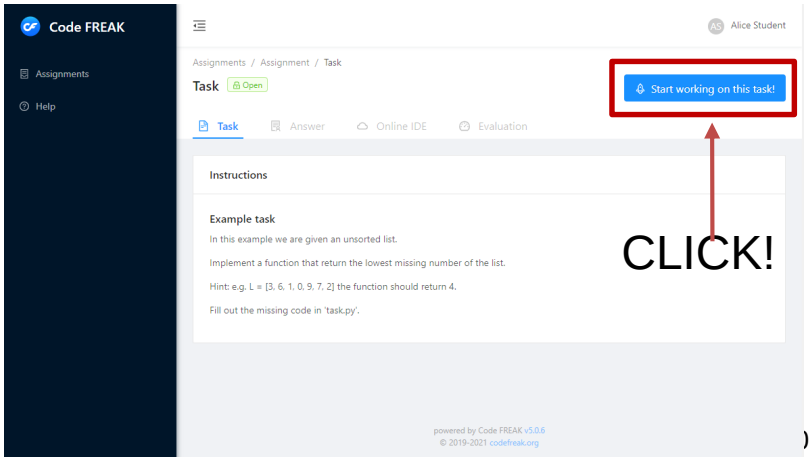
To start your weekly homework:

1. Login with <TUMID>@mytum.de to get to https://adse.ftm.ed.tum.de/submissions/. If you click on *Assignments*, the homework won't be visible yet.

2. To subscribe to the current homework, you have to go back to https://adse.ftm.ed.tum.de/ and click on *Weekly Homework → Homework <XYZ>.* Now you should get redirect to Assignments / Homework <XYZ>.

3. To add the homework to your assignments, just do *Task 1 → Details → Start working on this task1.* Now, the homework is stored in your assignment pool



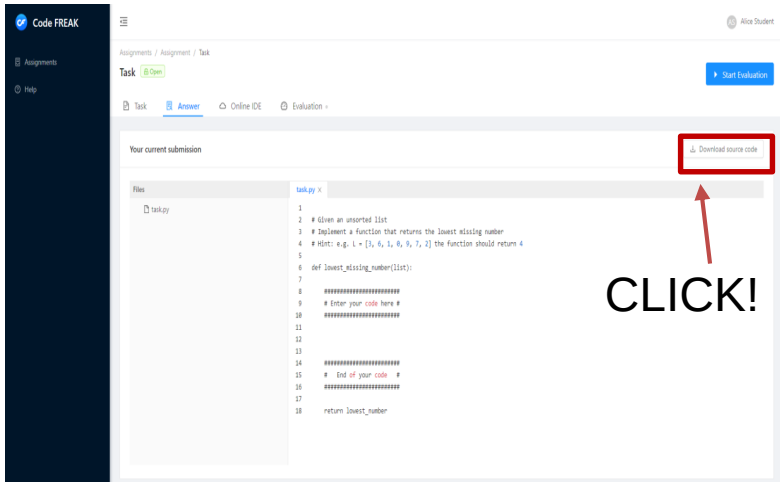


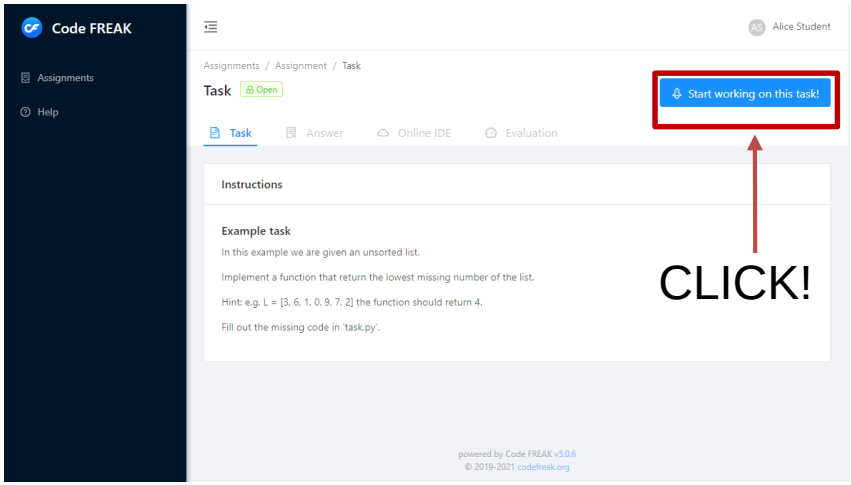
Working on your weekly homework – Code FREAK

