Lab0 Getting started with NP

Introduction to Human language Technology

Piek Vossen

Overview

- AI and coding
- Preparations
- Jupyter notebooks
- Get the lab materials (notebooks, data) from Github
- Create a virtual environment with the right version of Python and all required packages
- How to use a terminal
- A word on Text editors

Should I use AI for coding?

ChatGPT's answer

That depends on **how** you want to use AI for coding and what your goals are. Here's a breakdown to help you decide:

✅ **When AI can be very helpful for coding:**

- **Boilerplate and repetitive code**: AI can generate standard patterns quickly (e.g., setting up APIs, writing test scaffolding, configuring frameworks).
- **Learning & exploration**: If you're learning a new language, framework, or algorithm, AI can give you examples and explanations.
- **Debugging assistance**: AI can point out bugs, suggest fixes, or explain error messages.
- **Speeding up prototyping**: You can get a working draft quickly and then refine it yourself.
- **Documentation & comments**: AI can summarize or generate documentation.

⚠️ **Things to watch out for:**

- **Code quality**: AI-generated code may work but not follow best practices, be inefficient, or be insecure.
- **Understanding**: Relying too much on AI without understanding the code can slow your growth as a programmer.
- **Privacy/security**: Don't paste sensitive or proprietary code into AI tools unless you're sure it's safe.
- **Maintenance**: If you don't fully understand the generated code, maintaining it later can be painful.

🧠 **Best way to use AI in coding:**

Think of AI as a **coding assistant** rather than a replacement.

- Use it to **speed up** your work, but always **review and understand** the output.
- Treat it like **pair programming** with someone who's fast but not always careful.
- The stronger your own coding skills, the more effectively you'll use AI.

👉 Do you want to use AI mainly to **learn coding faster**, or to **be more productive in your existing projects**? That changes the advice I'd give you.

Should I use AI for coding?

My answer

- Yes, it is a great tool to get started
- But you need to train your brain to think about problems:
  - coding is like making a puzzle (can be fun)
  - coding is like structuring your thoughts
  - writing is like structuring your thoughts and explaining it to yourself and others
- And you cannot trust it, so you need to correct and direct it
  - debugging AI code may take more time than writing your own code
- What is my added value in a society run by AI?
- Always, always, always cite and give credentials to AI for the coding it did: —> *code plagiarism*

Preparations

- Hardware
  - Bring your own laptop (Windows, Mac or Linux are all fine) with rights to install software and data.
  - At least 16GB of memory and 500GB of disk capacity.
  - Linux or MacBooks because these are most compatible with the environment of the staff and other researchers in the field. As a Linux system, Ubuntu is recommended, check if your laptop supports it. You can find lists of compatible laptops online (e.g. here: https://ubuntu.com/certified/laptops)
  - Windows works for most things but it is a little bit more difficult.
- Python
  - If you follow(ed) the Python for Text Analysis course parallel to this course, you will have the basic skills to attend.
  - Otherwise, install Anaconda on your local machine which also installs Python. We work with Python 3.10 or 3.11. You can follow the instructions to install anaconda given here:
    - https://www.anaconda.com/download
  - The installer will also install a graphical interface with various tools among which Jupyter notebooks and Jupyter lab. We will not use this interface but work from a Terminal or Command line.

Jupyter notebooks

https://jupyter.org/

- Easy way of coding and documenting what you do and see through a web browser interface:
  - run your python commands through (small) cells of code, step by step
  - visualise output results easily, e.g. using graphs, tables
- Good for trying out and teaching,
- and, you can use your notebooks with some smal adaptations in Google's Colab and Kaggle environments to run it on their powerful hardware with GPUs
- Not good for bigger projects and complex code: for that you should use advanced IDEs (Integrated Development Environment)
  - Visual studio
  - Pycharm
  - etc…

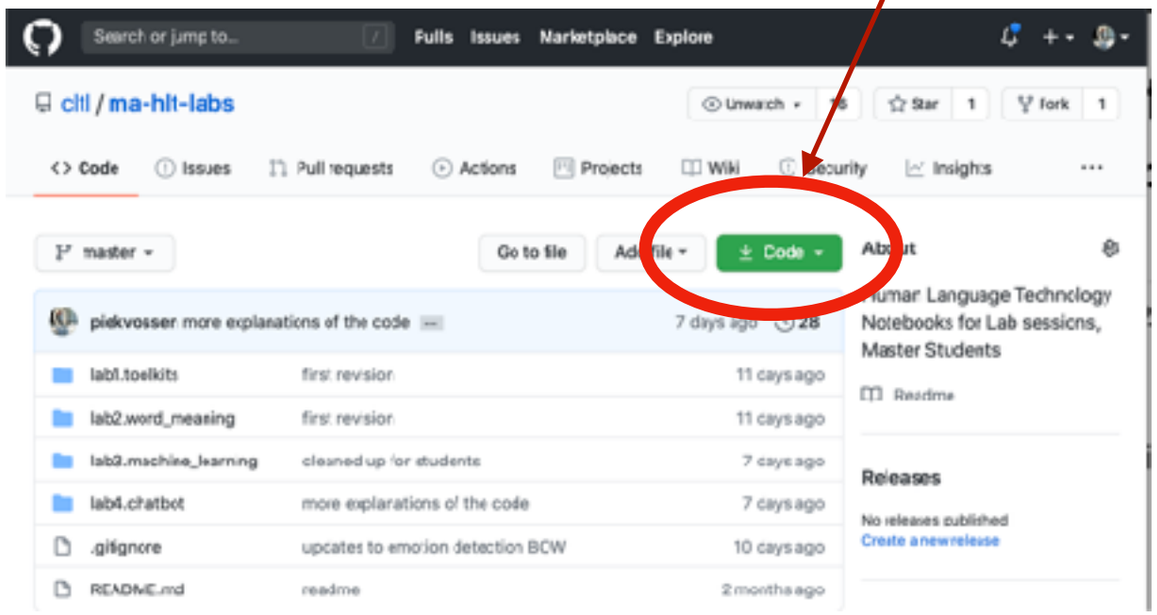
Using Jupyter notebooks

- Cells in a notebook contain code or text (Markdown). If you run a cell, it will either run the code or render the text from the Markdown.
- There are five ways to run a cell:
  - Click the 'play' button next to the 'stop' and 'refresh' button in the toolbar.
  - Alt + Enter runs the current cell and creates a new cell.
  - Ctrl + Enter runs the current cell without creating a new cell.
  - Shift + Enter runs the current cell and moves to the next one.
  - Use the menu and select Kernel -> Restart Kernal and Run All Cells
- The instructions are written in Markdown:
- https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet
- https://www.dataquest.io/blog/jupyter-notebook-tutorial/

