

Python

automation, web scraping, image
processing

1st date

- Web scraping, data cleansing and data preparation
 - Identifying and selecting relevant elements in HTML source code
 - Web scraping (e.g. with BeautifulSoup)
 - Cleansing and preparing data

2nd date

- Analyzing data, saving data in other file formats, presenting data
 - Basic data analysis techniques with Python
 - Saving data in different file formats
 - Creating diagrams and visualizations
 - Creating interactive dashboards (e.g. with Dash or Flask)

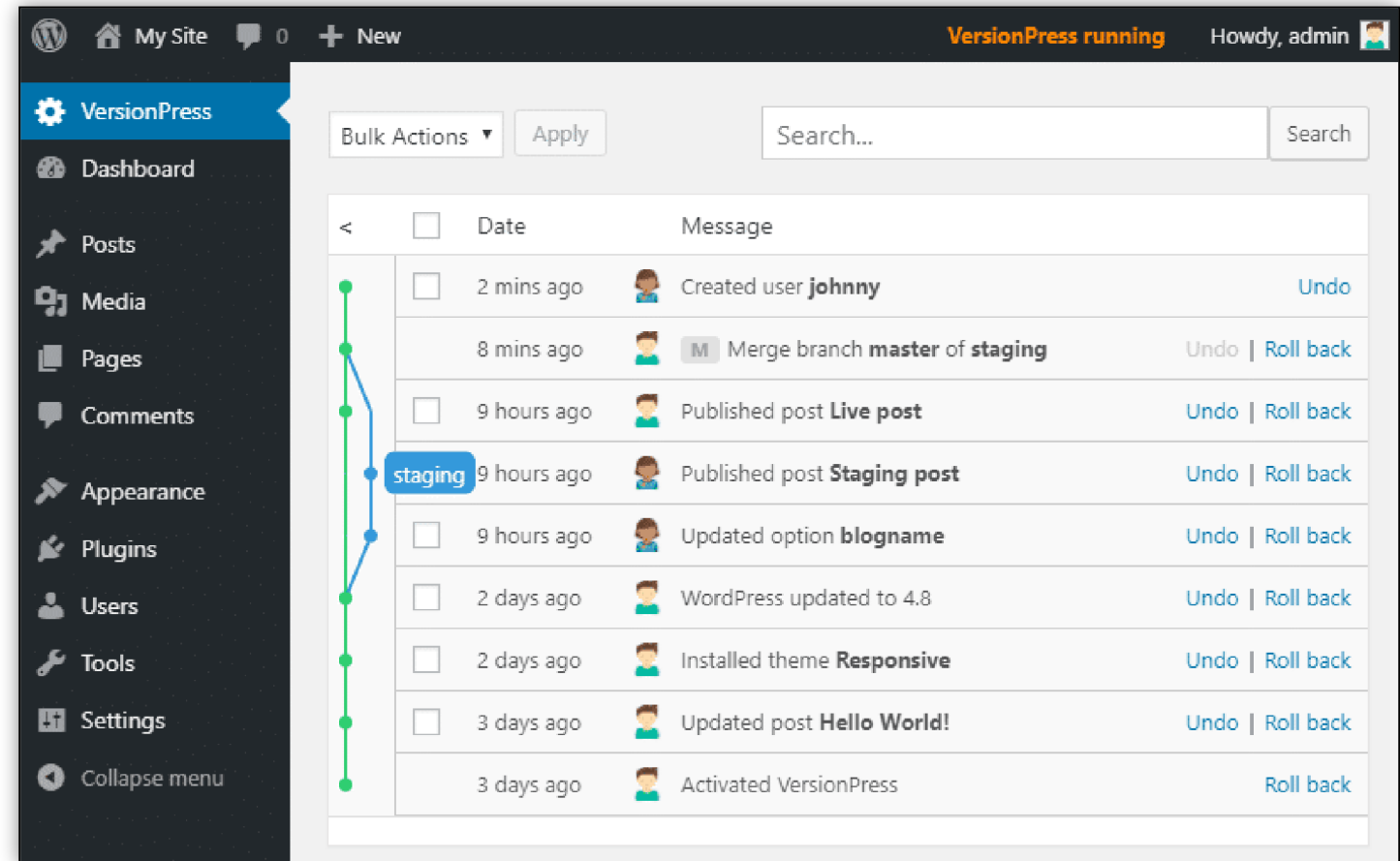
3rd date

- Image processing and automation with Python
 - Introduction to image processing with Python (e.g. with OpenCV)
 - Simple manipulation of images
 - Using Python to automate tasks and workflows

Preparation

- To be successful, a little preparation is required
 - Python installed
 - A Python IDE (Pycharm Community/Edu or VSCode)
 - *Git installed (<https://www.git-scm.com/downloads>)**
 - Permissions to install packages with PIP

Git



Git [git] is a free software for distributed version management of files, which was initiated by Linus Torvalds.

Why Git

In addition to Git, exchange via ZIP file is also possible (and is also offered).

However, this has the disadvantage that a new package must be downloaded for each course adaptation / more content + data must be synchronized.

