Lab0 Getting started with NP

Introduction to Human language Technology

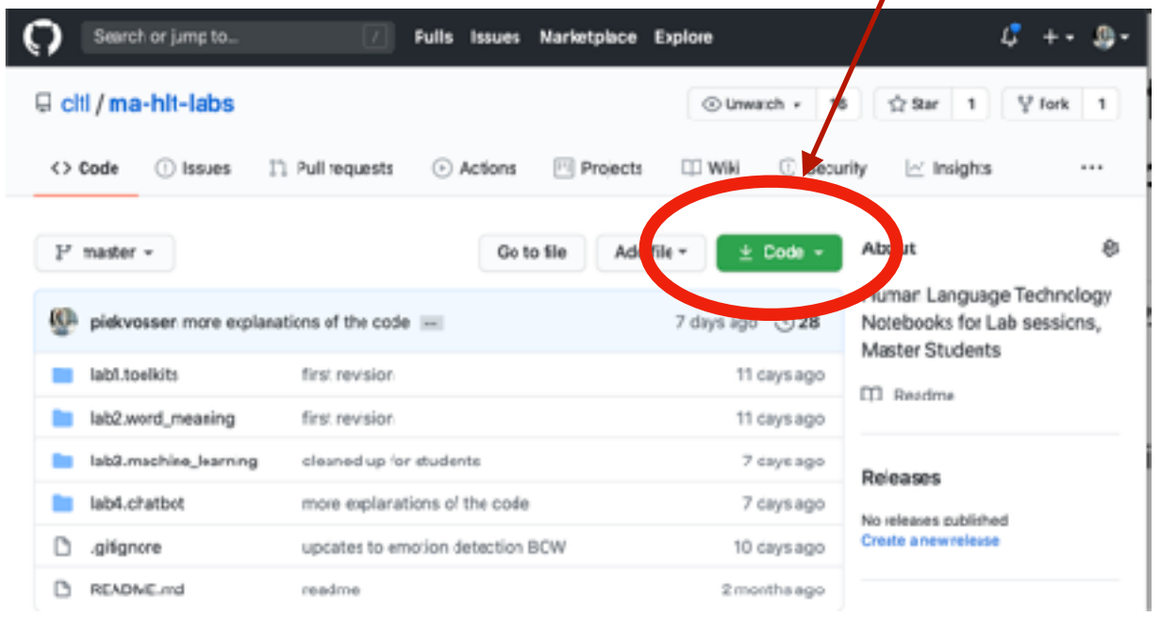
Piek Vossen

Overview

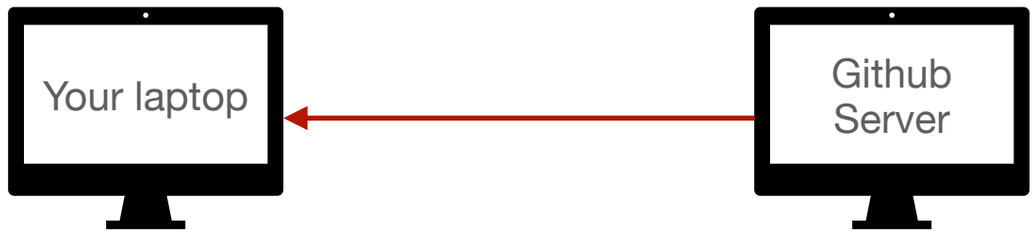
- Download the lab materials from Github
- How to use a terminal
- Text editors
- Virtual environments

Getting the Lab notebooks

Check for updates



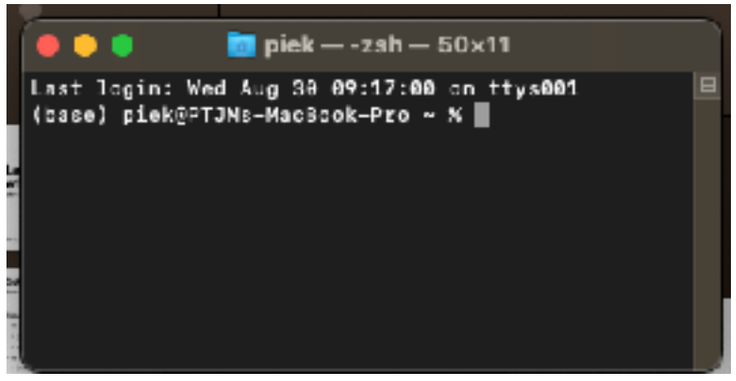
- https://github.com/cltl/ma-hlt-labs
  - Git installed:
    - Clone with ssh:
    - > git clone git@github.com:cltl/ma-hlt-labs.git
- No local Git installed:
  - Download ZIP file
  - Unpack anywhere



Preparations

- Hardware
  - Bring your own laptop (Windows, Mac or Linux are all fine) with rights to install software and data.
  - At least 8GB 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 more difficult.
- Python
  - If you follow 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 but work from a Terminal or Command line.