Getting the Lab notebooks

Check for updates

Get the download link



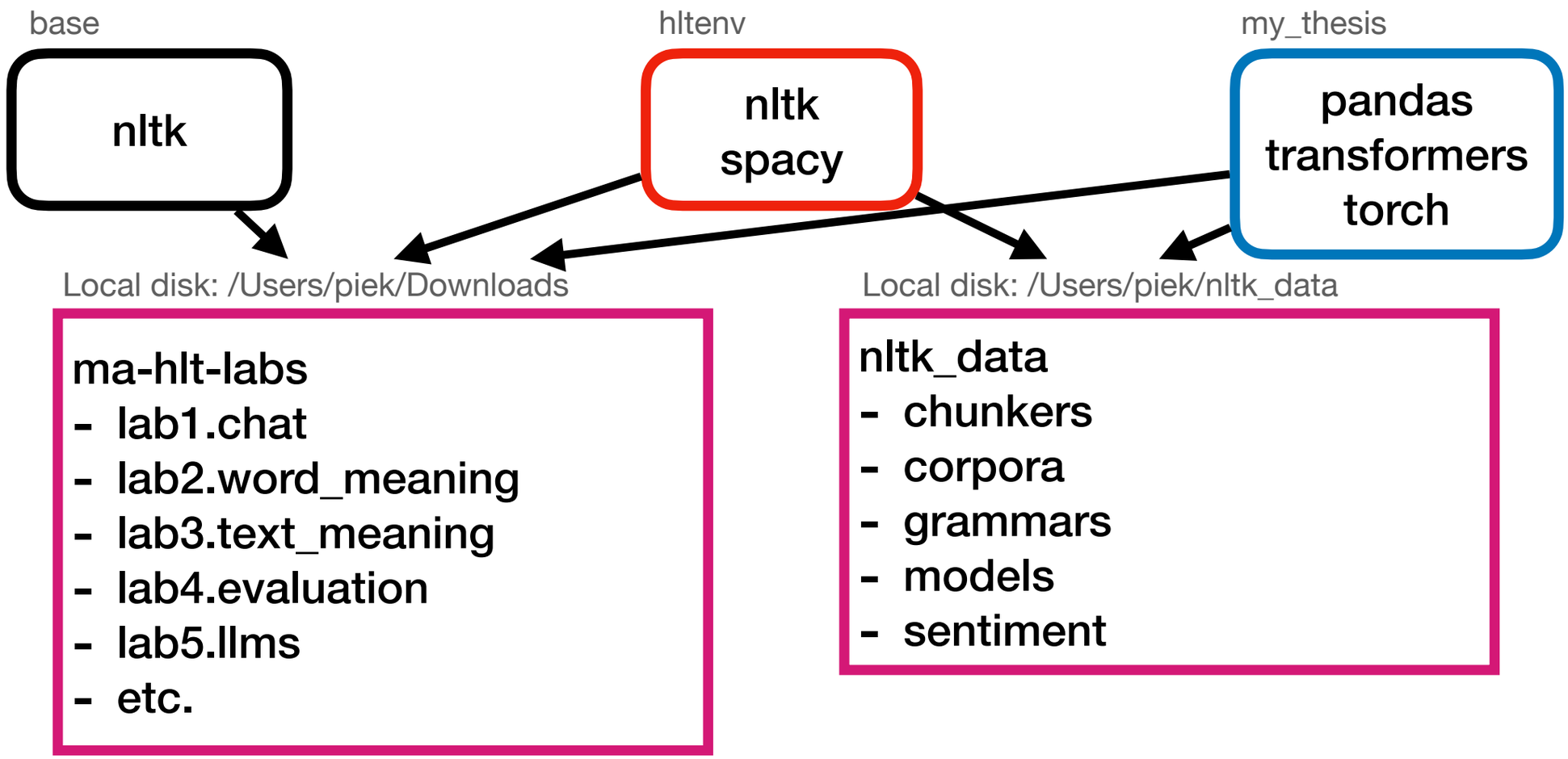
- https://github.com/cltl/ma-hlt-labs
  - Git installed:
    - Clone with ssh:
    - > git clone git@github.com:cltl/ma-hlt-labs.git
  - No local Git installed:
    - Download ZIP file
    - Unpack anywhere

- /Users/piek/Downloads/ma-hlt-labs/
  - lab1.chat
  - lab2.word_meaning
  - lab3.machine_learning
  - Lab4.evaluation
- ma-hlt-labs/
  - lab1.chat
  - lab2.word_meaning
  - lab3.machine_learning
  - Lab4.evaluation

Virtual environments

Installing software within a save silo

- When installing many packages with even more dependencies, conflicts may arise between versions.
- It is wise to create a new virtual environment for each project/course and install and run your code within it.
- https://docs.python.org/3/library/venv.html
- https://packaging.python.org/en/latest/guides/installing-using-pip-and-virtual-environments/