To work on your weekly homework:

1. To start coding, choose click on *Task <XYZ>* → *Details* → *Start working on this task!*

2. Download the source code of the task to work on it locally on your computer in PyCharm. Make sure you have a neat folder structure for the exercises. (see Slide 1 – 25). Open your PyCharm Project

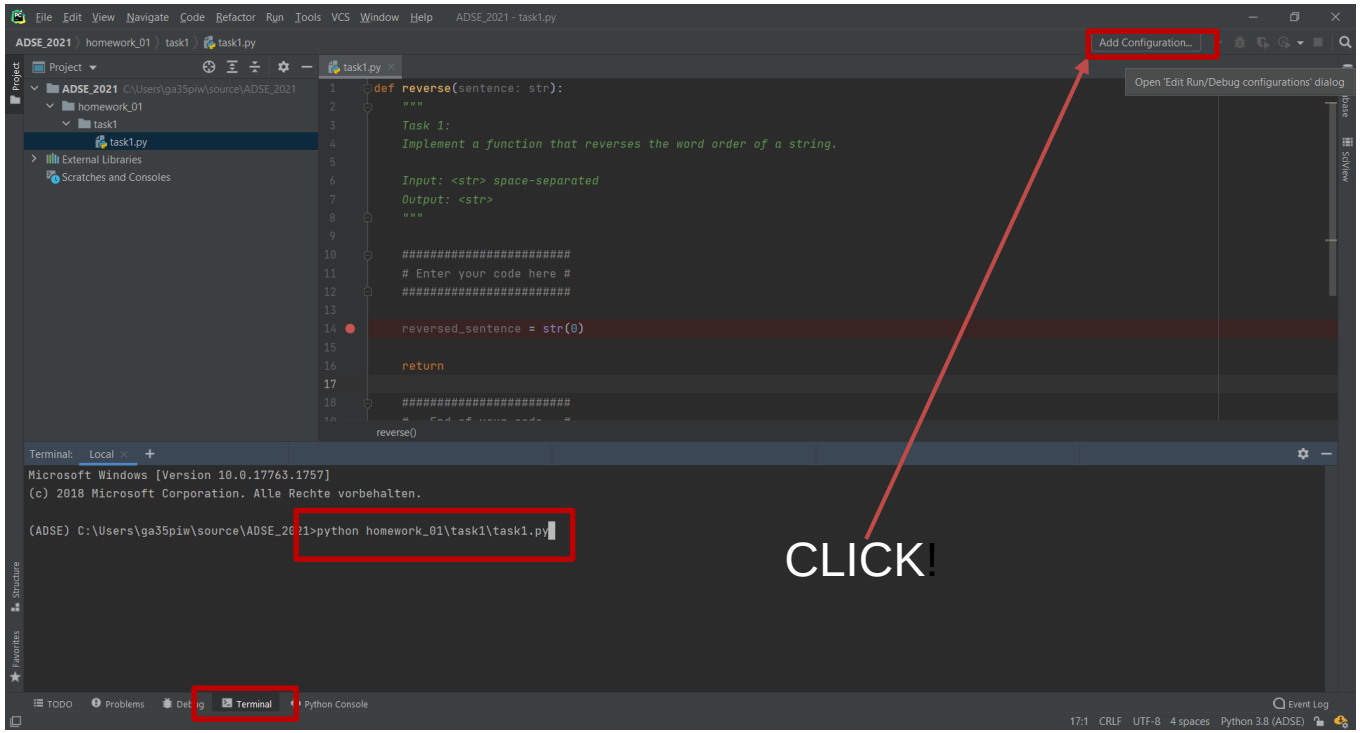




Working on your weekly homework – Code FREAK

To solve the task in PyCharm