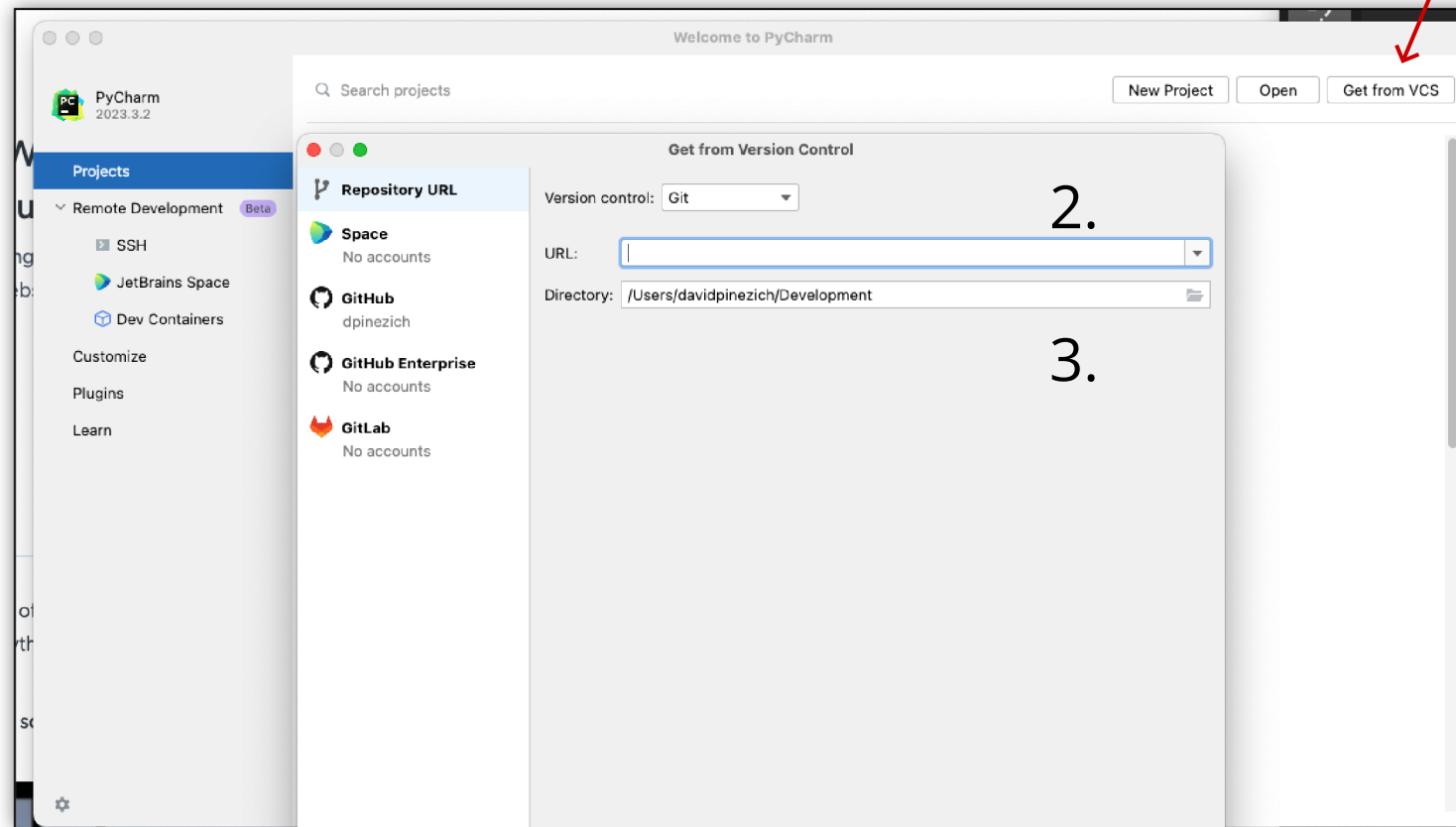
How do we use Git?

Although Git is about collaborative work, we use it **unilaterally**.

I provide the materials, you can simply obtain and update them via **git pull**.

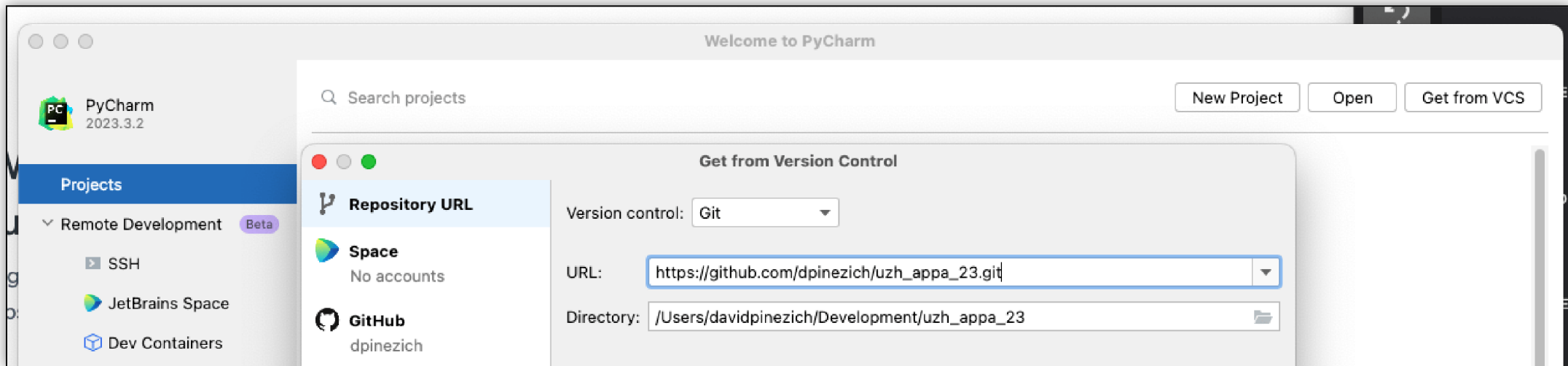
Git uses a delta algorithm, which means it's important that we don't work in the same folder - but more on that later.

