

# **Lab0 Getting started with NP**

**Introduction to Human language Technology**

# Overview

- AI and coding
- Preparations
- Jupyter notebooks
- Get the lab materials (notebooks, data) from Github
- How to use a terminal
- A word on Text editors
- Virtual environments

# 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 higher. 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 small 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...

# Getting the Lab notebooks

*Check for updates*

Get the download link

- <https://github.com/cltl/ma-hlt-labs>

- Git installed:

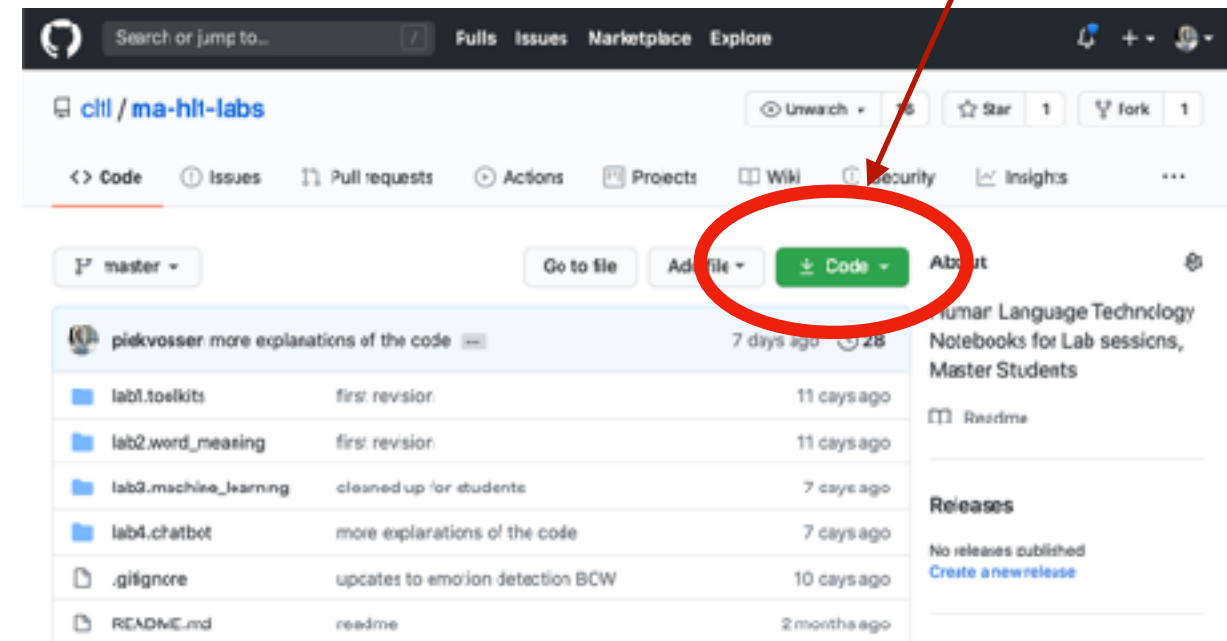
- Clone with ssh:

- `> git clone git@github.com:cltl/ma-hlt-labs.git`

- No local Git installed:

- Download ZIP file

- Unpack anywhere

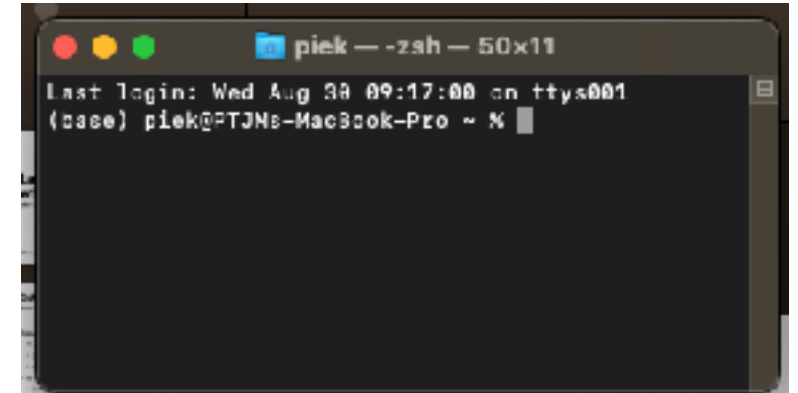


- `/Users/piek/Downloads/ma-hlt-labs/`
  - lab1.chat
  - lab2.word\_meaning
  - lab3.machine\_learning
  - Lab4.evaluation



- `ma-hlt-labs/`
  - lab1.chat
  - lab2.word\_meaning
  - lab3.machine\_learning
  - Lab4.evaluation

# 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.
  - Linux, Mac (Unix) users can launch a terminal:
    - In macOS: Open "Finder" ⇒ Go ⇒ Utilities ⇒ Select "Terminal".
    - In Ubuntu: Open "Dash" ⇒ type "Terminal"; or choose "Applications" ⇒ "Installed" ⇒ Select "Terminal".
  - Windows also has a terminal, which can be installed and activated as explained here:
    - <https://docs.microsoft.com/en-us/windows/terminal/get-started>
  - You can also simulate a linux terminal in Windows through Git bash:
    - <https://www.atlassian.com/git/tutorials/git-bash>
  - Useful links:
    - <https://www.ibm.com/developerworks/aix/library/au-unixtext/>
    - <https://tldp.org/LDP/intro-linux/html/intro-linux.html>
    - [https://www3.ntu.edu.sg/home/ehchua/programming/howto/Unix\\_SurvivalGuide.html](https://www3.ntu.edu.sg/home/ehchua/programming/howto/Unix_SurvivalGuide.html)



# 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

```
piek — -zsh — 80x8
Last login: Fri Jul 21 09:23:20 on ttys005
(base) piek@PTJMs-MacBook-Pro ~ %
```

```
piek — -zsh — 80x8
Last login: Fri Jul 21 09:23:20 on ttys005
(base) piek@PTJMs-MacBook-Pro ~ % pwd
/Users/piek
(base) piek@PTJMs-MacBook-Pro ~ %
```

```
(base) piek@PTJMs-MacBook-Pro ~ % ls
2205.01068.pdf      Help              Rele
9781108485760book.pdf  Library          Reso
AndroidStudioProjects  Movies           Resu
Applications         Music            Tool
Biblio               OneDrive - Vrije Universiteit Amsterdam VU
CLTL                 Pictures         Wino
CV                   Piek_iTunes     Zote
Code                 Presentaties    anac
Committees          Prive           cert
Conferences         Projects        dala
Desktop             Public          exam
Documents           PycharmProjects gens
Downloads           REVIEWS        gett
(base) piek@PTJMs-MacBook-Pro ~ %
```