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

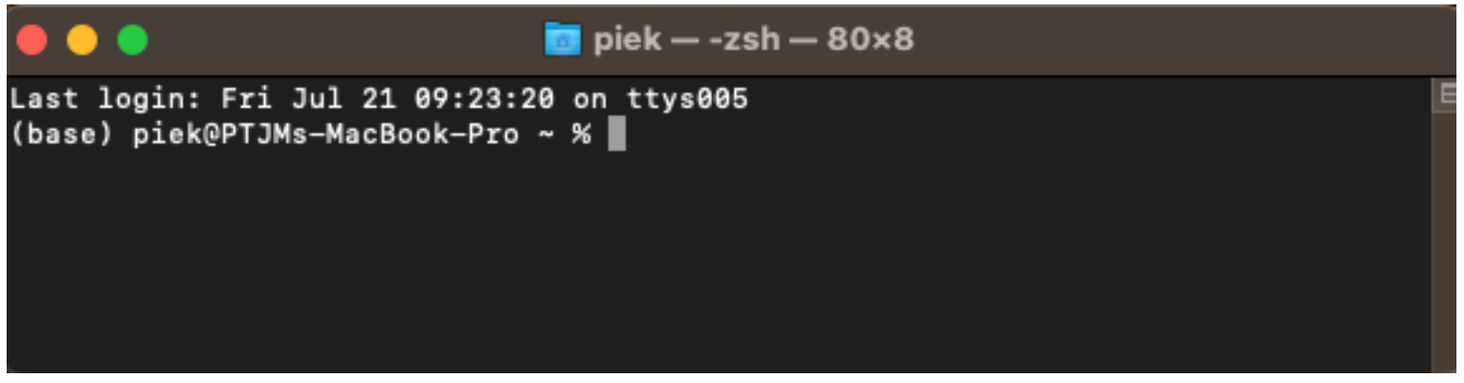
Using a terminal

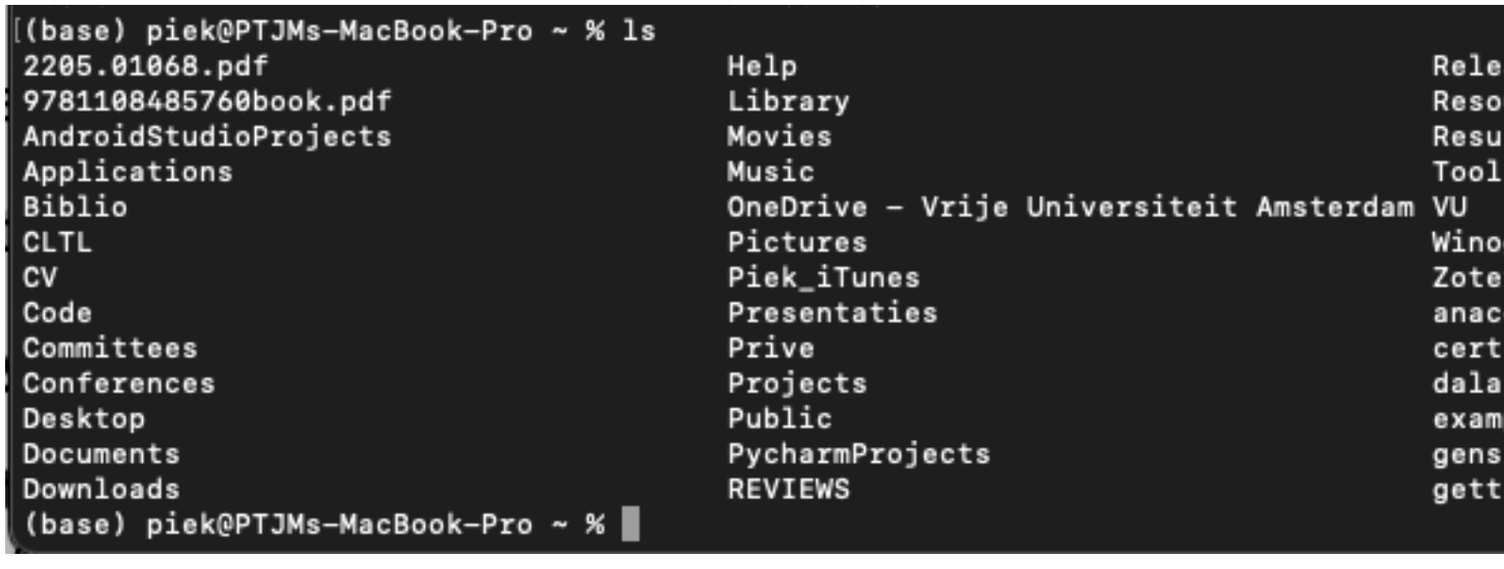
- A terminal gives you access to a so-called Unix/Linux *shell*.
    - Working with shell commands is extremely fast and efficient.
    - A taste of the power of shell commands "Unix for Poets" by Kenneth Church:
        - https://www.cs.upc.edu/~padro/Unixforpoets.pdf
        - How to count words, sort 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.