Git in Pycharm



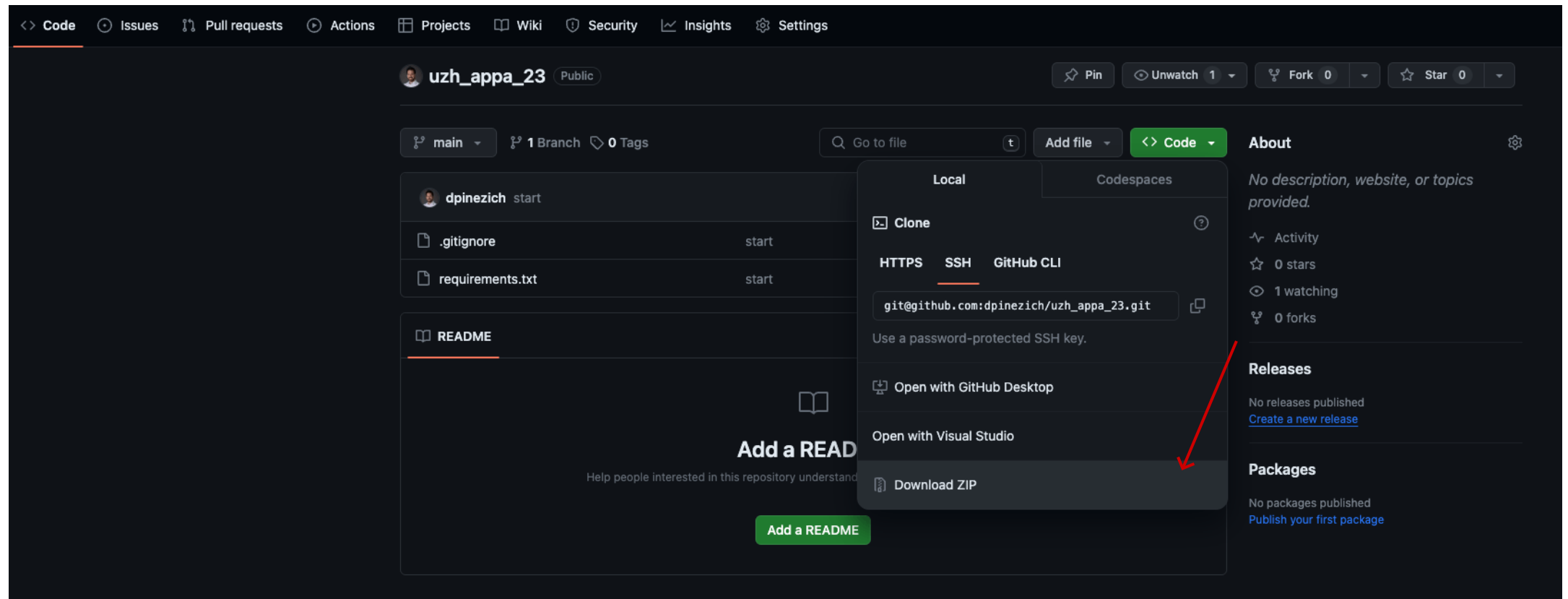
Git in Pycharm

https://github.com/dpinezich/uzh_appa_24_en.git



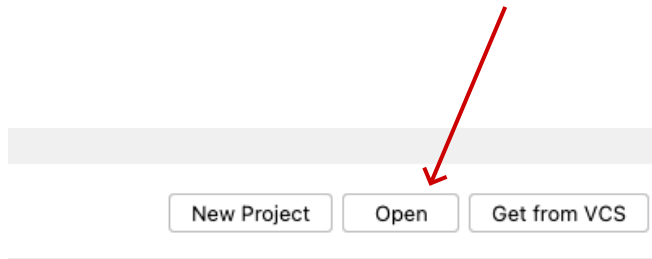
ZIP

https://github.com/dpinezich/uzh_appa_24_en

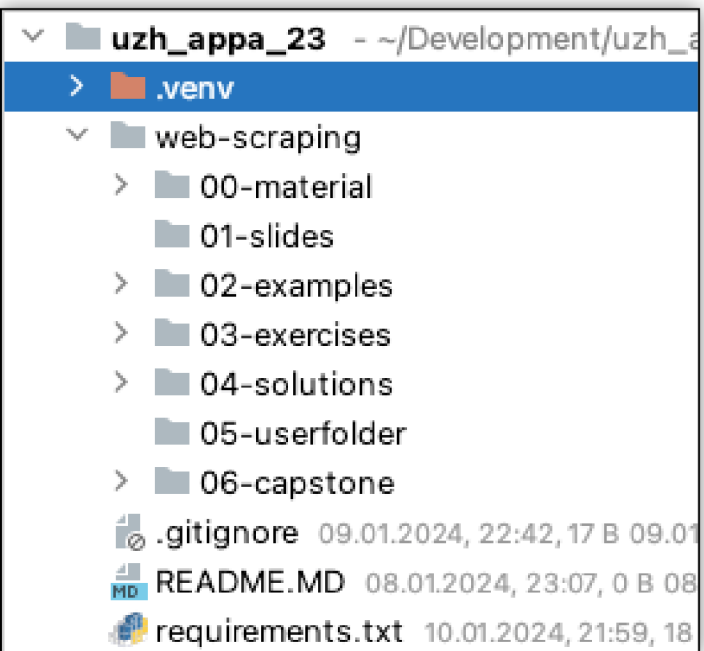


ZIP

Unzip on the desktop (please not in downloads)