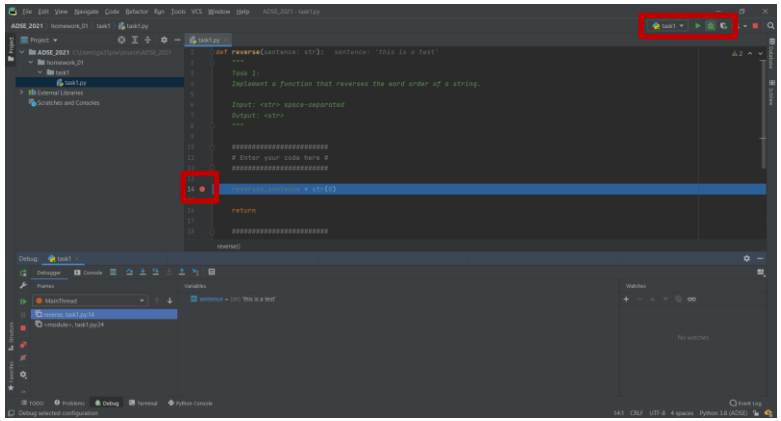
1. Either run the script via terminal command and / or create a configuration (Recommended for **debugging**)

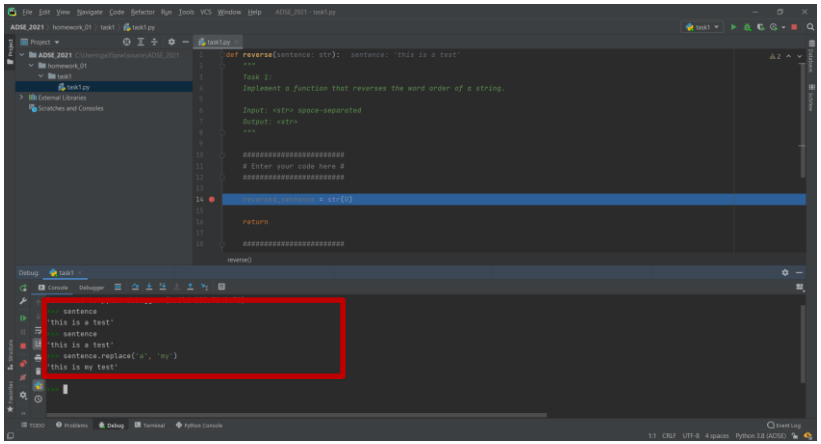


Working on your weekly homework – Code FREAK

To solve the task in PyCharm

2. Set a debug point, where the script should stop. Then run your script in the -mode and try things around until you think your solution is correct.