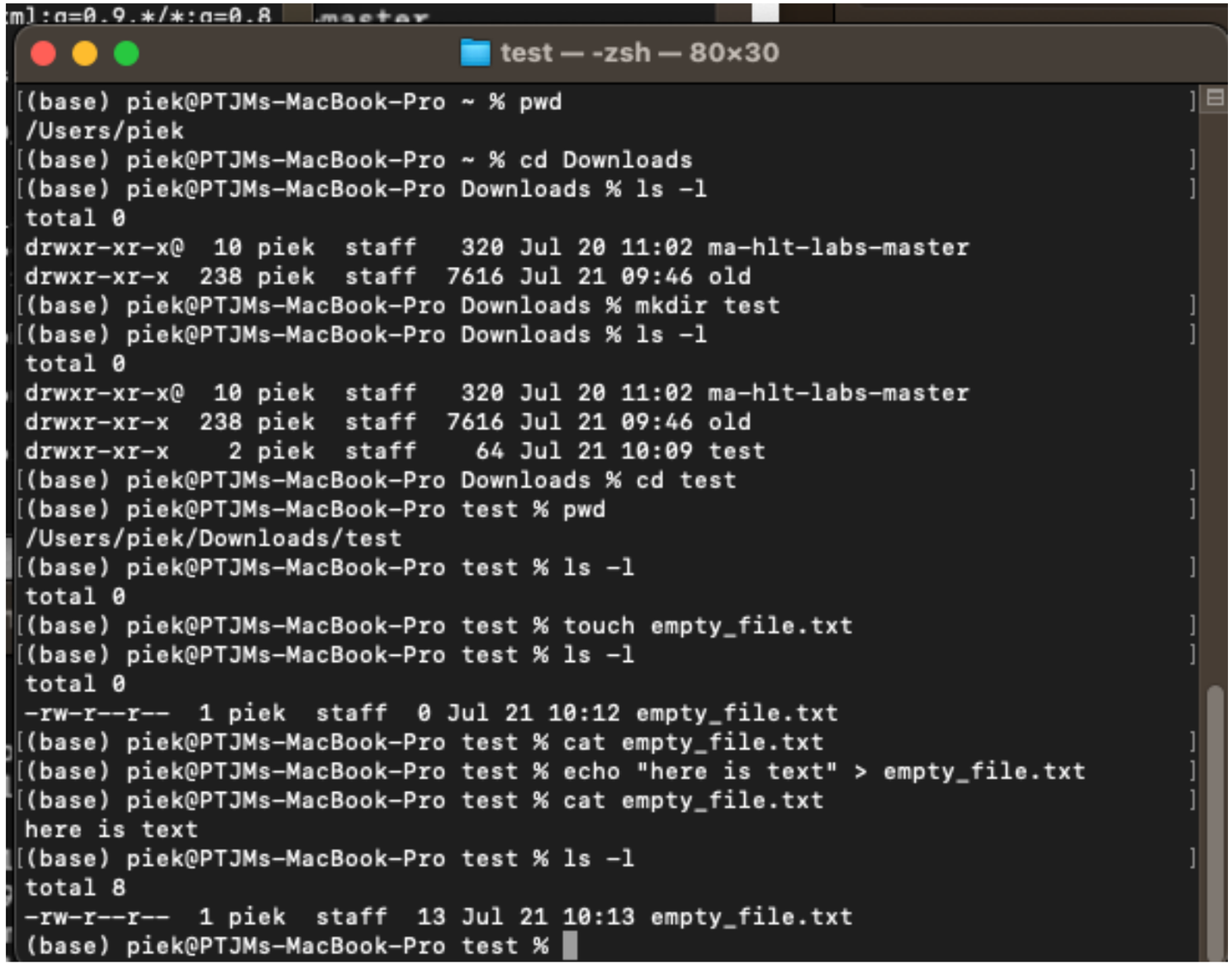
```
PycharmProjects
(base) piek@PTJMs-MacBook-Pro ~ % cd Downloads
(base) piek@PTJMs-MacBook-Pro Downloads % pwd
/Users/piek/Downloads
(base) piek@PTJMs-MacBook-Pro Downloads % ls
ma-hlt-labs-master      old
(base) piek@PTJMs-MacBook-Pro Downloads % cd ma-hlt-labs-master
(base) piek@PTJMs-MacBook-Pro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

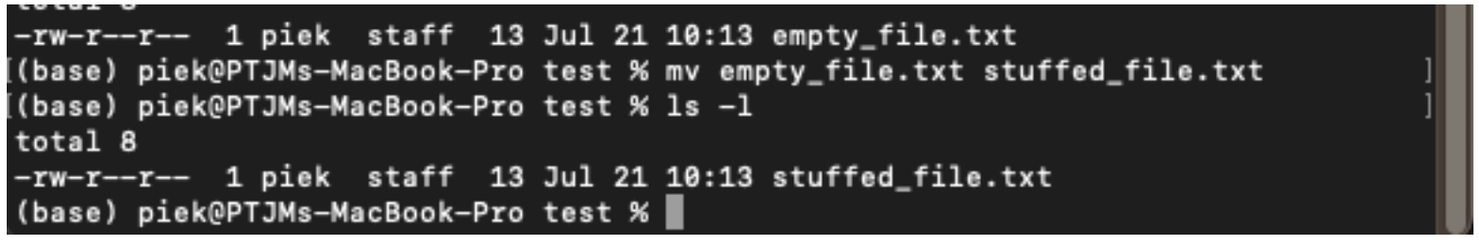
```
drwxr-xr-x@ 15 piek  staff   480 Jul 20 11:02 lab5.final_assignment
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % pwd
/Users/piek/Downloads/ma-hlt-labs-master
(base) piek@PTJMs-MacBook-Pro ma-hlt-labs-master % cd ..
(base) piek@PTJMs-MacBook-Pro Downloads % pwd
/Users/piek/Downloads
(base) piek@PTJMs-MacBook-Pro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