Typical structure

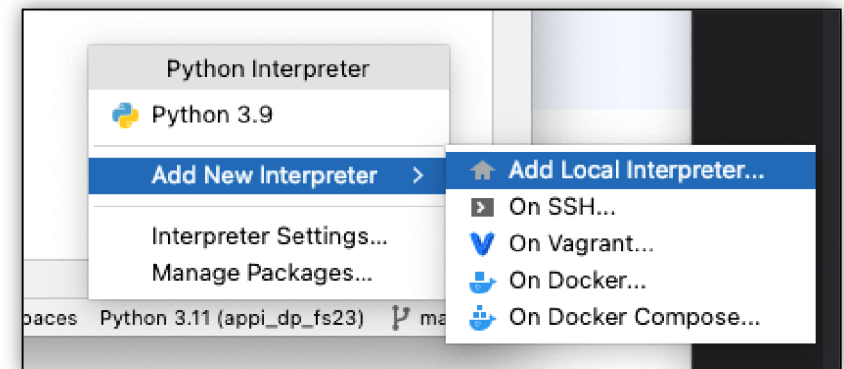
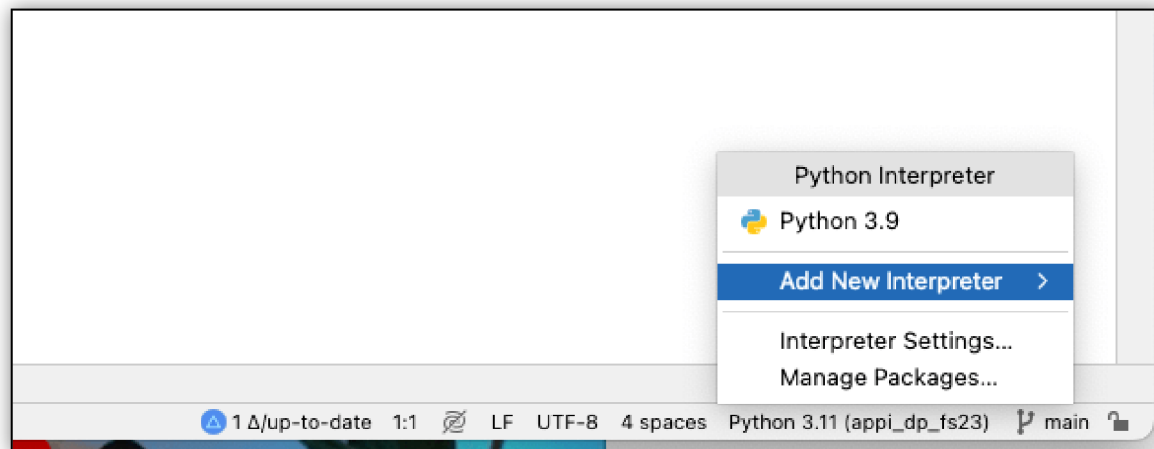


- 00-material: is used for additional material, data sets, cheat sheets
- 01-slides: contains all slides shown and additional slides
- 02-examples: contains ready-made examples
- 03-exercises: contains examples that provide a starting point for exercises
- 04-solutions: contains all solutions to the tasks set in the slides also contains solutions to unfinished 03-exercises
- 05-userfolder: YOUR playground to save your solutions, attempts, second opinions, etc. :-)
- 06-capstone: A somewhat more elaborate, real-life final project

Checkpoint

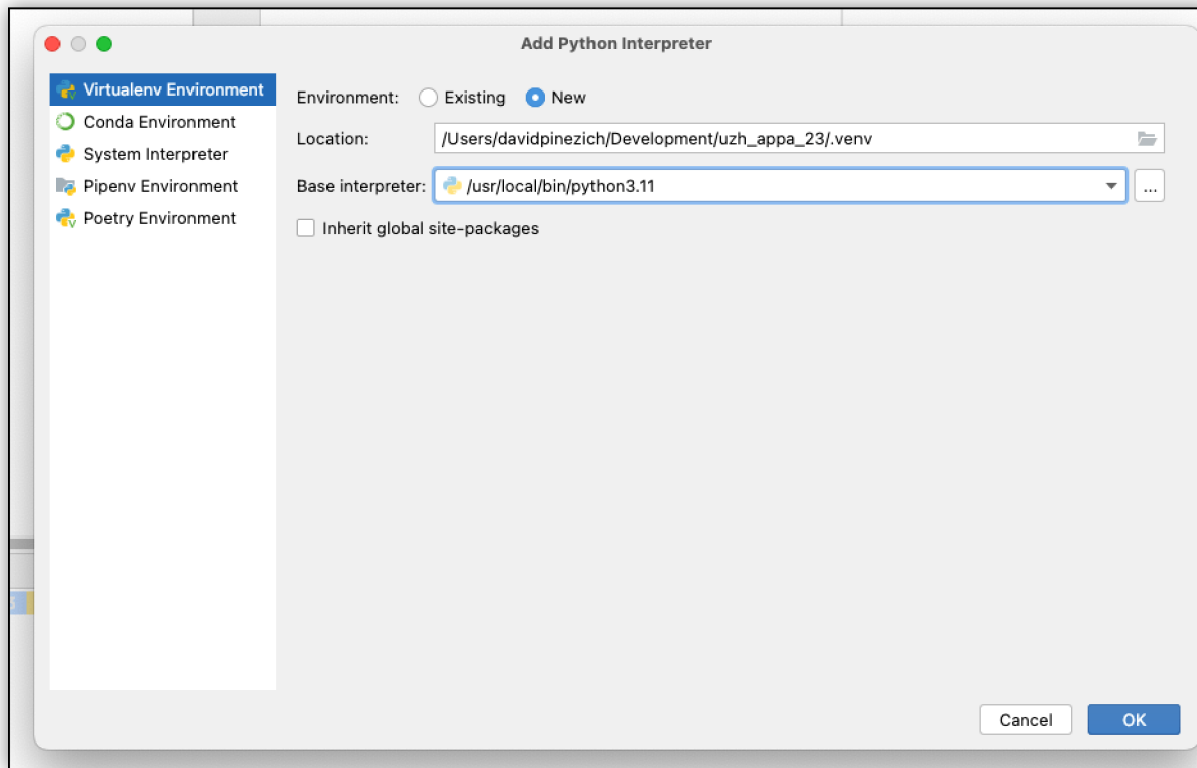
We have now reached the same checkpoint with both Git and ZIP.