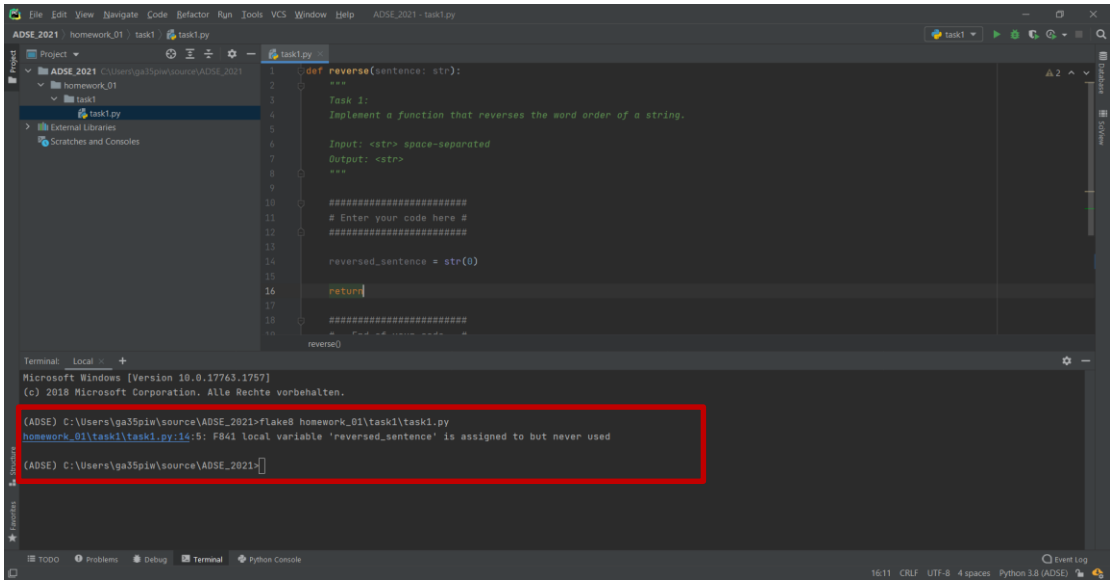




Working on your weekly homework – Code FREAK

To solve the task in PyCharm

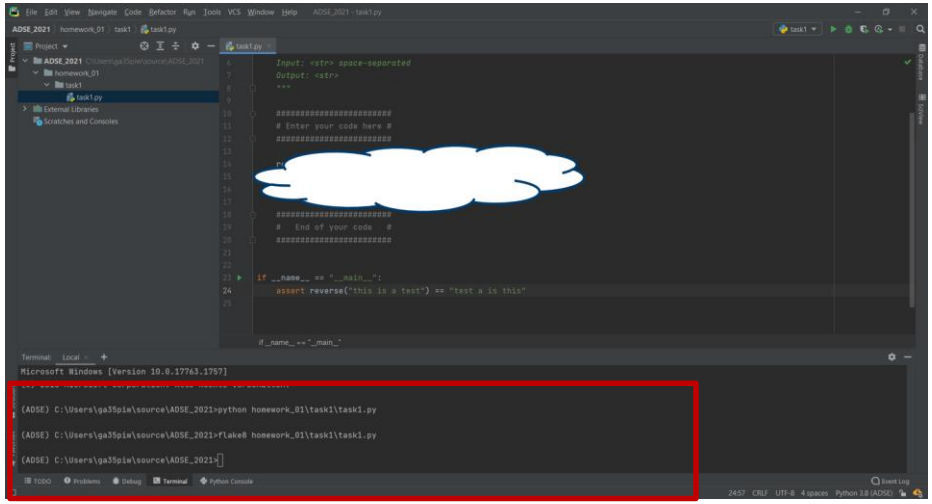
3. To check your Code-Style, run the following commands:
   `cd homework_<XYZ>\task<XYZ>`
   `flake8 task<XYZ>.py`
   → You will also get a point for pretty code, so do the Code-Style Checking

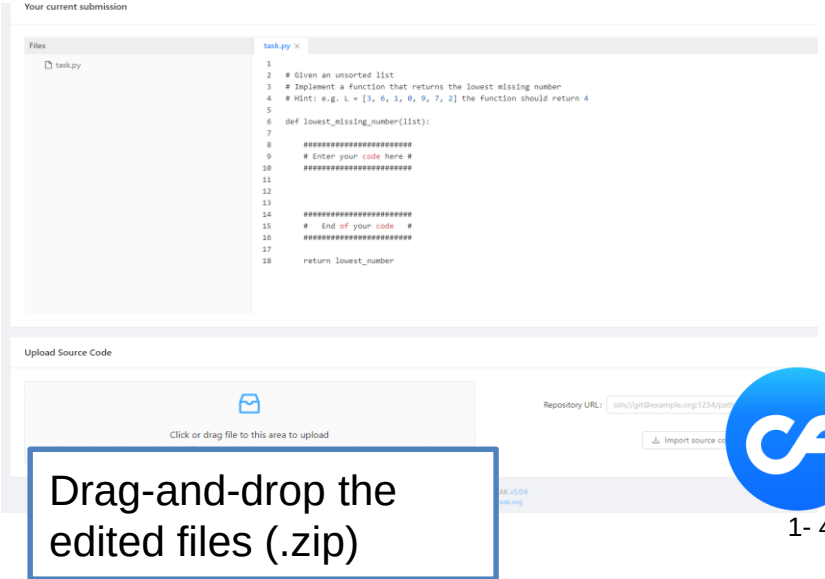


Working on your weekly homework – Code FREAK

To solve the task in PyCharm

4. When you are ready to go, you shouldn't get any more errors. If so, compress your edited files to a .zip and upload it back to Code FREAK (Just where you downloaded them.
   .. But, be aware: The final evaluation will be done in Code FREAK