Preparations

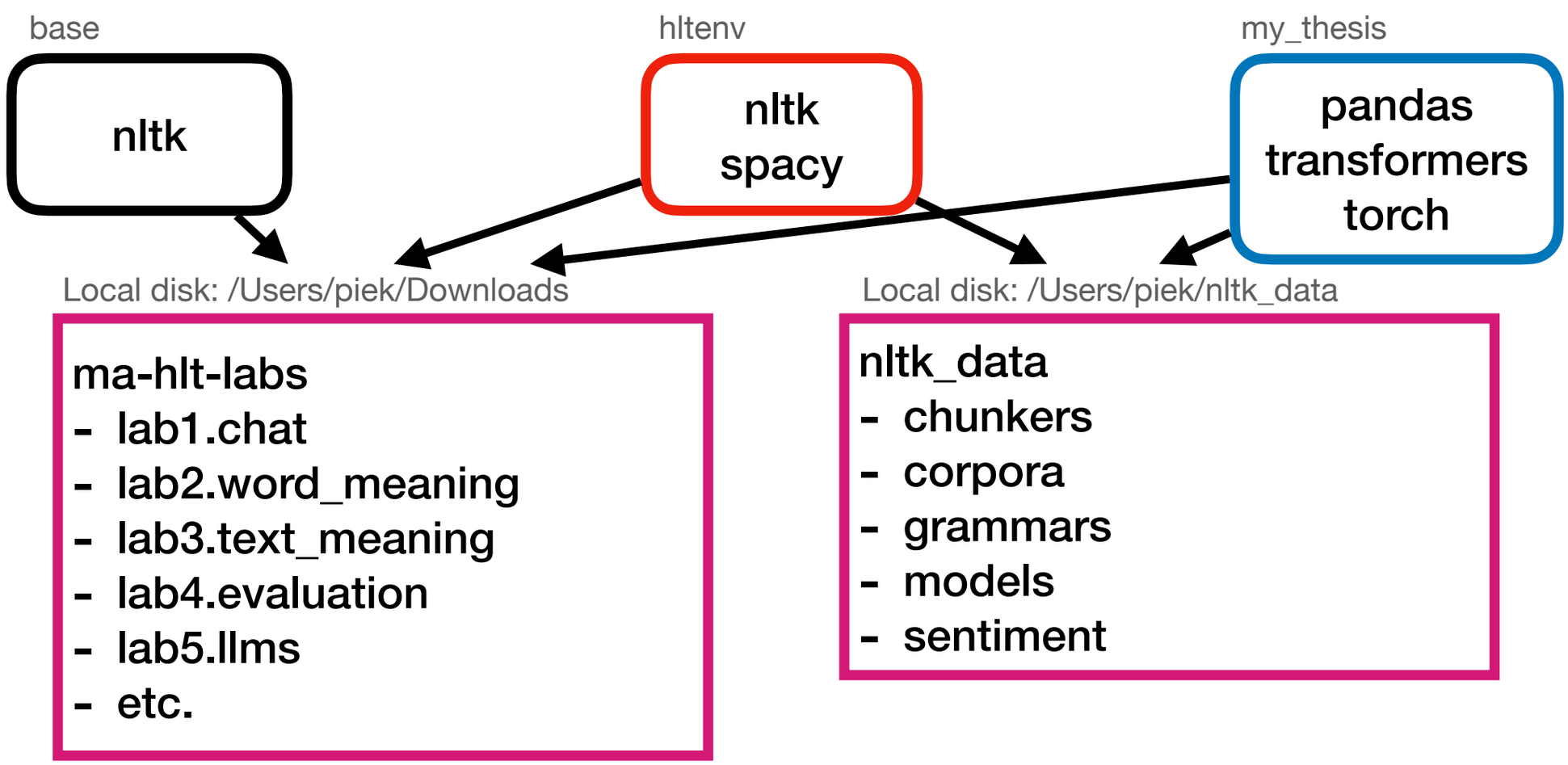
Text editors

- 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/

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 and install and run your code within