# 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.

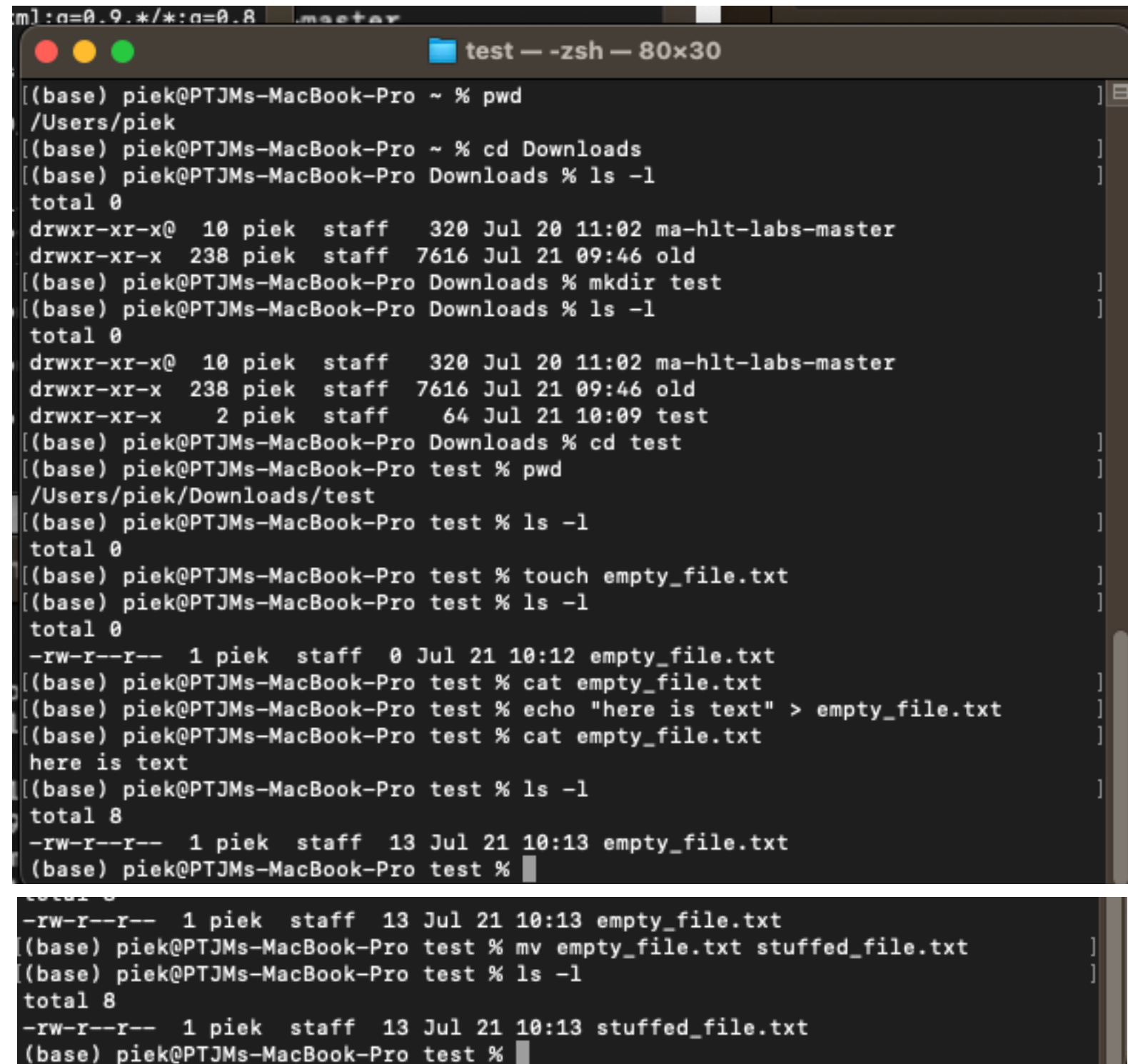
```
PycharmProjects
1. [(base) piek@PTJMs-MacBook-Pro ~ % cd Downloads ]
[(base) piek@PTJMs-MacBook-Pro Downloads % pwd ]
/Users/piek/Downloads
co [(base) piek@PTJMs-MacBook-Pro Downloads % ls ]
co ma-hlt-labs-master      old
[(base) piek@PTJMs-MacBook-Pro Downloads % cd ma-hlt-labs-master ]
[(base) 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
ds [(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.



```
(base) piek@PTJMs-MacBook-Pro ~ % pwd
/Users/piek
(base) piek@PTJMs-MacBook-Pro ~ % cd Downloads
(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 % mkdir 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
drwxr-xr-x   2 piek  staff   64 Jul 21 10:09 test
(base) piek@PTJMs-MacBook-Pro Downloads % cd test
(base) piek@PTJMs-MacBook-Pro test % pwd
/Users/piek/Downloads/test
(base) piek@PTJMs-MacBook-Pro test % ls -l
total 0
(base) piek@PTJMs-MacBook-Pro test % touch empty_file.txt
(base) piek@PTJMs-MacBook-Pro test % ls -l
total 0
-rw-r--r--  1 piek  staff   0 Jul 21 10:12 empty_file.txt
(base) piek@PTJMs-MacBook-Pro test % cat empty_file.txt
(base) piek@PTJMs-MacBook-Pro test % echo "here is text" > empty_file.txt
(base) piek@PTJMs-MacBook-Pro test % cat empty_file.txt
here is text
(base) piek@PTJMs-MacBook-Pro test % ls -l
total 8
-rw-r--r--  1 piek  staff  13 Jul 21 10:13 empty_file.txt
(base) piek@PTJMs-MacBook-Pro test % mv empty_file.txt stuffed_file.txt
(base) piek@PTJMs-MacBook-Pro test % ls -l
total 8
-rw-r--r--  1 piek  staff  13 Jul 21 10:13 stuffed_file.txt
(base) piek@PTJMs-MacBook-Pro test %
```



# 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 %
```