Next, we need a virtual environment so that we can install packages without problems.



VENV

Python 3.8 or higher is fine, 3.11/.12+ is recommended.



please do not select inherit global site-packages, we want a "fresh" start.



Install packages

We can install packages in three ways:

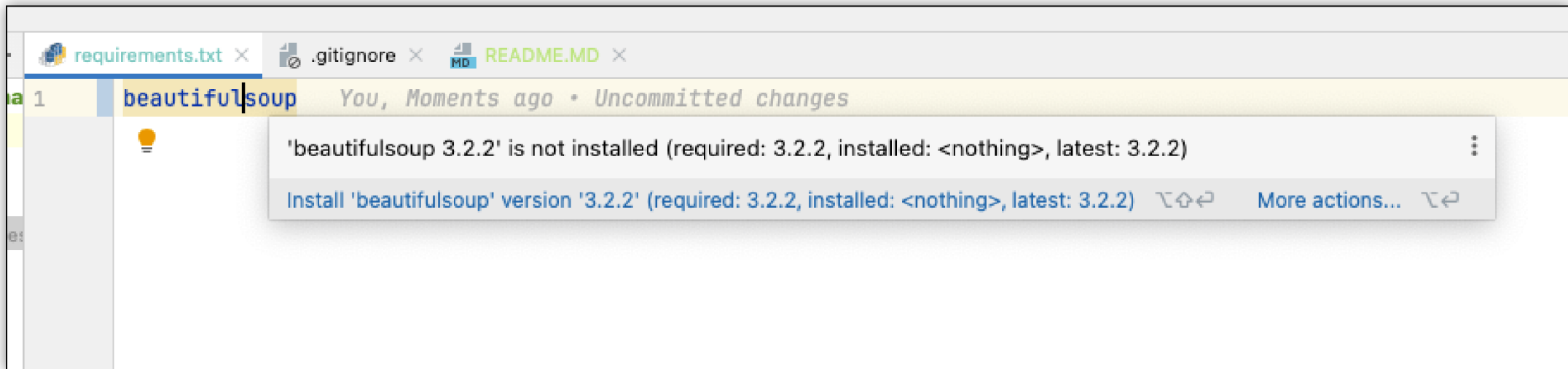
- PyCharm Console
- PyCharm selection
- Requirements.txt

Please restart PyCharm briefly after for re-synchronization.

requirements.txt

A simple way to specify the installation:

Double-click on requirements.txt:



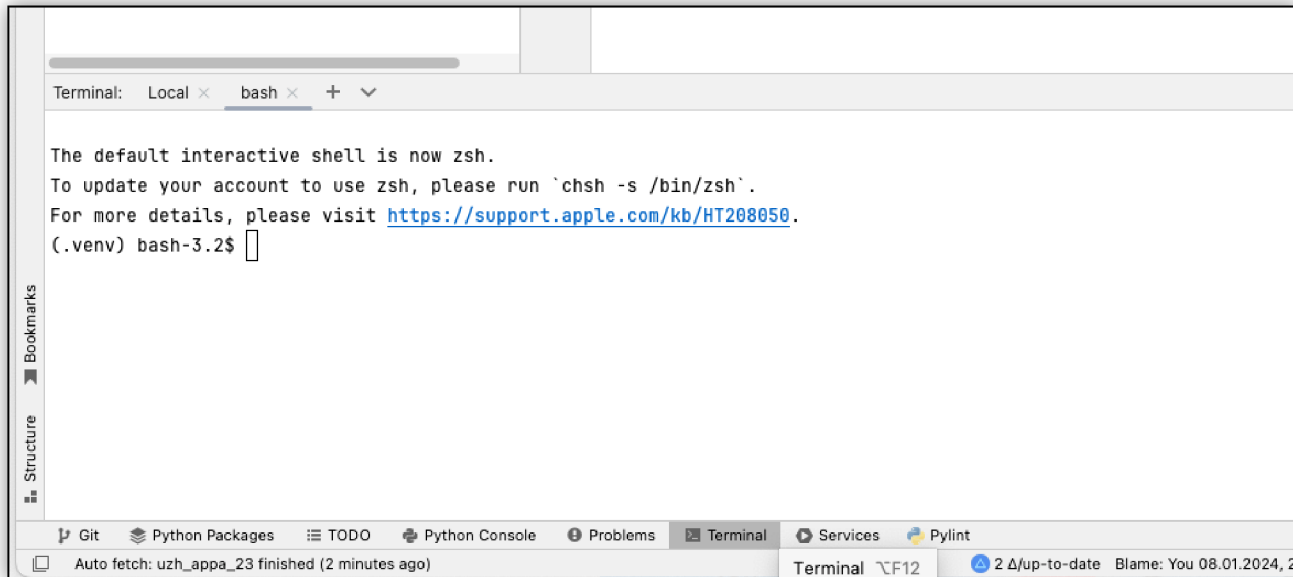
Or



PyCharm Console

A console is supplied with PyCharm (and VSCode).