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":

```
>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.

Activating 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":
    >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. Activate this environment:

    # On Mac/Linux
    >source hltenv/bin/activate

    # On Windows:
    >hltenv\Scripts\activate.bat

After activation, your prompt is prefixed with (hltenv)

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

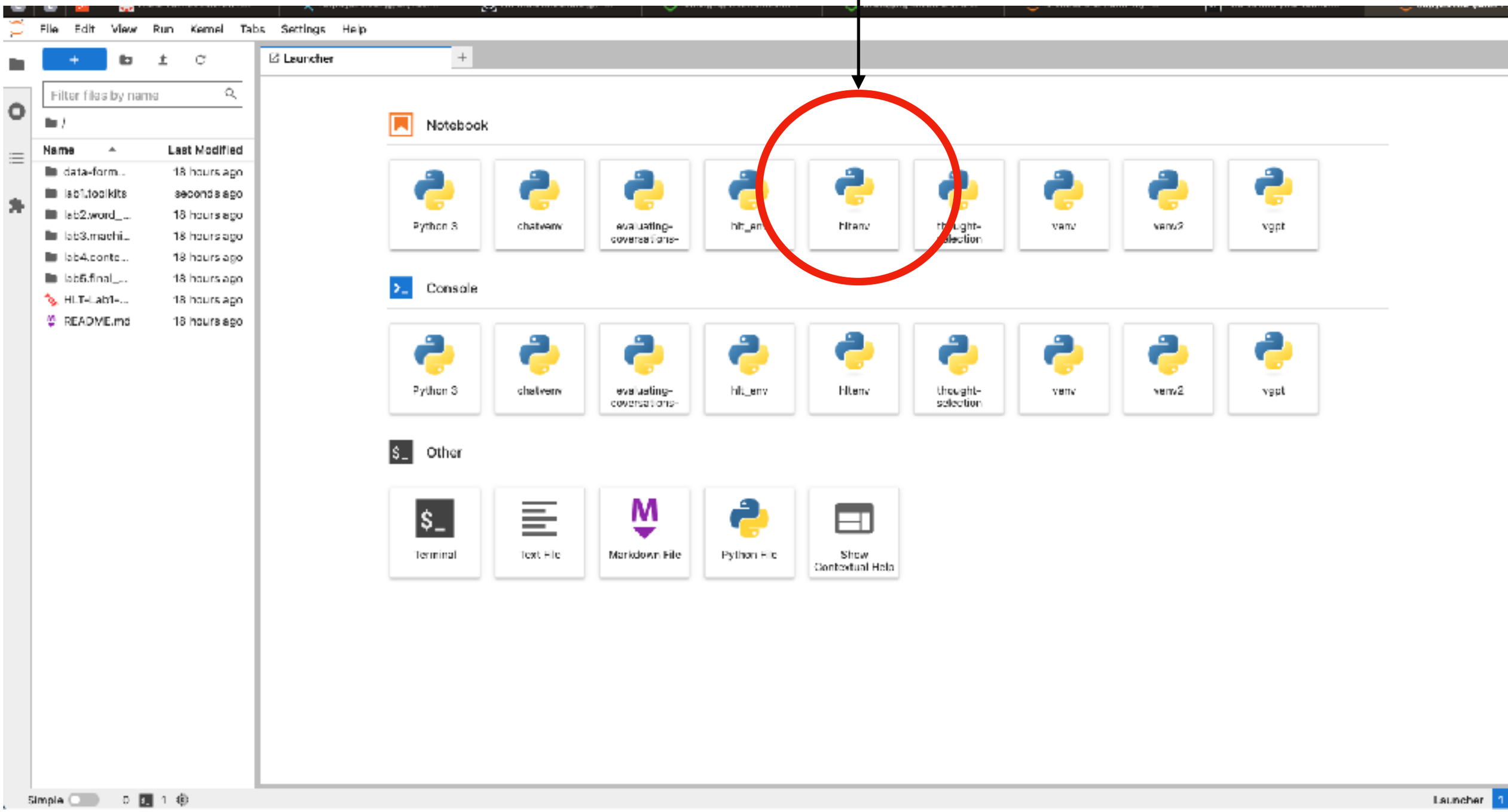
Open a terminal on your local machine and navigate to the folder where you have unpacked the Github repository for **cltl/ma-hlt-labs**.

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:

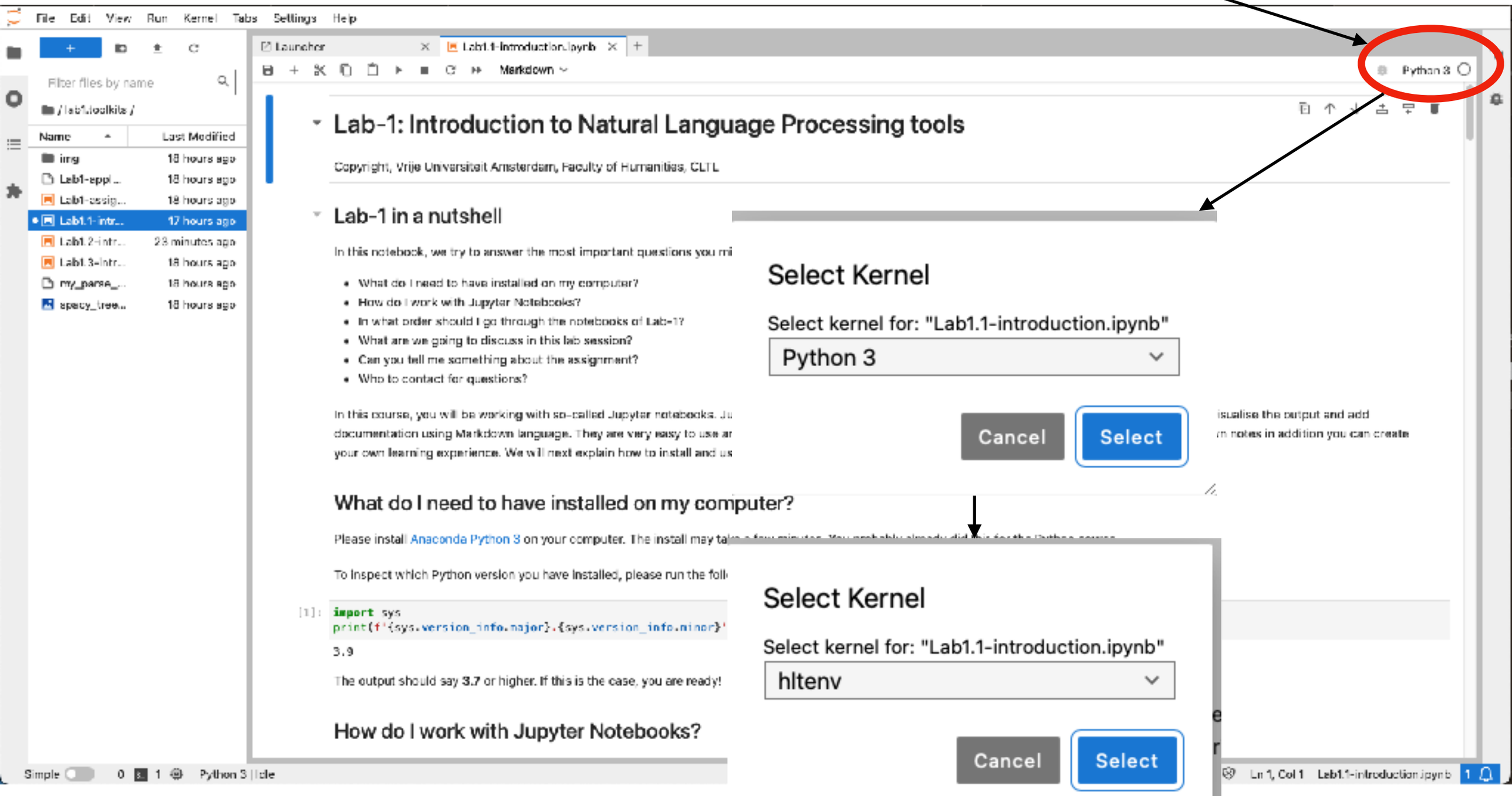
```
1. Create a virtual environment with the name "hltenv":
    >python –m venv hltenv