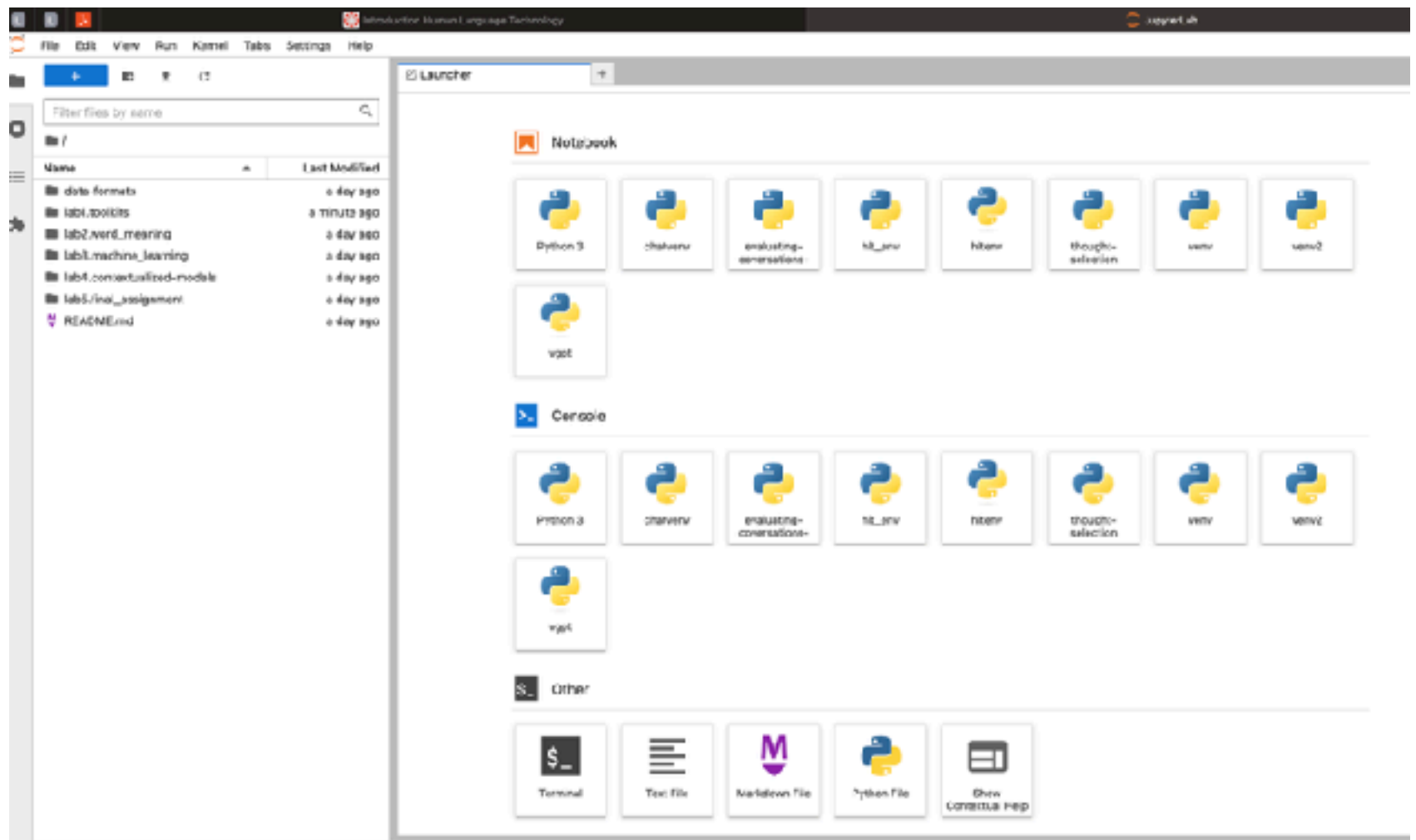
# Launching Jupyter notebooks

- If you have successfully installed Anaconda, you should be able to launch Jupyter lab from the command line:

1. open the command line or a terminal
2. navigate to the directory where the notebooks are (see the navigation instructions)
3. Type "jupyter lab" after the prompt and press enter as shown here.
4. Your default browser should open the page as shown below with the lab session folders to the left.
5. Navigate to the folder where you stored the notebooks for Lab 1 and open the notebook with the introduction.