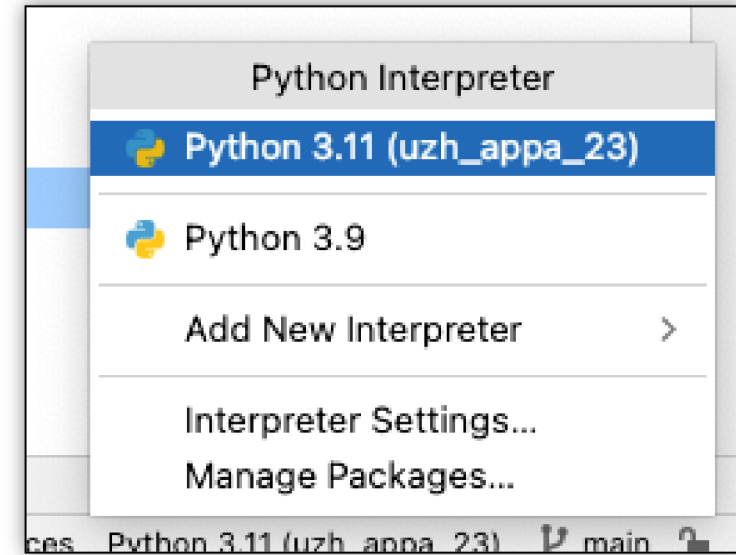
With the restart, our .venv was also recognized (see (.venv)).



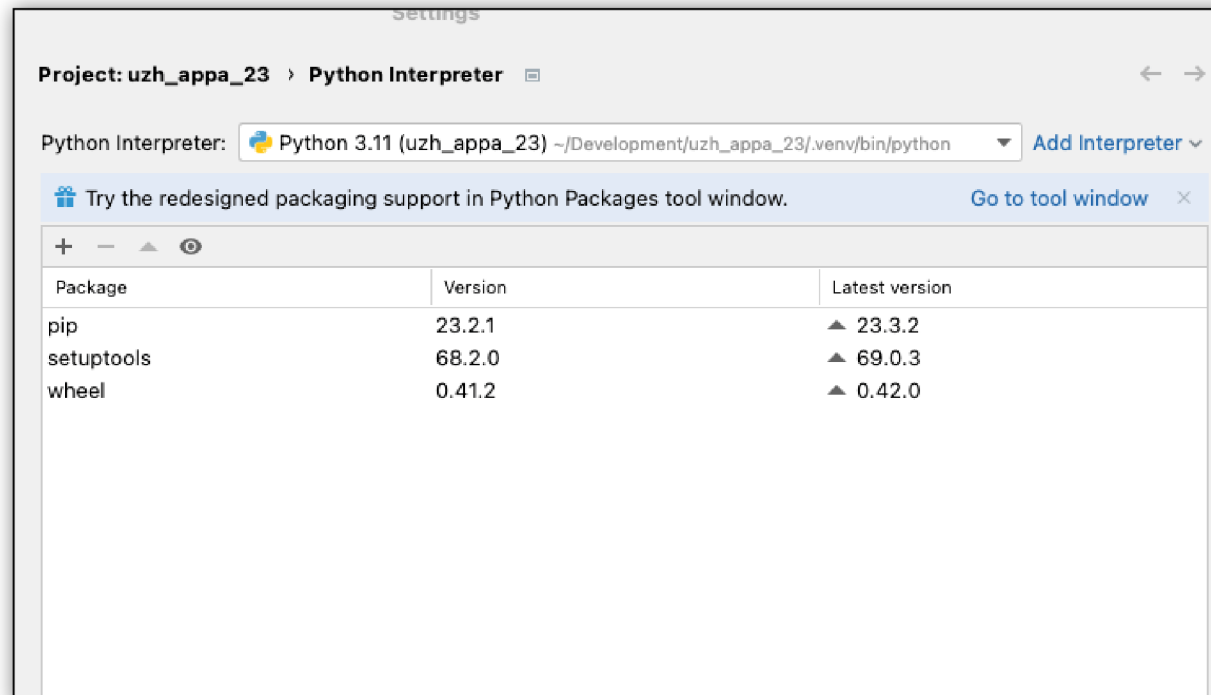
pip install "package"