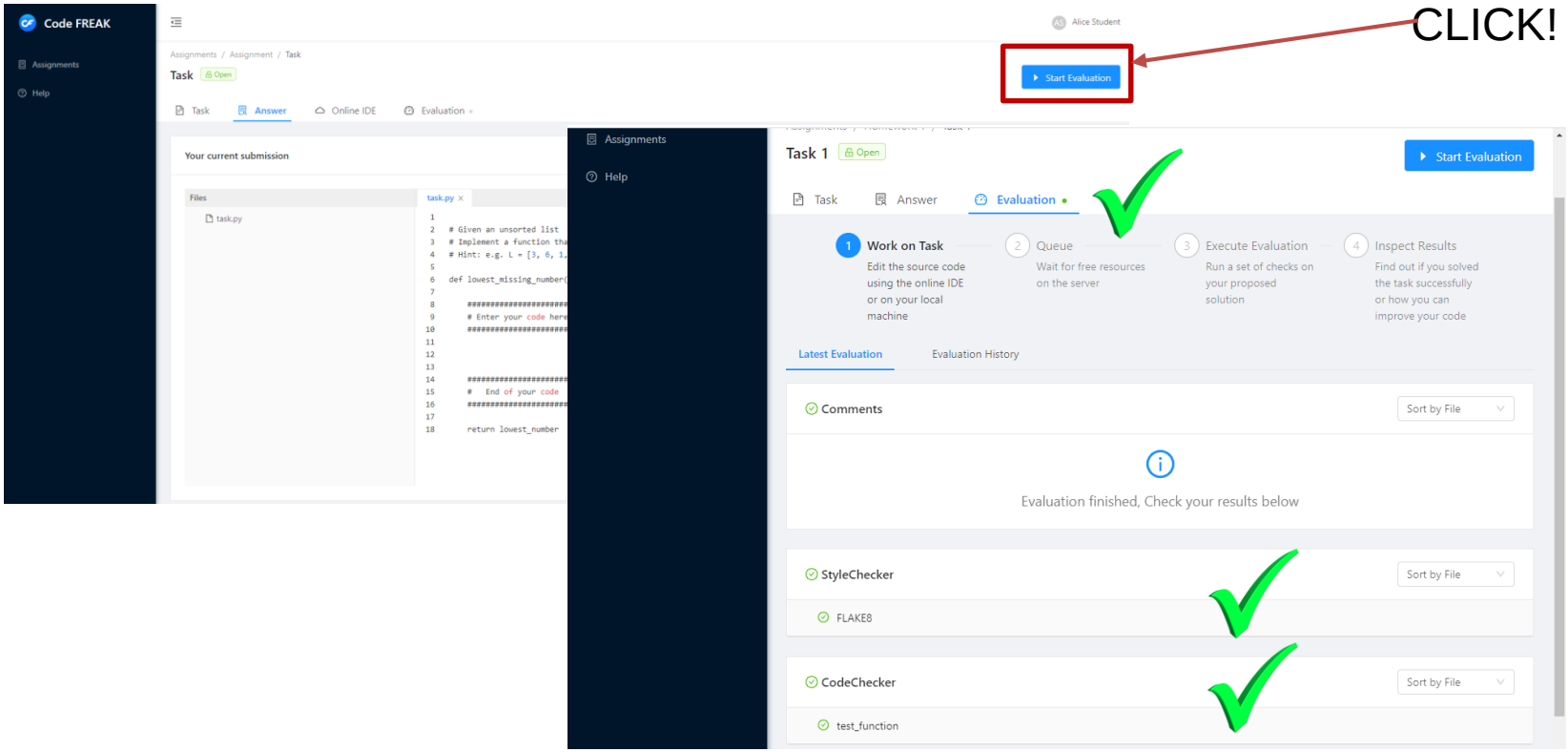


Drag-and-drop the edited files (.zip)

Working on your weekly homework – Code FREAK

Submission

Evaluate the task, check your results and (if everything is correct) go on with the next task



Working on your weekly homework – Code FREAK

Reset your code

In case, you totally messed your code up, you can reset a task completely.

WARNING: All your previous upload are lost after the reset!