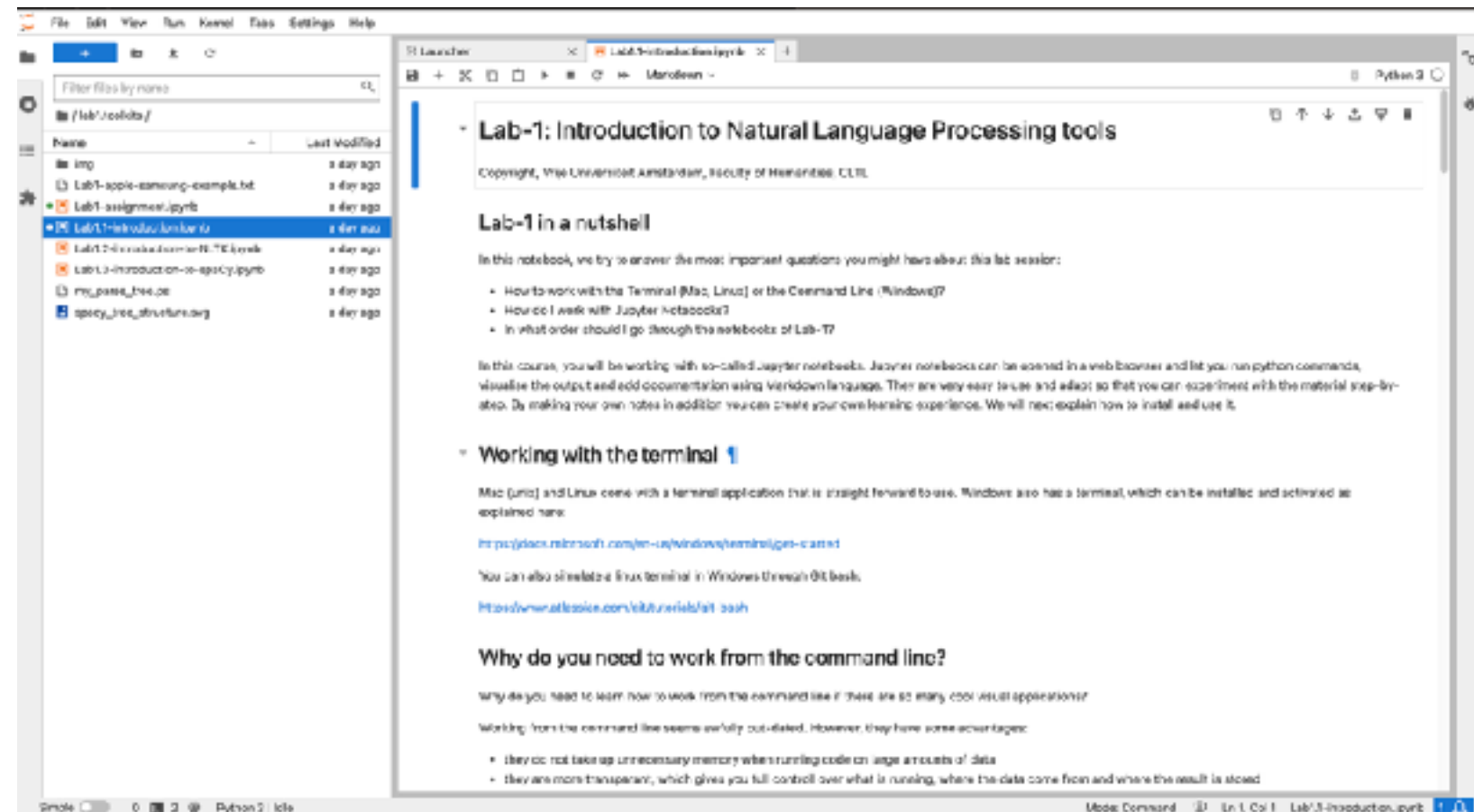
- Documentation for JupyterLab can be found here: <https://jupyterlab.readthedocs.io/en/stable/>

```
ma-hlt-labs-master — -zsh — 80x24
Last login: Fri Jul 21 09:58:57 on ttys001
(base) piek@PTJMs-MacBook-Pro ~ % cd Downloads/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 % jupyter lab
```



# Using Jupyter notebooks

- Notebooks contain instructions and so-called 'code blocks'. The instructions are paragraphs of text that explain the concepts we are going to use. The 'code blocks' contain Python code.
- Some tips:
  - 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 Kernel 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/>
- To stop Jupyter you can close the tab in the browser but you also need to stop Jupyter in the terminal:
  - Go to the terminal and press CTRL-c This will interrupt the program and ask you to confirm: Shutdown this Jupyter server (y/[n])?
  - If confirmed it terminates
  - You can also press CTRL-c twice to kill it directly.



```
ma-hit-labs-master ~ -zsh - 80x24
[2023-07-21 18:40:01.984 ServerApp] Got events for closed stream <zmq.eventloop
p.ZMQStream.ZMQStream object at 0x1103ea7c0>
[2023-07-21 18:40:09.707 ServerApp] Starting buffering for 32eca775-5469-49e6-9f6b-0ee9ebb1cb5
[2023-07-21 18:41:42.192 ServerApp] Connecting to kernel a0164e17-c83f-4430-8c
a3-35720326e3ba.
^C [2023-07-21 18:49:39.570 ServerApp] interrupted
[2023-07-21 18:49:39.570 ServerApp] Serving notebooks from local directory: /U
ser/piek/Downloads/ma-hit-labs-master
2 active kernels
Jupyter Server 2.6.0 is running at:
http://localhost:8880/?token=5b57e8d31ea57a498878483d1563be5eb3da2e0c
9544
http://127.0.0.1:8880/?token=5b57e8d31ea57a498878483d1563be5eb3da2e0c
61ec9544
Shutdown this Jupyter server (y/[n])? y
[2023-07-21 18:49:34.823 ServerApp] Shutdown confirmed
[2023-07-21 18:49:34.825 ServerApp] Shutting down 6 extensions
[2023-07-21 18:49:34.826 ServerApp] Shutting down 2 kernels
[2023-07-21 18:49:34.827 ServerApp] Kernel shutdown: a0164e17-c83f-4430-8c
a3-35720326e3ba
[2023-07-21 18:49:34.828 ServerApp] Kernel shutdown: 32eca775-5469-49e6-9f6b-0
ee9ebb1cb5
(base) pick@PTJMs-MacBook-Pro ma-hit-labs-master %
```