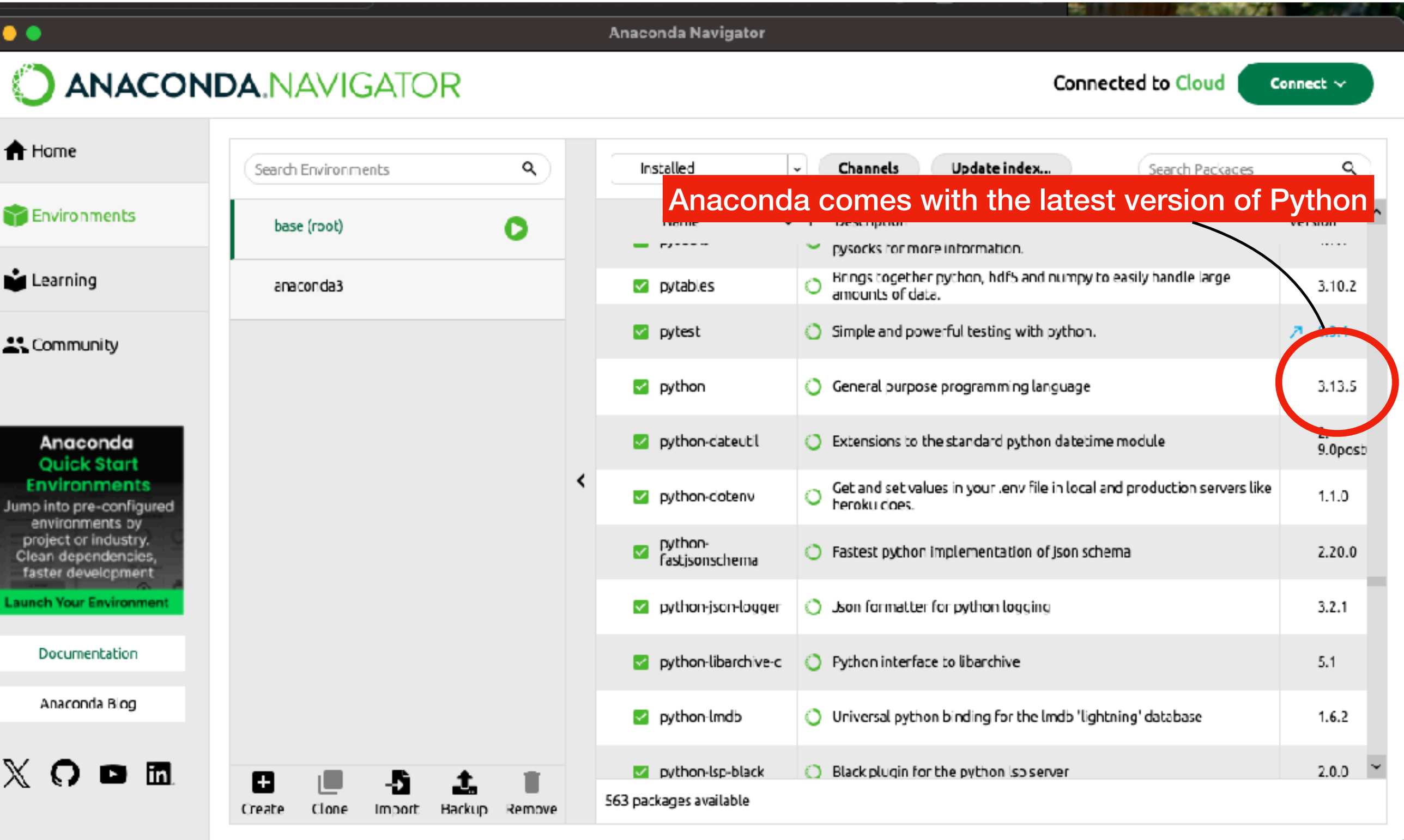
Creating a virtual environment

Install Python 3.11 and install all required packages

- We will create a virtual environment using the Anaconda navigator as it works the same for Windows, Mac and Linux
- Select the Environments tab in the left panel of the Navigator. It shows two environments: base and anaconda3, which both use the latest version of Python (e.g. 3.13)
- We create a new environment called “hltenv” that uses Python 3.11 and in which we will install all packages needed for this course.
- Push the Create [+] button at the bottom of the screen and fill in the name “hltenv” and select Python version **3.11.13**
- Push Create again and wait till all software is installed.
- Open a terminal from the “hltlenv” following the play button: 
- Install all the packages that are in the “requirements.txt” file of ma-hltl-labs

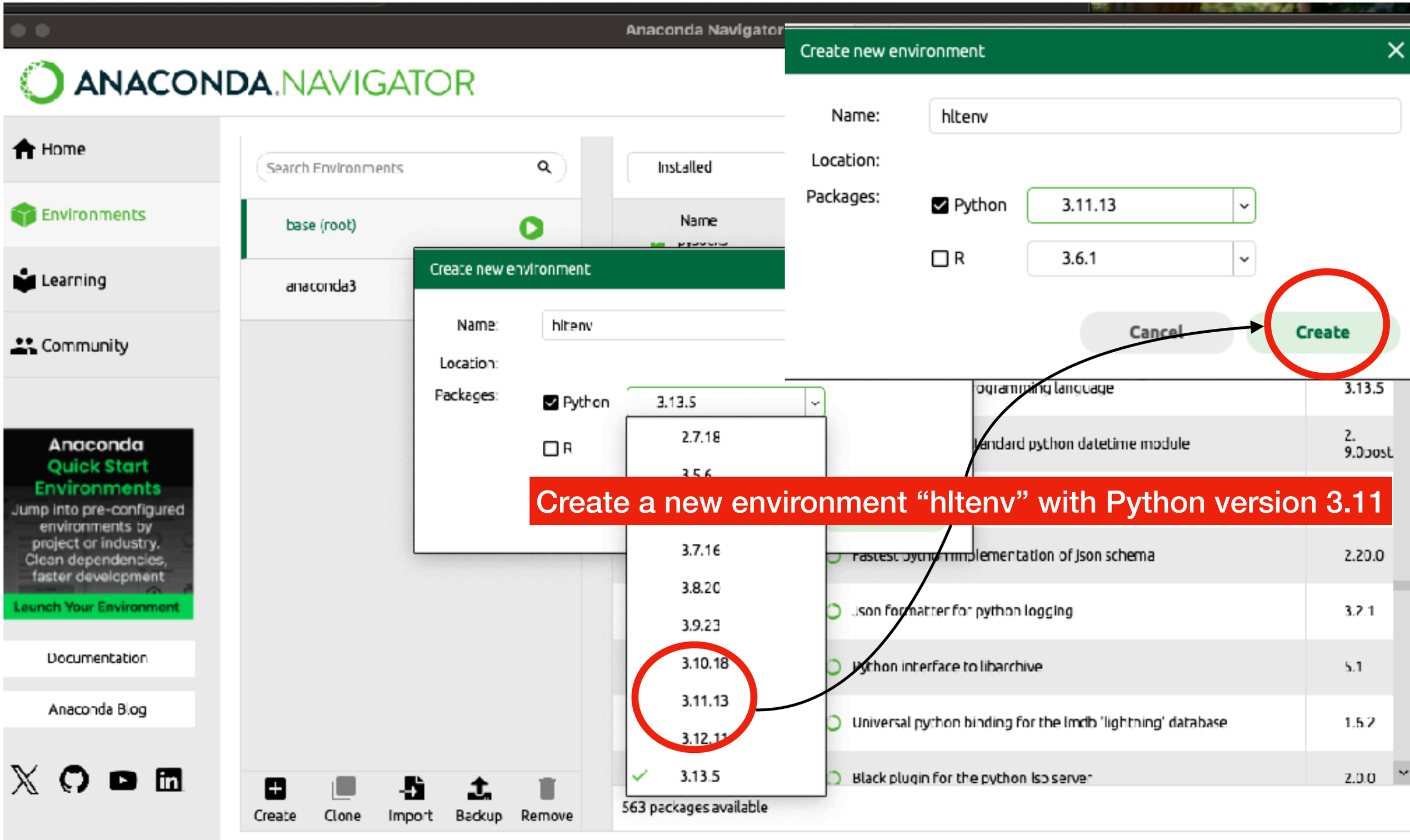
Creating a virtual environment

Python version in anaconda, base has the latest version of Python



Creating a virtual environment

Python version in anaconda



Creating a virtual environment Python 3.11

piek — -zsh — 102×15

```
Last login: Mon Sep  8 11:47:58 on ttys002
. /opt/anaconda3/bin/activate && conda activate /opt/anaconda3/envs/hltenv;
(base) piek@piek-2 ~ % . /opt/anaconda3/bin/activate && conda activate /opt/anaconda3/envs/hltenv;
(hltenv) piek@piek-2 ~ % which python
/opt/anaconda3/envs/hltenv/bin/python
(hltenv) piek@piek-2 ~ % which pip
/opt/anaconda3/envs/hltenv/bin/pip
(hltenv) piek@piek-2 ~ % pip list
Package    Version
---------- -------
pip        25.2
setuptools 78.1.1
wheel      0.45.1
(hltenv) piek@piek-2 ~ %
```

- The prompt is prefixed with (hltenv)
- “which python” and “which pip” show a path to the hot folder in the vents of anaconda
- pip list shows no additional packages installed

ANACONDA.NAVIGA

Home

Environments

Learning

Community

Anaconda Quick Start Environments

Jump into pre-configured environments by project or industry. Clean dependencies, faster development

Launch Your Environment

Documentation

Anaconda Blog

Search Environments

base (root)

anaconda3

hltenv

Open Terminal

Open with Python

Open with IPython

Open with Jupyter Notebook

Create Clone Import Backup Remove

libcxx	Llvm c++ standard library	19.1.7
	A portable foreign function interface library	3.4.4
	Library for text-based user interfaces	6.5
	Openssl is an open-source implementation of the ssl and tls protocols	3.0.17
pip	Pypa recommended tool for installing python packages	25.2
python	General purpose programming language	3.11.13
readline	Library for editing command lines as they are typed in	8.3
setuptools	Download, build, install, upgrade, and uninstall python packages	78.1.1
sqlite	Implements a self-contained, zero-configuration, sql database engine	3.50.2
tk	A dynamic programming language with gui support. bundles tcl and	8.6.15