PyCharm Selection

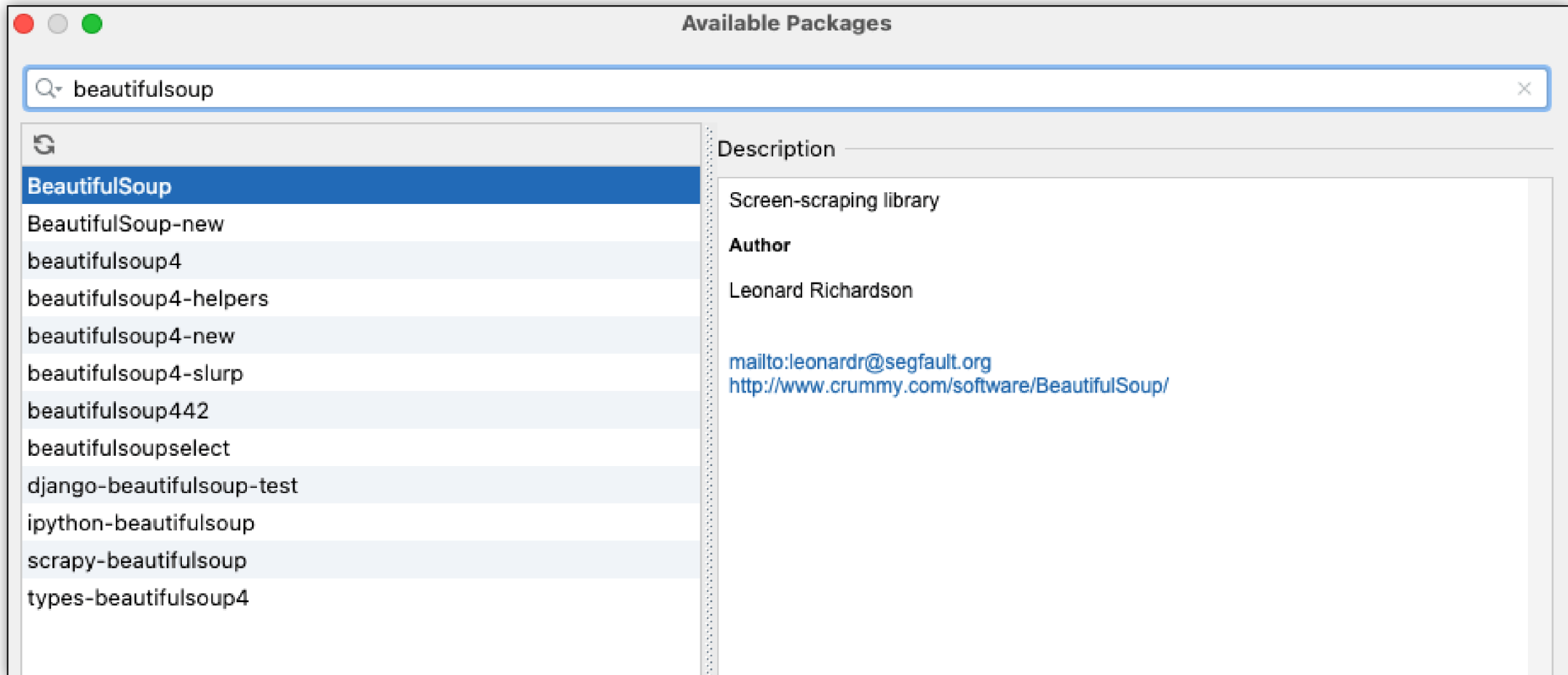
1) Click on Interpreter Settings



2) Click "plus"

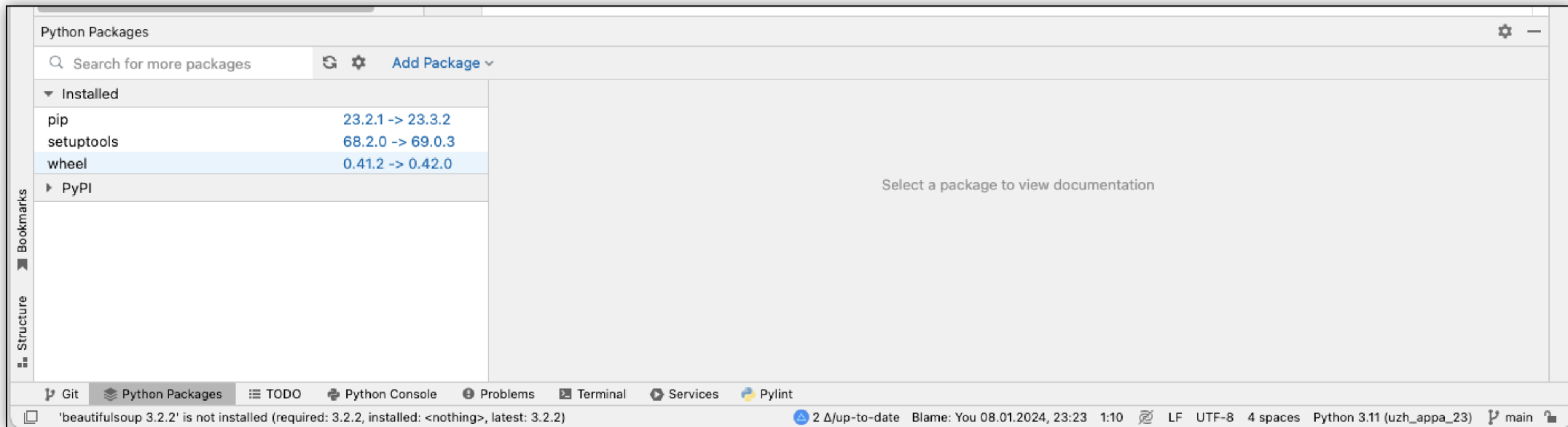


PyCharm Selection



PyCharm Selection (Modern)

Tool Window

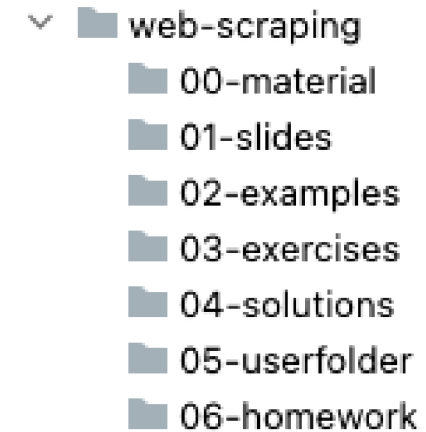


Exercise - Install package

1) Create a new file in 05-userfolder
`test_sample.py`

2) Content:

```
1 # content of test_sample.py
2 def inc(x):
3     return x + 1
4
5
6 def test_answer():
7     assert inc(3) == 5
```



web-scraping

- 00-material
- 01-slides
- 02-examples
- 03-exercises
- 04-solutions
- 05-userfolder
- 06-homework