# Text editors

## What is in a text file?

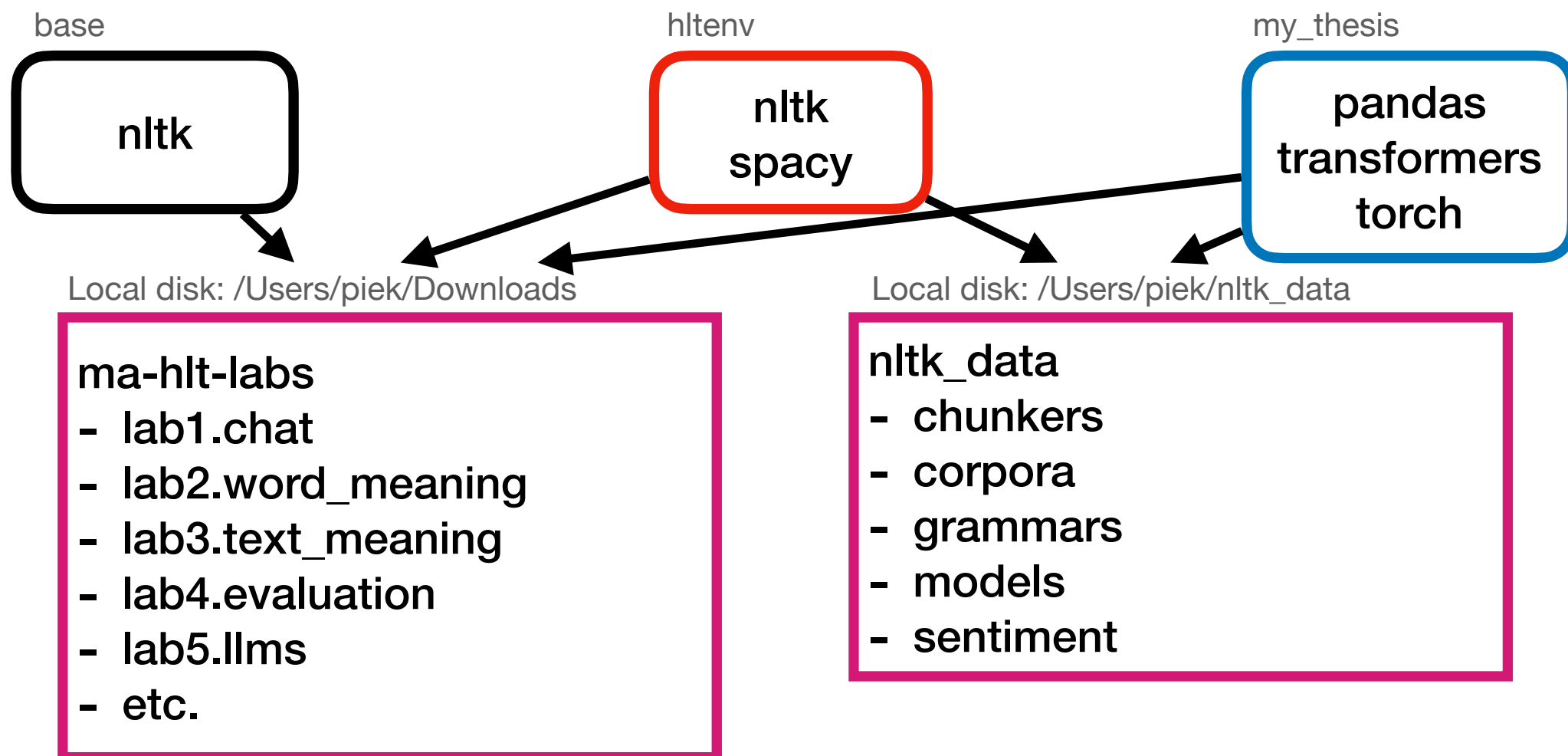
- We will work with text that is stored in files.
- Typically, we do NOT use Word, HTML or PDF as these contain a lot more than just the text or they are in binary format.
- Our code needs the pure text as input to work.
- For inspecting the text, use a “plain” text editor (not Word):
  - Windows Notepad++: <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.



# 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 Python virtual environment

## Virtual environment runs Python and packages in a save silo

Open a terminal on your local machine and navigate to the folder where you have unpacked the Github repository for **cltl/ma-hlt-labs**.

Once you are in the right location, you need to run the following commands in this order within the folder that you downloaded from Github:

1. Create a virtual environment with the name "hltenv":

```
(base)>python -m venv hltenv
```

A new folder "hltenv" is created on your disk, in which all packages will be installed in a local Python environment.

2. Check the path to the home variable in the pyenv.cfg file:

```
(base)>hltenv/pyenv.cfg
```

This file contains a specification for three configuration variables: home, include-system-site-packages and version. The home variable should be set to the folder where the python executable is of your virtual environment. Assuming that you created the virtual environment in: "/Users/piek/Downloads/ma-hlt-lab-master/hltenv":

- \* MAC/Linux: home = /Users/piek/Downloads/ma-hlt-lab-master/hltenv/bin
- \* Windows: home = /Users/piek/Downloads/ma-hlt-lab-master/hltenv/Scripts

If your home directory is different, adapt it to the correct path and save the pyenv.cfg file.

# Activating a Python virtual environment

## Virtual environment runs Python and packages in a save silo

2. Activate this environment:

# On Mac/Linux

```
(base)>source hltenv/bin/activate
```

# On Windows:

```
(base)>hltenv\Scripts\activate
```

After activation, your prompt is prefixed with (hltenv)

# On Mac/Linux

```
(hltenv)(base)>
```

3. Stop this environment:

# On Mac/Linux

```
(hltenv)(base)>deactivate  
(base)>
```

What is installed in my environment?

```
(hltenv) (base) piek@piek-2 test % pip list  
Package      Version  
-----  
pip           23.0.1  
setuptools    65.5.0  
  
[notice] A new release of pip is available: 23.0.1 -> 25.2  
[notice] To update, run: pip install --upgrade pip  
(hltenv) (base) piek@piek-2 test %
```

```
(hltenv) (base) piek@piek-2 test % ls -l hltenv/lib/python3.10/site-packages  
total 8  
drwxr-xr-x  5 piek  staff   160 Aug  2 13:04 _distutils_hack  
-rw-r--r--  1 piek  staff   151 Aug  2 13:04 distutils-precedence.pth  
drwxr-xr-x  9 piek  staff   288 Aug  2 13:04 pip  
drwxr-xr-x 10 piek  staff   320 Aug  2 13:04 pip-23.0.1.dist-info  
drwxr-xr-x  6 piek  staff   192 Aug  2 13:04 pkg_resources  
drwxr-xr-x 48 piek  staff  1536 Aug  2 13:04 setuptools  
drwxr-xr-x 10 piek  staff   320 Aug  2 13:04 setuptools-65.5.0.dist-info
```

# Activating a Python virtual environment

## Virtual environment runs Python and packages in a save silo

3. Upgrade pip to the latest version:

```
(hltenv)(base)>python -m pip install --upgrade pip
```

4. Install Jupyter in this environment:

```
(hltenv)(base)>pip install jupyter
```

```
#### making sure jupyter runs in the venv
```

```
(hltenv)(base)>pip install ipykernel
```

```
(hltenv)(base)>python -m ipykernel install --user --name=hltenv
```

5. Launch Jupyter lab within the environment:

You can launch Jupyter lab from the Anaconda graphical interface but you need to make sure you select the correct virtual environment. The best option is to open a terminal (this can be done through Anaconda/Environments as well by opening a terminal):

```
(hltenv)(base)>jupyter lab
```