17 packages available

Installing packages in “htlenv”

- Navigate to the location where you downloaded the ma-htl-labs,
- The “requirements.txt” has a list of packages with specific versions that you need to install for the course
- Upgrade pip if necessary:
  - (**hltenv**)>python -m pip install —upgrade pip
- Install all packages from requirements.txt:
  - (**hltenv**)>pip install -r requirements.txt

ma-hlt-labs — -zsh — 123×18

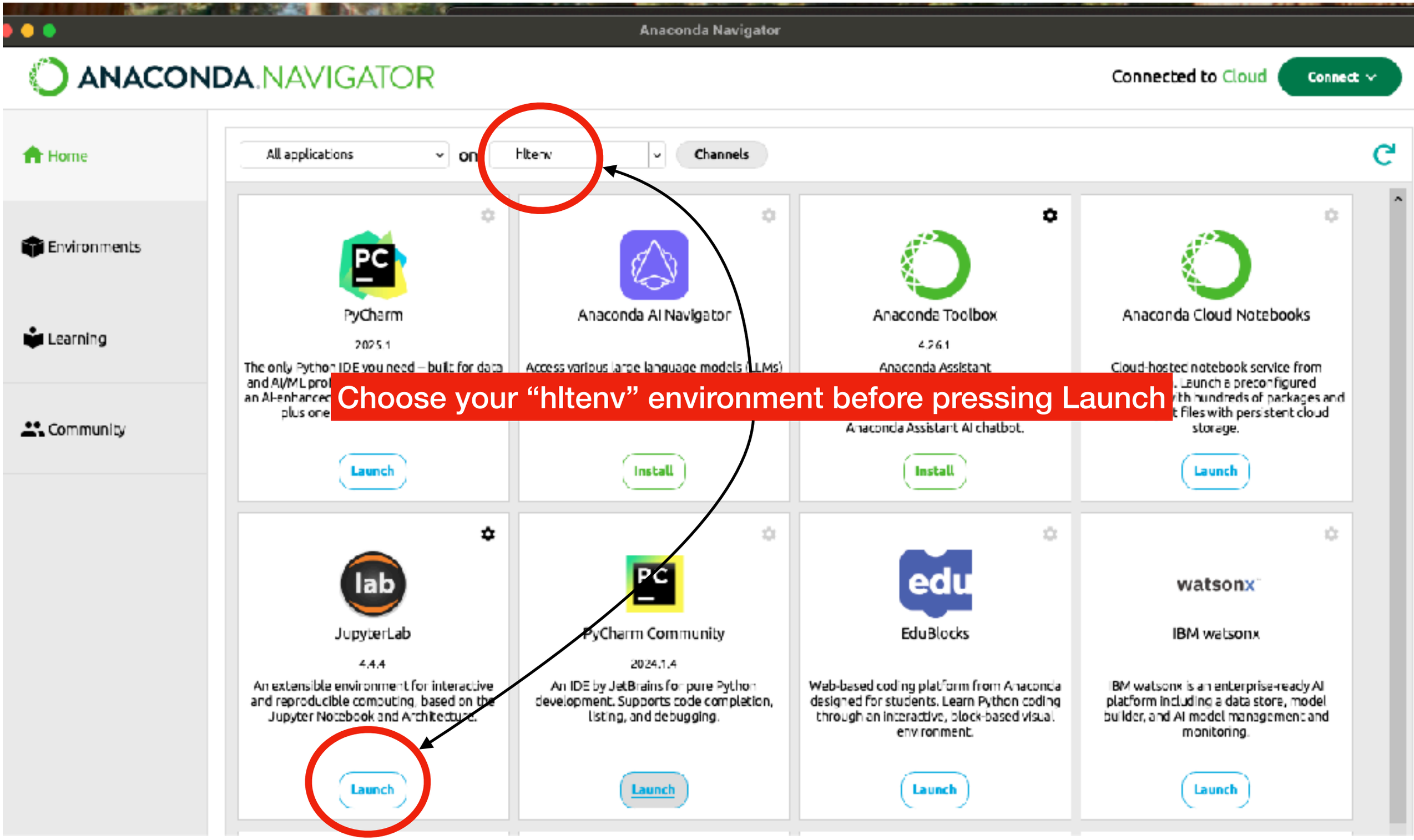
```
total 2176
-rw-r--r--   1 piek  staff  1105430 Sep  8 11:53 HLT-getting-started.pdf
-rw-r--r--   1 piek  staff     2786 Sep  8 11:53 README.md
drwxr-xr-x   9 piek  staff      288 Sep  8 11:53 data-formats
drwxr-xr-x   6 piek  staff      192 Sep  8 11:53 lab0.getting_started
drwxr-xr-x   8 piek  staff      256 Sep  8 11:54 lab1.chat
drwxr-xr-x  12 piek  staff      384 Sep  8 11:54 lab2.word_meaning
drwxr-xr-x  11 piek  staff      352 Sep  8 11:53 lab3.text_meaning
drwxr-xr-x  10 piek  staff      320 Sep  8 11:53 lab4.annotation_and_evaluation
drwxr-xr-x  11 piek  staff      352 Sep  8 11:53 lab5.large_language_models
drwxr-xr-x   5 piek  staff      160 Sep  8 11:53 lab6.generative_llms
drwxr-xr-x   5 piek  staff      160 Sep  8 11:53 lab7.final_assignment
-rw-r--r--   1 piek  staff      377 Sep  8 11:53 requirements.txt
(hltenv) piek@piek-2 ma-hlt-labs % python -m pip install --upgrade pip

Requirement already satisfied: pip in /opt/anaconda3/envs/hltenv/lib/python3.11/site-packages (25.2)
(hltenv) piek@piek-2 ma-hlt-labs % pip install -r requirements.txt
```