Übung - Exercise - Install package

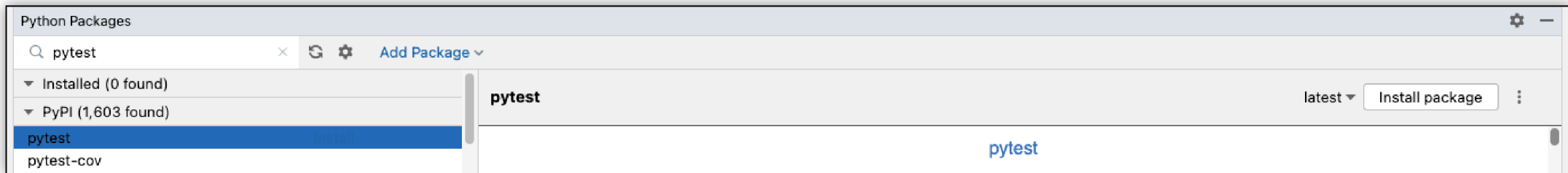
3) Install pytest (no matter which way)



4)

Übung - Exercise - Install package

3) Install pytest (no matter which way)

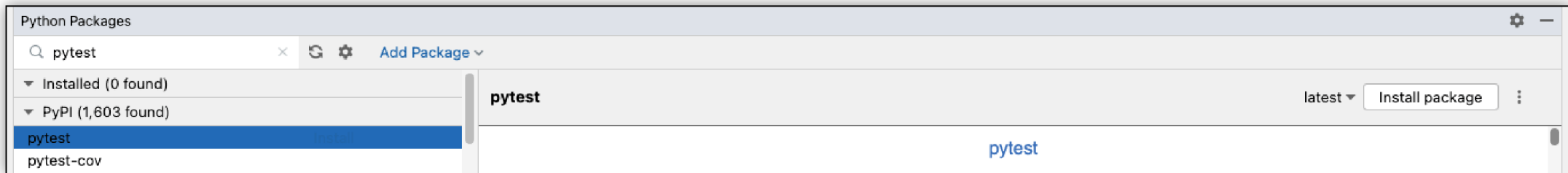


4)

```
1  # content of test_sample.py
2  1 usage new *
3  def inc(x):
4      return x + 1
5
6  new *
7  def test_answer():
8      assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```


Übung - Exercise - Install package

3) Install pytest (no matter which way)



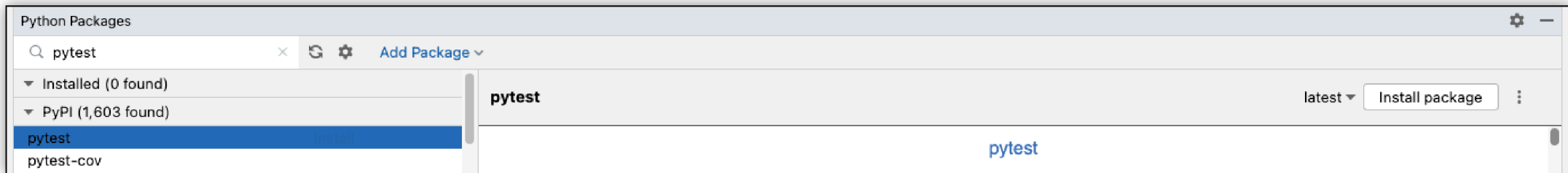
4)

```
1  # content of test_sample.py
2  1 usage new *
3  def inc(x):
4      return x + 1
5
6  new *
7  def test_answer():
8      assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```



Übung - Exercise - Install package

3) Install pytest (no matter which way)



4)

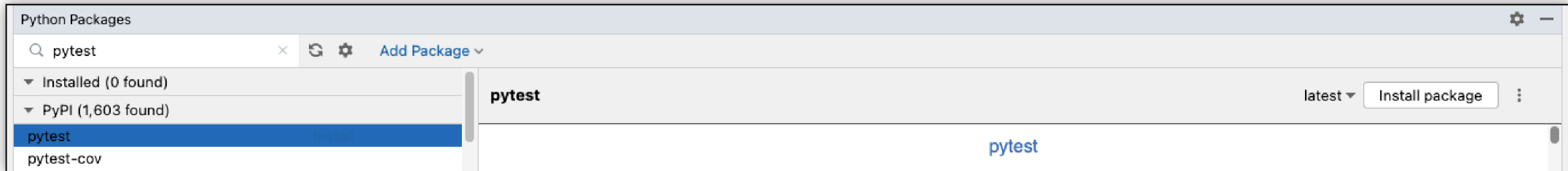
```
1  # content of test_sample.py
2  1 usage new *
3  def inc(x):
4      return x + 1
5
6  new *
7  def test_answer():
8      assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```



```
1  # content of test_sample.py
2  1 usage new *
3  def inc(x):
4      return x + 1
5
6  new *
7  def test_answer():
8      assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```

Übung - Exercise - Install package

3) Install pytest (no matter which way)



4)

```
1 # content of test_sample.py
2 1 usage new *
3 def inc(x):
4     return x + 1
5
6 new *
7 def test_answer():
8     assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```



```
1 # content of test_sample.py
2 1 usage new *
3 def inc(x):
4     return x + 1
5
6 new *
7 def test_answer():
8     assert inc(3) == 5
9
10 You, 3 minutes ago • Uncommitted changes
```

Ready to get started?