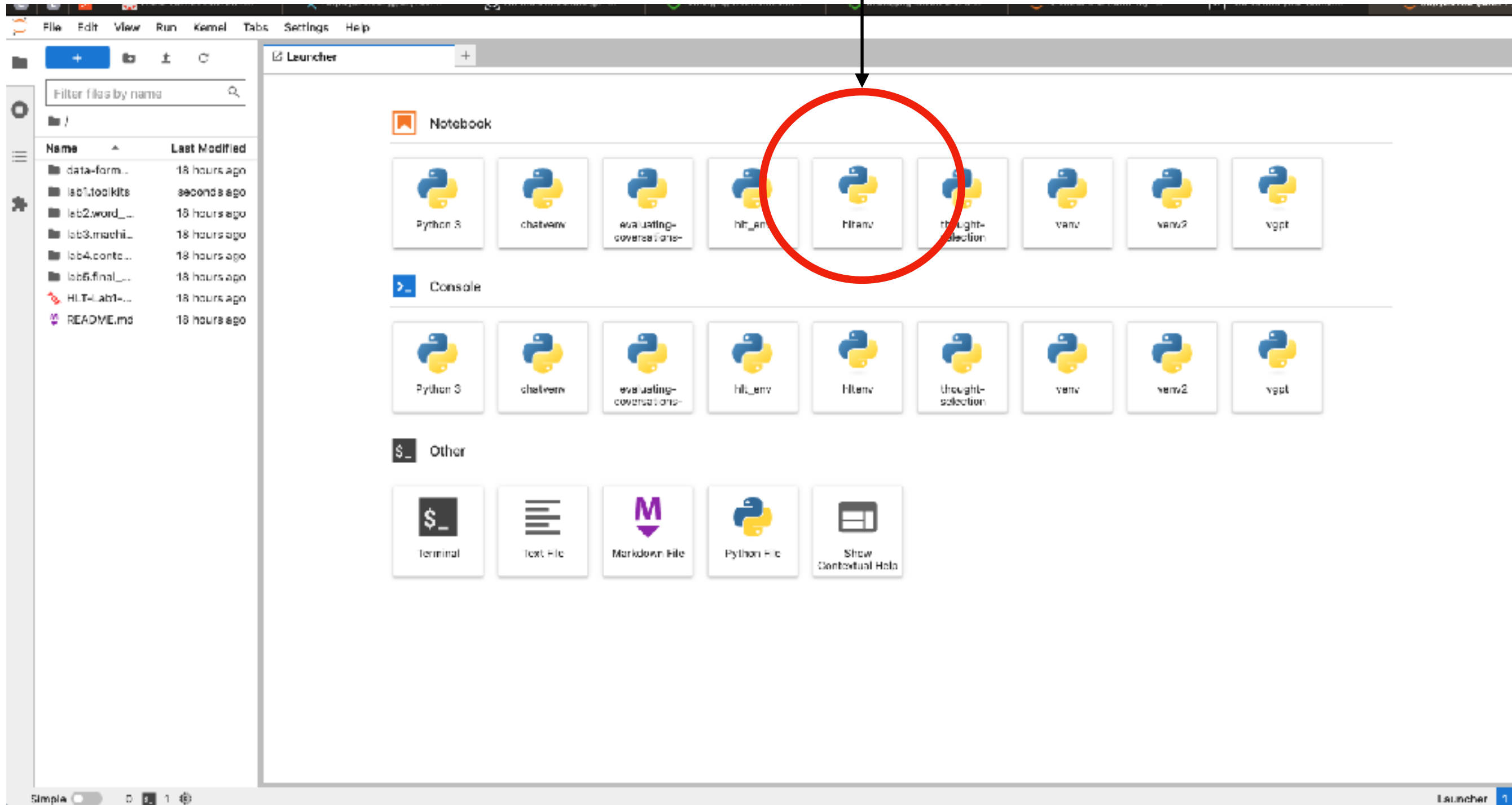
This will start Jupyter as a server. It also open your default browser and shows your the Jupyter lab interface that connects to this server as a client.

6. Stopping the server:

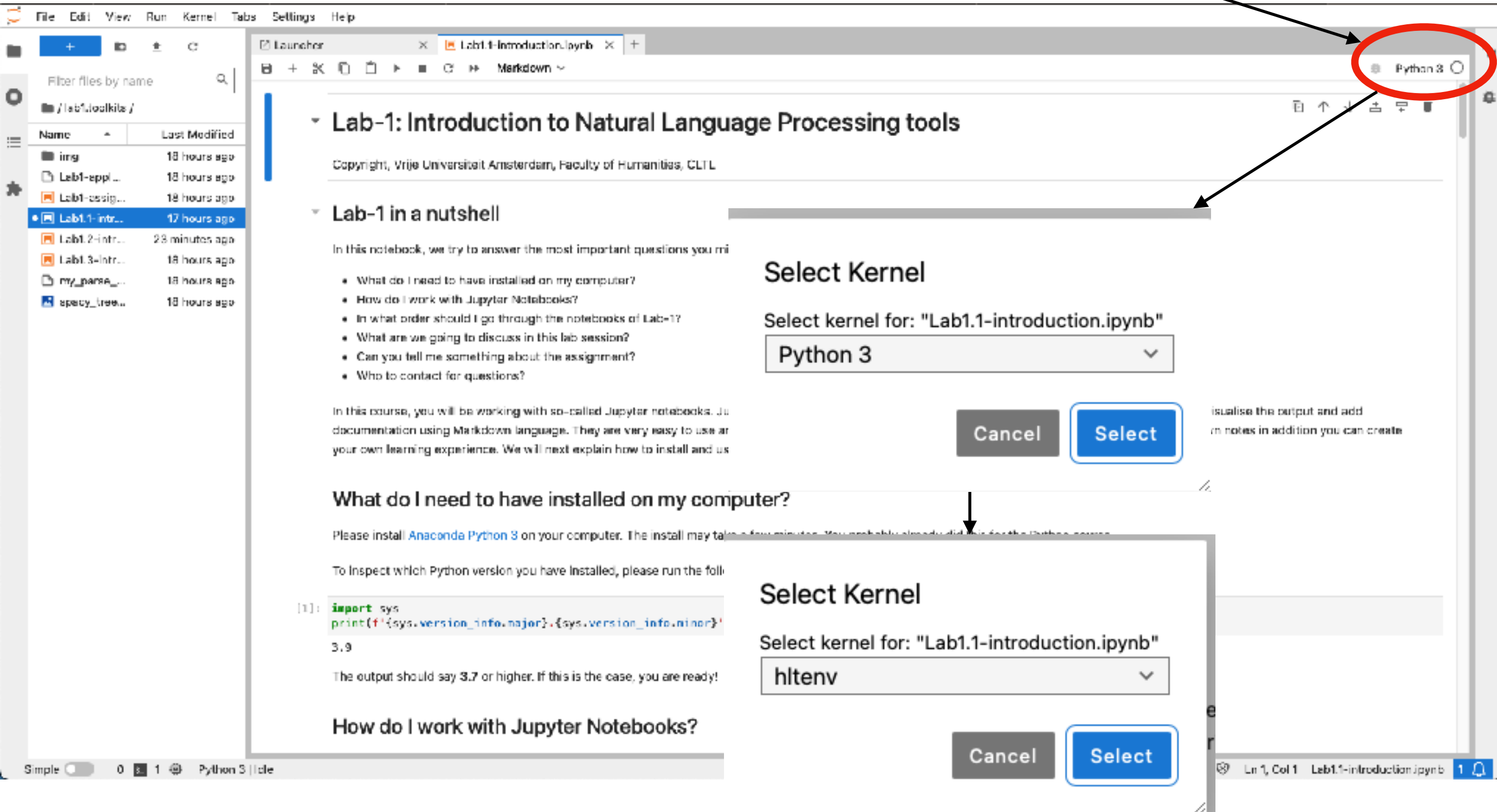
When you saved your work in the interface and you want to stop Jupyter you need to return to the terminal where Jupyter is running and press **CONTROL-C** (possibly multiple times) until the server is stopped and you can access the terminal again. You now can no longer use the browser interface either.