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

2. Activate this environment:

    # On Mac/Linux
    >source hltenv/bin/activate

    # On Windows:
    >hltenv\Scripts\activate.bat

After activation, your prompt is prefixed with (hltenv)

3. Upgrade pip to the latest version:
    (hltenv)>python –m pip install ––upgrade pip

4. Install Jupyter in this environment:
    (hltenv)>pip install jupyter
    #### making sure jupyter runs in the venv
    (hltenv)>pip install ipykernel
    (hltenv)>python –m ipykernel install ––user ––name=hltenv
```

Make sure Jupyter uses my hltenv

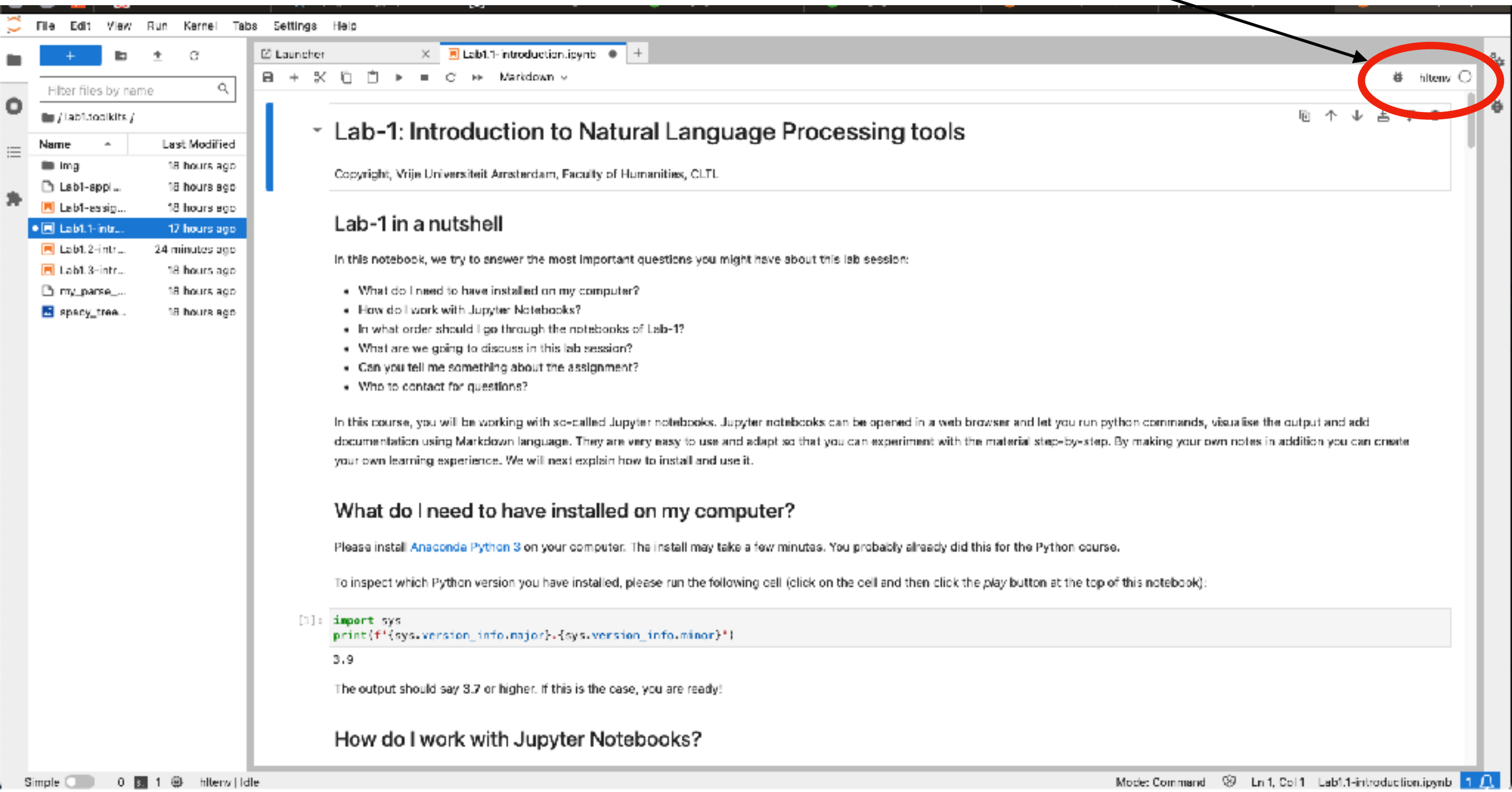
virtual environment is available



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



This notebook now uses “hltenv”



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:

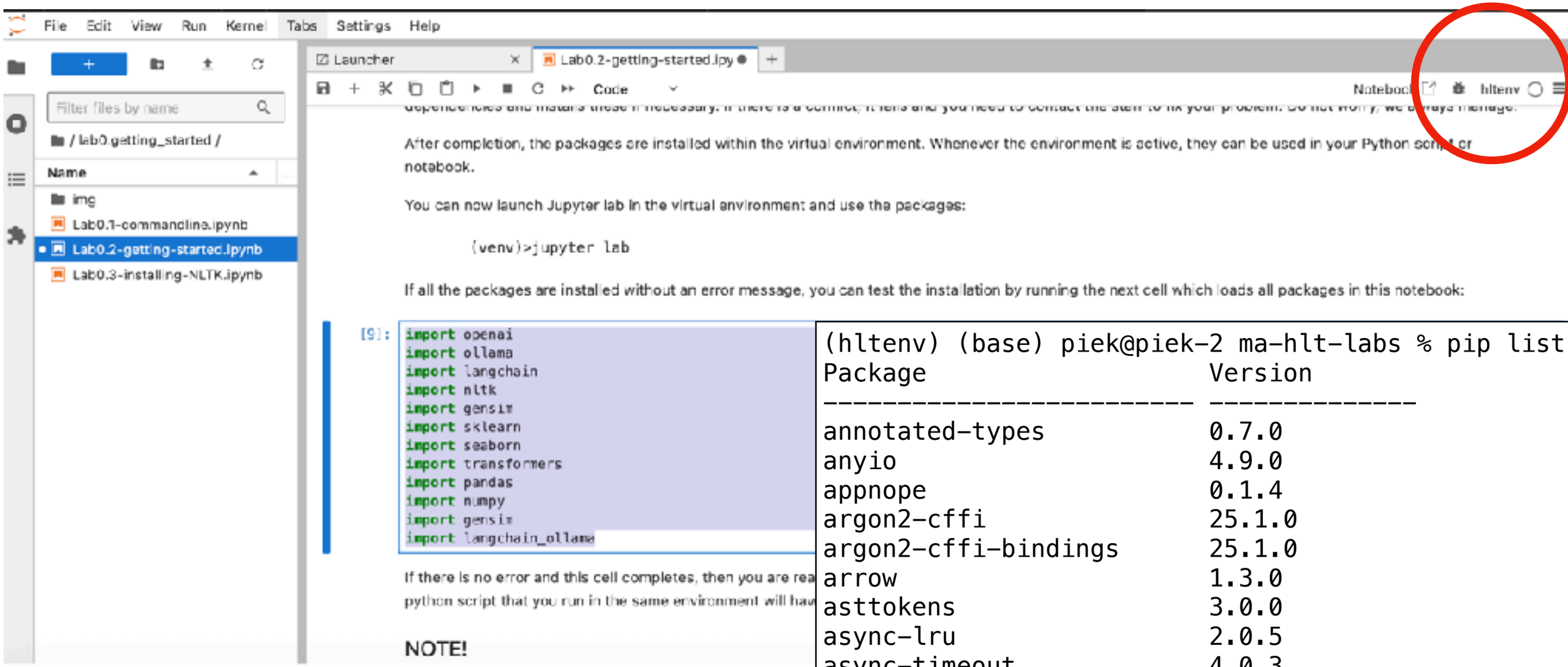
```
(hltenv)pip install –r requirements.txt
```

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)>jupyter lab
```

Installing packages in a Python virtual environment




```
(hltenv) (base) piek@piek-2 ma-hlt-labs % pip list
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
```