>pip list now shows a long list of installed packages

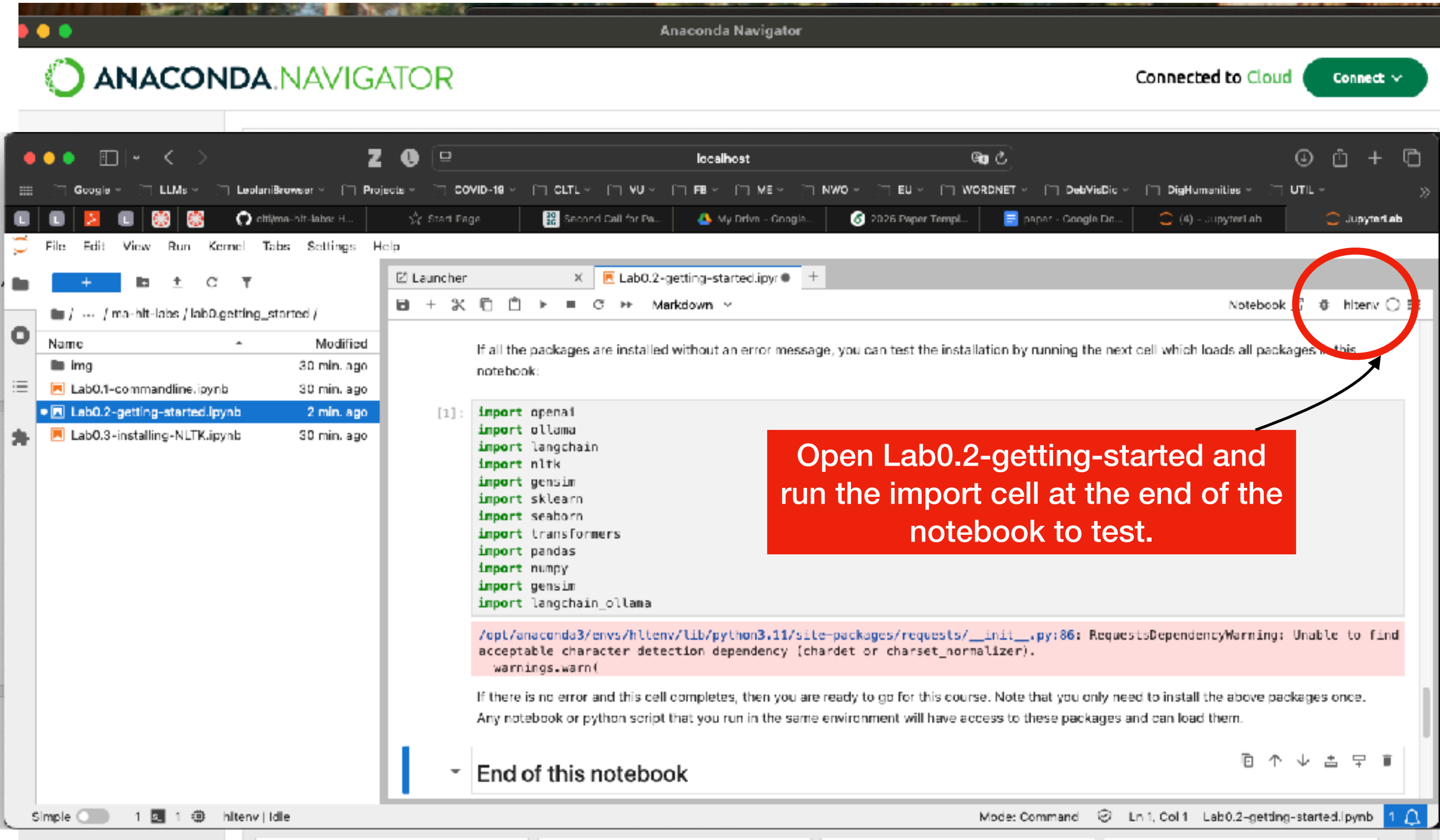
ma-hlt-labs — -zsh —

```
(hltenv) piek@piek-2 ma-hlt-labs % pip list
Package                  Version
------------------------ -----------
annotated-types          0.7.0
anyio                    4.10.0
blis                     0.7.11
catalogue                2.0.10
certifi                  2025.8.3
charset-normalizer       3.4.3
click                    8.2.1
confection               0.1.5
contourpy                1.3.3
cycler                   0.12.1
cymem                    2.0.11
distro                   1.9.0
filelock                 3.19.1
fonttools                4.59.2
fsspec                   2025.9.0
gensim                   4.3.3
greenlet                 3.2.4
h11                      0.16.0
hf-xet                   1.1.9
                         1.0.9
                         0.28.1
                         0.34.4
                         3.10
                         3.1.6
                         0.10.0
joblib                   1.5.2
jsonpatch                1.33
jsonpointer              3.0.0
kiwisolver               1.4.9
langchain                0.3.21
langchain-core           0.3.75
langchain-ollama         0.2.1
langchain-text-splitters 0.3.11
langsmith                0.3.45
MarkupSafe               3.0.2
matplotlib               3.10.3
mpmath                   1.3.0
murmurhash               1.0.13
```

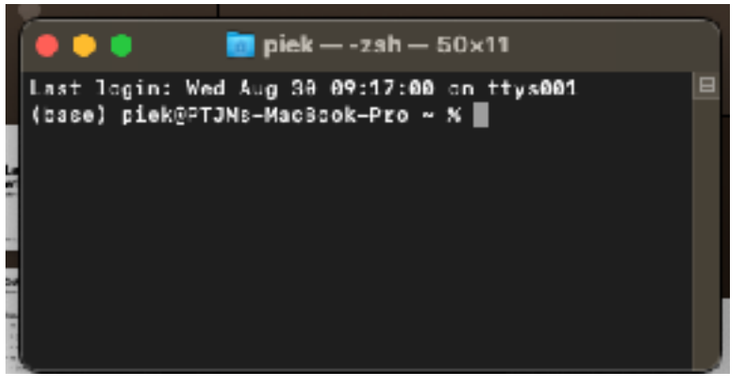
Launching jupyter lab in your “hltlenv” environment



Launching jupyter lab in your “hltlenv” environment



The Terminal



- Terminal or Command line
  - The Terminal or Command line lets you type in basic commands that will be carried out by the computer.
  - Working with a terminal gives you more control over your code, is more efficient, and is the only way to run code on remote servers that do not come with a graphical interface.
  - We will use the terminal that can be launched from the Anaconda environment to make sure it uses the right environment and version of Python