# Make sure Jupyter uses my hltenv

virtual environment is available



This notebook uses the standard Python 3 environment.  
Double click to change it to “htenv”



File Edit View Run Kernel Tabs Settings Help

Filter files by name

/lab1lookits/

Name	Last Modified
ing	18 hours ago
Lab1-appl...	18 hours ago
Lab1-assign...	18 hours ago
Lab1.1-intro...	17 hours ago
Lab1.2-intro...	23 minutes ago
Lab1.3-intro...	18 hours ago
my_parse...	18 hours ago
specy_tree...	18 hours ago

## Lab-1: Introduction to Natural Language Processing tools

Copyright, Vrije Universiteit Amsterdam, Faculty of Humanities, CLTL

### Lab-1 in a nutshell

In this notebook, we try to answer the most important questions you mi

- What do I need to have installed on my computer?
- How do I work with Jupyter Notebooks?
- In what order should I go through the notebooks of Lab-1?
- What are we going to discuss in this lab session?
- Can you tell me something about the assignment?
- Who to contact for questions?

In this course, you will be working with so-called Jupyter notebooks. Ju documentation using Markdown language. They are very easy to use ar your own learning experience. We will next explain how to install and us

### What do I need to have installed on my computer?

Please install [Anaconda Python 3](#) on your computer. The install may take a few minutes. You probably already did this for the Python course.

To inspect which Python version you have installed, please run the foll

```
[1]: import sys
print(f'{sys.version_info.major}.{sys.version_info.minor}')
```

3.9

The output should say 3.7 or higher. If this is the case, you are ready!

### How do I work with Jupyter Notebooks?

Simple 0 1 Python 3 | Idle

### Select Kernel

Select kernel for: "Lab1.1-introduction.ipynb"

Python 3

Cancel Select

visualise the output and add in notes in addition you can create

### Select Kernel

Select kernel for: "Lab1.1-introduction.ipynb"

htenv

Cancel Select

Ln 1, Col 1 Lab1.1-introduction.ipynb 1

This notebook now uses “htenv”

The screenshot shows a Jupyter Notebook environment. On the left is a file browser with a search bar and a list of files. The main area displays a notebook titled 'Lab1.1: Introduction to Natural Language Processing tools'. The notebook content includes a title, a copyright notice, a section 'Lab-1 in a nutshell', a list of questions, a paragraph about Jupyter notebooks, a section 'What do I need to have installed on my computer?', a paragraph about installing Anaconda Python 3, a code cell for checking the Python version, and a section 'How do I work with Jupyter Notebooks?'. A red circle highlights the 'htenv' button in the top right corner of the notebook interface.

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/lab1/toolkits/

Name	Last Modified
img	18 hours ago
Lab1-appl...	18 hours ago
Lab1-essig...	18 hours ago
Lab1.1-intr...	17 hours ago
Lab1.2-intr...	24 minutes ago
Lab1.3-intr...	18 hours ago
my_parse...	18 hours ago
spacy_tree...	18 hours ago

Lab1.1: Introduction to Natural Language Processing tools

Copyright, Vrije Universiteit Amsterdam, Faculty of Humanities, CLTL

### Lab-1 in a nutshell

In this notebook, we try to answer the most important questions you might have about this lab session:

- What do I need to have installed on my computer?
- How do I work with Jupyter Notebooks?
- In what order should I go through the notebooks of Lab-1?
- What are we going to discuss in this lab session?
- Can you tell me something about the assignment?
- Who to contact for questions?

In this course, you will be working with so-called Jupyter notebooks. Jupyter notebooks can be opened in a web browser and let you run python commands, visualise the output and add documentation using Markdown language. They are very easy to use and adapt so that you can experiment with the material step-by-step. By making your own notes in addition you can create your own learning experience. We will next explain how to install and use it.

### What do I need to have installed on my computer?

Please install [Anaconda Python 3](#) on your computer. The install may take a few minutes. You probably already did this for the Python course.

To inspect which Python version you have installed, please run the following cell (click on the cell and then click the play button at the top of this notebook):

```
[1]: import sys
print(f'{sys.version_info.major}-{sys.version_info.minor}')
3.9
```

The output should say 3.7 or higher. If this is the case, you are ready!