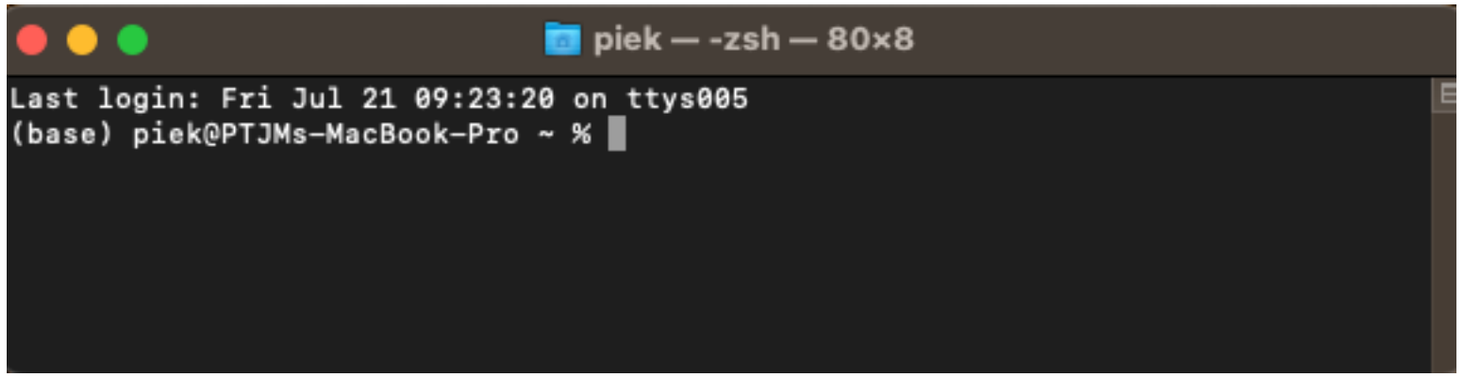
Using a terminal

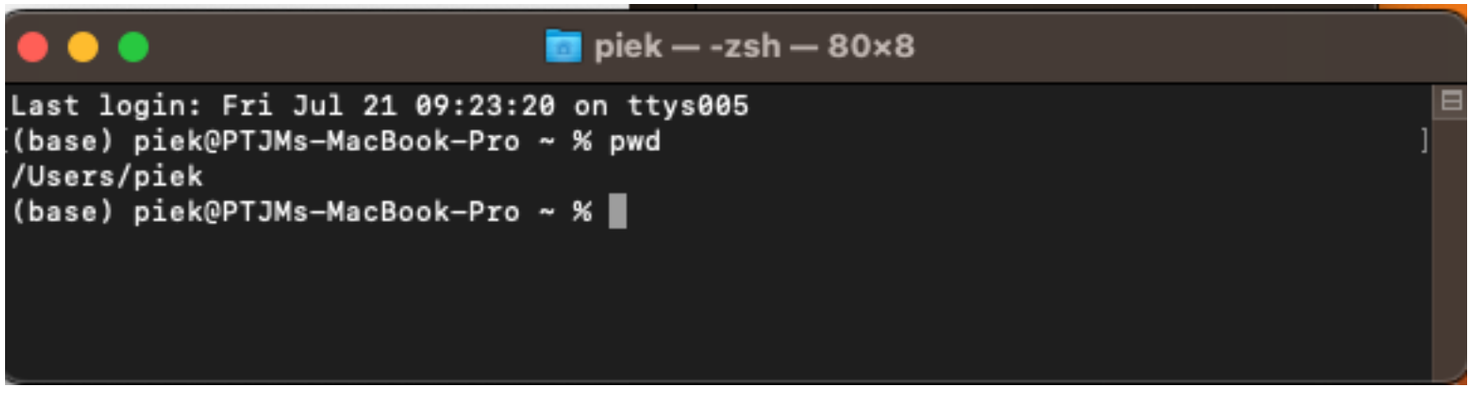
- A terminal gives you access to a so-called Unix/Linux *shell*.
  - Working with shell commands is extremely fast and efficient.
  - A taste of the power of shell commands "Unix for Poets" by Kenneth Church:
    - https://www.cs.upc.edu/~padro/Unixforpoets.pdf
    - How to count words, sort wordlists, make n-grams and make concordances for large amounts of text just using shell commands
- Basic commands for most of the classes (some will not work for Windows, for some there is a Windows alternative):
  - **pwd**: gives the path to the current directory
  - **ls** (Mac/Linux), **dir** (Windows): gives a list of what is stored in the current directory
  - **cd**: change directory, either going up to the parent or going in a subdirectory
  - **mkdir**: create a new directory in the current directory
  - **touch**: create a new empty file in the current directory
  - **echo** & **>**: return any text between quotes which can be redirected as a stream, e.g. echo “hello world” > file.txt
  - **cat:** (type Windows): print the content of a file to the screen
  - **mv**: move a file or rename a file
  - **rmdir**: remove a subfolder when empty
  - **rm**: permanently remove files and folders

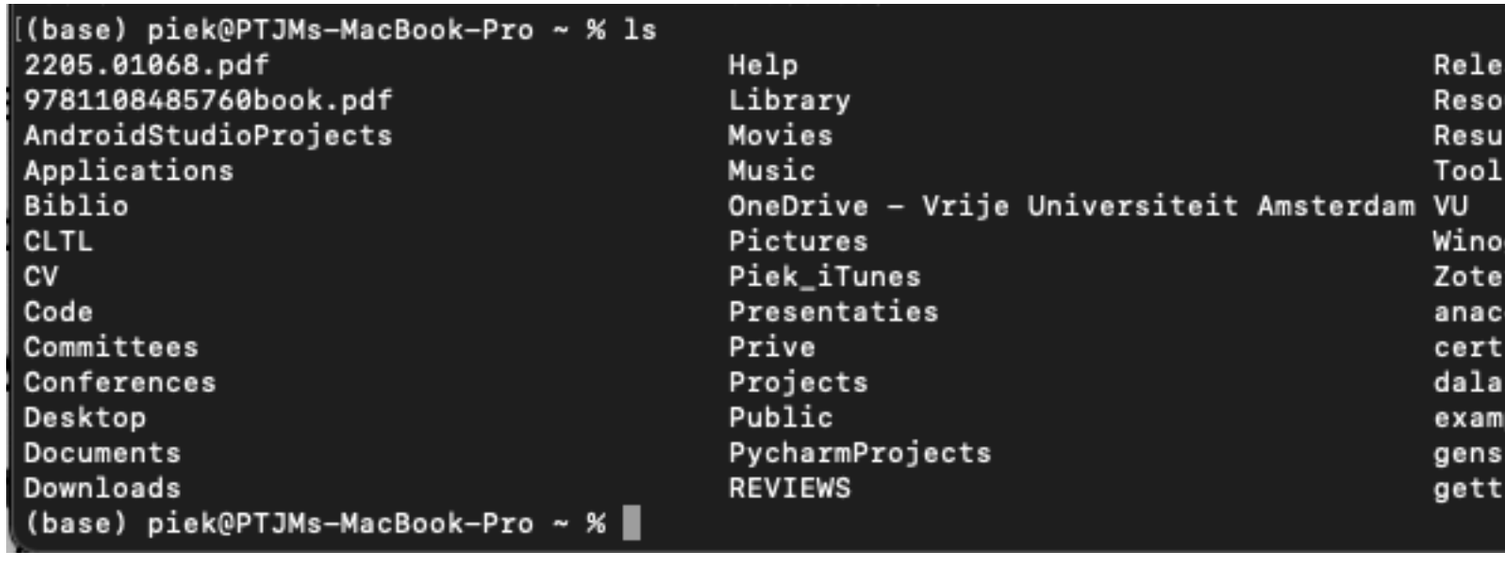
Using a terminal

pwd, ls, dir

- When you open a terminal or command line box, you will be somewhere on your hard disk.
- You will see a prompt (% or >) after which you can type a command to the computer.
- Where are you on your disk?
    - In your terminal, type “pwd” directly after the prompt (%) and hit enter. My computer echos “/Users/piek” which is my home directory
- What is in my home dir?
    - Type “ls” (Mac/Linux) or “dir” (Windows) and you get a listing of files and subdirectories in the directory that you are now
    - My home directory has familiar subdirectories such as “Desktop”, “Documents” and “Downloads” and lots of other stuff







Using a terminal

cd

- The “cd” command stands for change directory and expects the name of a subdirectory
- Try any subdirectory that is listed, here we move into the “Downloads” subdirectory “cd Downloads”:
    - TIP: type “cd Downl” and press the TAB key. You will see that it is autocompleted to “cd Downloads”. This comes in handy when you have to type long names or pathes.
- Before the prompt “%”, we now see “Downloads” appear. Let’s use “pwd” again to check the path.
    - “/Users/piek/Downloads” so we went down into a subdirectory from my home dir.
- Using the command “ls” we can see what is in Downloads.
    - We see here folder with the name “old” and the folder “ma-hlt-labs-master” with the lab sessions that we downloaded from Github.
    - We can use “cd” again to enter it. Type “cd ma-hlt” and use the TAB key to complete.
- We use “ls” to get a listing. The option “-l” also make the terminal show details such as creation date/time, size and ownership. The list shows the different subfolders for the lab sessions of this course. Use “pwd” to see the full path.
- So far we went down but you also want to go up to a parent directory or grandparent. For this we use “cd ..”, where the double periods stand for one-level up.
    - Using “cd ..”, we go back up to “Downloads”, again to “piek”, next to “Users” and finally to “/“ which is the top root of the disk (different on Windows).
    - After going up, we go down again to “/Users/piek”, where we started.

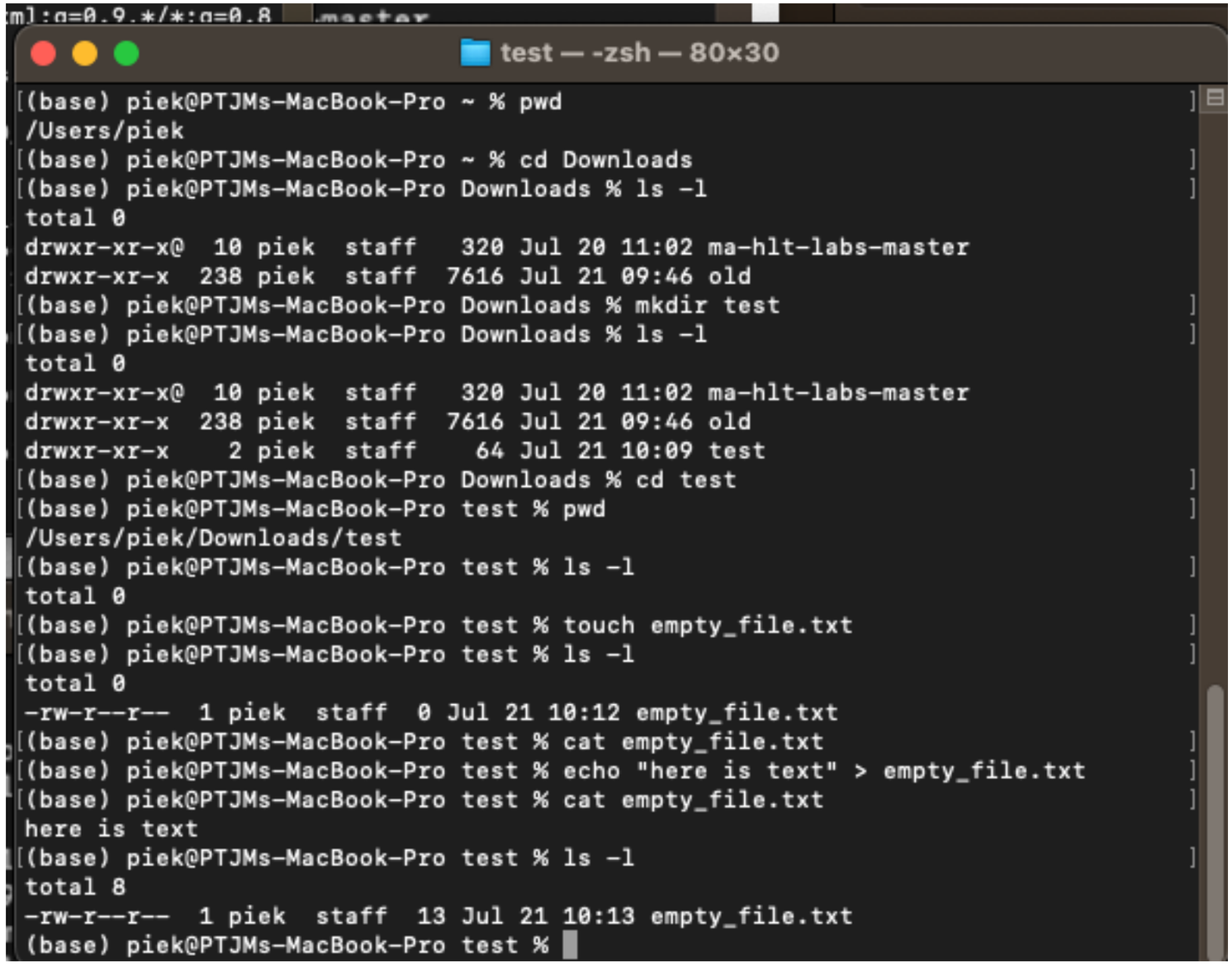
```
PycharmProjects
(base) piek@PTJMs-MacBook-Pro ~ % cd Downloads
(base) piek@PTJMs-MacBook-Pro Downloads % pwd
/Users/piek/Downloads
(base) piek@PTJMs-MacBook-Pro Downloads % ls
ma-hlt-labs-master      old
(base) piek@PTJMs-MacBook-Pro Downloads % cd ma-hlt-labs-master
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % ls -l
total 8
-rw-r--r--@  1 piek  staff  2520 Jul 20 11:02 README.md
drwxr-xr-x@  9 piek  staff   288 Jul 20 11:02 data-formats
drwxr-xr-x@ 10 piek  staff   320 Jul 20 11:02 lab1.toolkits
drwxr-xr-x@ 18 piek  staff   576 Jul 20 11:02 lab2.word_meaning
drwxr-xr-x@ 16 piek  staff   512 Jul 20 11:02 lab3.machine_learning
drwxr-xr-x@  9 piek  staff   288 Jul 20 11:02 lab4.contextualized-models
drwxr-xr-x@ 15 piek  staff   480 Jul 20 11:02 lab5.final_assignment
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % pwd
/Users/piek/Downloads/ma-hlt-labs-master
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master %
```

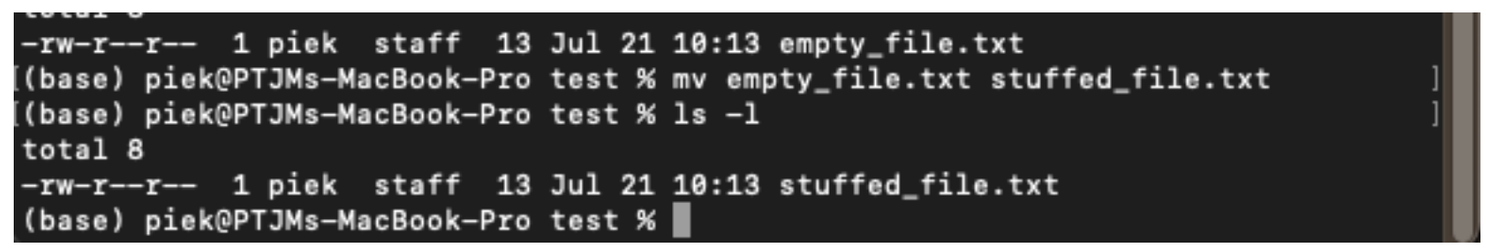
```
drwxr-xr-x@ 15 piek  staff   480 Jul 20 11:02 lab5.final_assignment
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % pwd
/Users/piek/Downloads/ma-hlt-labs-master
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % cd ..
(base) piek@PTJMs-MacBook-Pro Downloads % pwd
/Users/piek/Downloads
(base) piek@PTJMs-MacBook-Pro Downloads % cd ..
(base) piek@PTJMs-MacBook-Pro ~ % pwd
/Users/piek
(base) piek@PTJMs-MacBook-Pro ~ % cd ..
(base) piek@PTJMs-MacBook-Pro /Users % pwd
/Users
(base) piek@PTJMs-MacBook-Pro /Users % cd ..
(base) piek@PTJMs-MacBook-Pro / % pwd
/
(base) piek@PTJMs-MacBook-Pro / % cd Users
(base) piek@PTJMs-MacBook-Pro /Users % cd piek
(base) piek@PTJMs-MacBook-Pro ~ % pwd
/Users/piek
(base) piek@PTJMs-MacBook-Pro ~ %
```