### How do I work with Jupyter Notebooks?

Simple 0 1 htenv | Idle

Mode: Command Ln 1, Col 1 Lab1.1-introduction.ipynb

# Installing packages in a Python virtual environment

- Navigate to the folder with the hlt-ma-labs in which your virtual environment should be located.
- Activate the environment:

Mac/Linux:

```
>source hltenv/bin/activate
```

Windows:

```
hltenv\Scripts\activate.bat
```

- If the virtual environment is active, install the required packages as follows:

```
(hltenv)(base)pip install -r requirements.txt
```

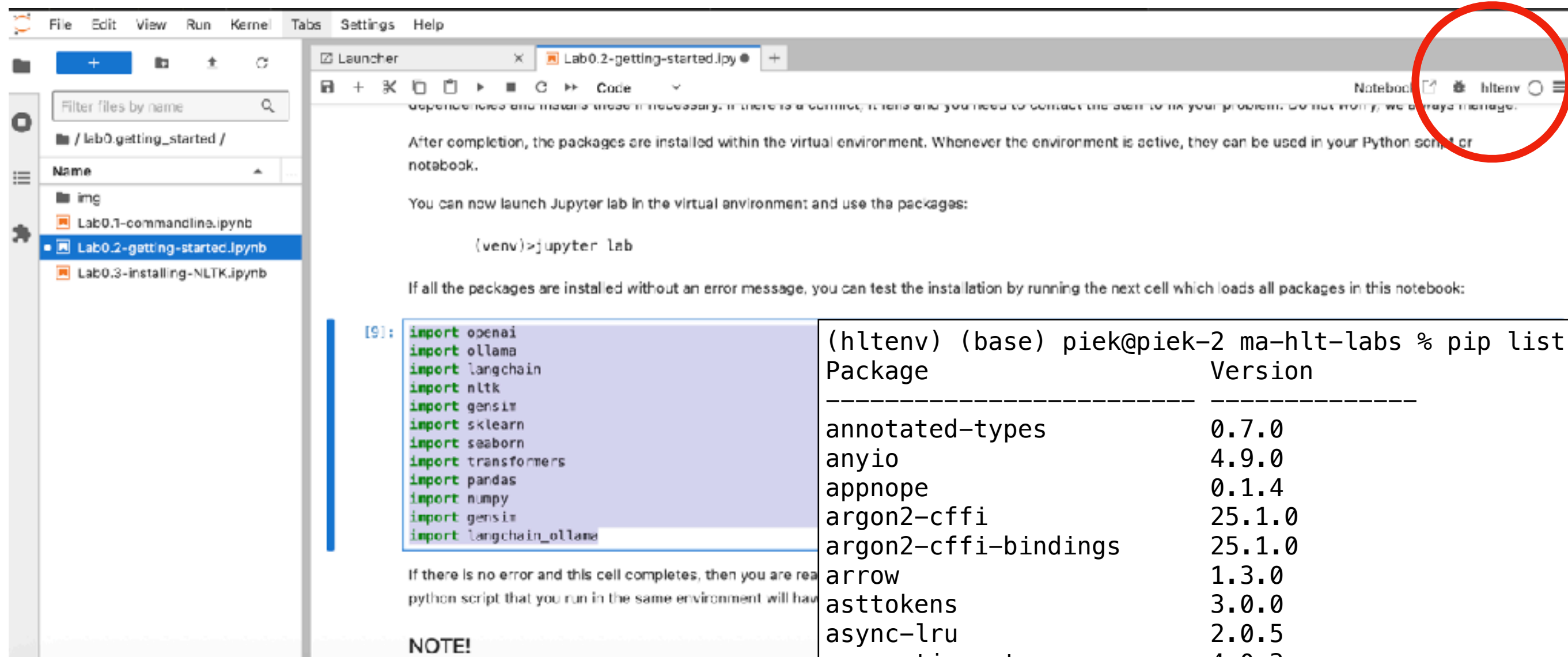
It may take a while to complete. It also checks dependencies and installs these if necessary. If there is a conflict, it fails and you need to contact the staff to fix your problem. After completion, the packages are installed within the virtual environment. If the environment is active, they can be used in your Python script or notebook that runs in the same environment.

- You can now launch Jupyter lab in the virtual environment to use the installed packages:

```
(hltenv)(base)>jupyter lab
```



# Installing packages in a Python virtual environment



The screenshot shows a JupyterLab interface with a file browser on the left and a notebook editor on the right. The notebook, titled 'Lab0.2-getting-started.ipynb', contains text explaining the installation of packages in a virtual environment. A code cell [9] lists imports for various packages. Below the code cell, there is a terminal output showing the result of a 'pip list' command. A red circle highlights the 'hltenv' button in the top right corner of the notebook interface.

File Edit View Run Kernel Tabs Settings Help

Filter files by name

/ lab0.getting\_started /

Name

- img
- Lab0.1-commandline.ipynb
- Lab0.2-getting-started.ipynb
- Lab0.3-installing-NLTK.ipynb

dependencies and materials (where necessary). If there is a connection error and you need to contact the team to fix your problem, do not worry, we always manage.

After completion, the packages are installed within the virtual environment. Whenever the environment is active, they can be used in your Python script or notebook.

You can now launch Jupyter lab in the virtual environment and use the packages:

```
(venv)>jupyter lab
```

If all the packages are installed without an error message, you can test the installation by running the next cell which loads all packages in this notebook:

```
[9]: import openai
import ollama
import langchain
import nltk
import gensim
import sklearn
import seaborn
import transformers
import pandas
import numpy
import gensim
import langchain_ollama
```

If there is no error and this cell completes, then you are ready to run a python script that you run in the same environment will have the same packages installed.

**NOTE!**

Package	Version
annotated-types	0.7.0
anyio	4.9.0
appnope	0.1.4
argon2-cffi	25.1.0
argon2-cffi-bindings	25.1.0
arrow	1.3.0
asttokens	3.0.0
async-lru	2.0.5
async-timeout	4.0.3
attrs	25.3.0
babel	2.17.0
beautifulsoup4	4.13.4
bleach	6.2.0
blis	0.7.11
catalogue	2.0.10
certifi	2025.8.3

```
(hltenv) (base) piek@piek-2 ma-hlt-labs % pip list | wc
171      342     5582
```