Using a terminal

mkdir, touch, echo, cat. mv

- In addition to navigating, you can also create directories and files.
- **mkdir** makes directories:
  - In the terminal screen dump, we navigated back to Downloads and used “mkdir test” to make a new directory.
  - Using “cd” we can enter it and get a listing, which shows it is empty: “total 0”.
- **touch** makes files:
  - We can use “touch” (Mac/Linux) to make a new file inside the “test” folder.
  - When we get a listing for “test”, we now see the file but it is 0 kb in size (empty).
- **cat** (type Windows) shows the content
  - The “cat” command (type on Windows) prints the content which is empty.
  - We use the “echo” command to return a text “here is a text” which we next redirect using “>” as a stream into the empty file.
  - We use “cat empty_file.txt” again to print its content. It is not longer empty as is also shown when we get a listing of the “test” directory: 13 kb.
- **mv** moves or renames a file:
  - The file is no longer empty so let’s rename it.
  - For this, we use the “mv” command (Mac/Linux), which can be used for moving files as well as renaming if the target directory is the same but the filename is different. Try “mv empty_file.txt stuffed_file.txt” and get a new listing.

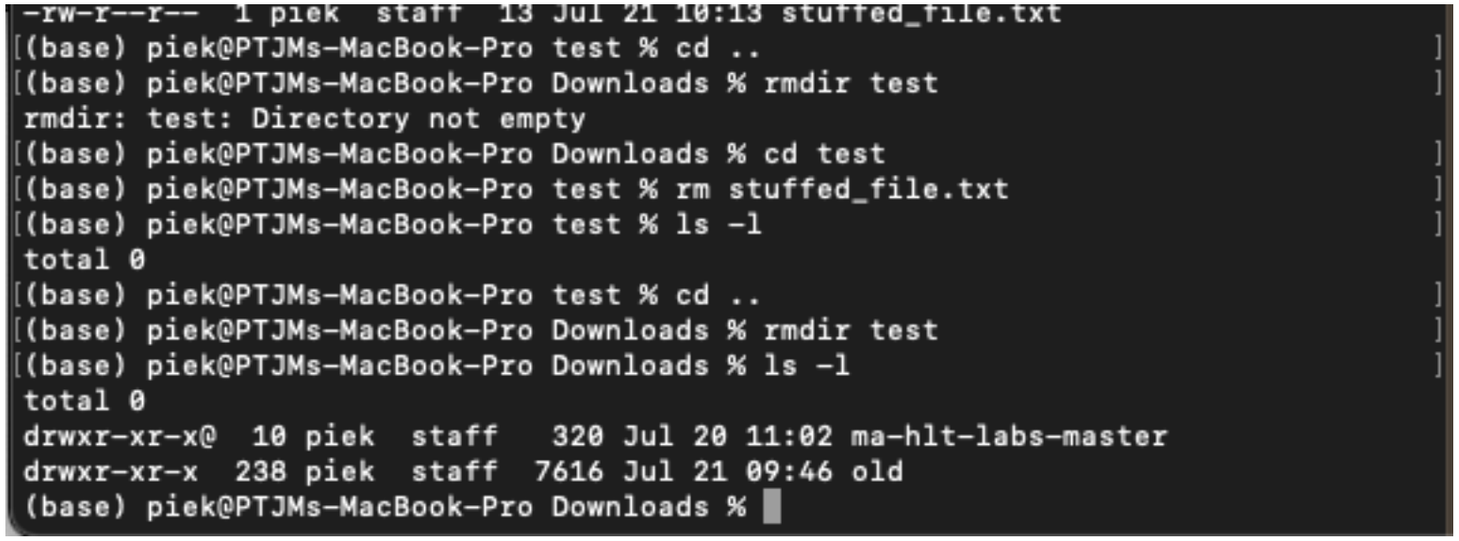




Using a terminal

rmdir, rm

- We created a “test” folder but now we want to clean up the mess.
- Removing by command line is efficient but also risky because there is no Trash to recover from.
    - “**rmdir**” removes a directory but only works when it is empty
    - “**rm**” removes files but with the option “**-r**” for recursively it also remove any subdirectory while emptying it first (everything from that point on is gone).
- We first try to remove the directory “test”. We navigate up to Downloads and type “rmdir test”:
    - We get the message: rmdir: test: Directory not empty, so this failed
- To remove it, we first need to empty it, so we navigate back in and use “rm stuffed_file.txt” to get rid of the file.
- Now we can go back up and use “rmdir test” from Downloads. The listing shows that it worked.
- We could have been more efficient by using “rm -r test” from Downloads, which would remove the content of “test” (both files and any subdirectories) and “test” itself.
    - However, this is very risky. Before doing this, check the content carefully using a listing, also check any subdirectories.
    - Doing this from the root (the top directory of your disk), will empty the full disk permanently



```
-rw-r--r--  1 piek  staff  13 Jul 21 10:13 stuffed_file.txt
(base) piek@PTJMs-MacBook-Pro test % cd ..
(base) piek@PTJMs-MacBook-Pro Downloads % rmdir test
rmdir: test: Directory not empty
(base) piek@PTJMs-MacBook-Pro Downloads % cd test
(base) piek@PTJMs-MacBook-Pro test % rm stuffed_file.txt
(base) piek@PTJMs-MacBook-Pro test % ls -l
total 0
(base) piek@PTJMs-MacBook-Pro test % cd ..
(base) piek@PTJMs-MacBook-Pro Downloads % rmdir test
(base) piek@PTJMs-MacBook-Pro Downloads % ls -l
total 0
drwxr-xr-x@  10 piek  staff   320 Jul 20 11:02 ma-hlt-labs-master
drwxr-xr-x  238 piek  staff  7616 Jul 21 09:46 old
(base) piek@PTJMs-MacBook-Pro Downloads %
```

Text editors

What is in a text file?

- We will work with text that is stored in files.
- Typically, we do NOT use Word, HTML or PDF as these contain a lot more than just the text or they are in binary format.
- Our code needs the pure text as input to work.
- For inspecting the text, use a “plain” text editor (not Word):
    - Windows Notepad++: https://notepad-plus-plus.org/
    - Mac/Linux:
        - Atom: https://github.com/atom/atom/releases/tag/v1.60.0
    - Mac:
        - Bbedit: https://www.barebones.com/products/bbedit/
- You can peek into a file without opening it from the command line, using the “cat” command (Linux, Mac) or “type” (Windows) followed by the file name.