This guide is meant for a direct and easy access to the DTU HPC to run Python (XYZ) scripts on Windows. For the Official Homepage for HPC support visit:

<https://www.hpc.dtu.dk/?page_id=3226>

(In my experience, Google is better at answering your questions about the HPC at DTU than searching around on the support page)

There are two main ways of accessing the cluster, through a graphical user interface and the terminal on your PC. Using the graphical user interface is only an advantage if you want to use the HPC for smaller scripts and editing them, because you’ll be using a terminal within it in order to submit scripts to the truly powerful GPUs. Skip to the SSH section if you intend to use the HPC through the terminal

# 

[**Initial Set-up 2**](#_xwtnjdz212ht)

[GUI 2](#_11o4dwabgla)

[SSH (Windows) 6](#_dsd8otjglciw)

[**Running Scripts 6**](#_43nsyjfo48vi)

[Python 6](#_rzwfcxo27dvj)

[Set-up 6](#_ayraunclm6rc)

[Do not load scipy as a way to save space, it may cause package conflicts…. 9](#_5blytfuzbcpv)

[Run 10](#_pgta6g2mwcuy)

[Run Jupyter Notebook 12](#_4d8apg3vdxc1)

[Clean up 12](#_on7p707avrod)

[R 12](#_j4eib9aujvsz)

[Installing packages 12](#_ha3dp3g4wd0q)

[Matlab 14](#_ftz6hpr1s6cl)

# 

# 

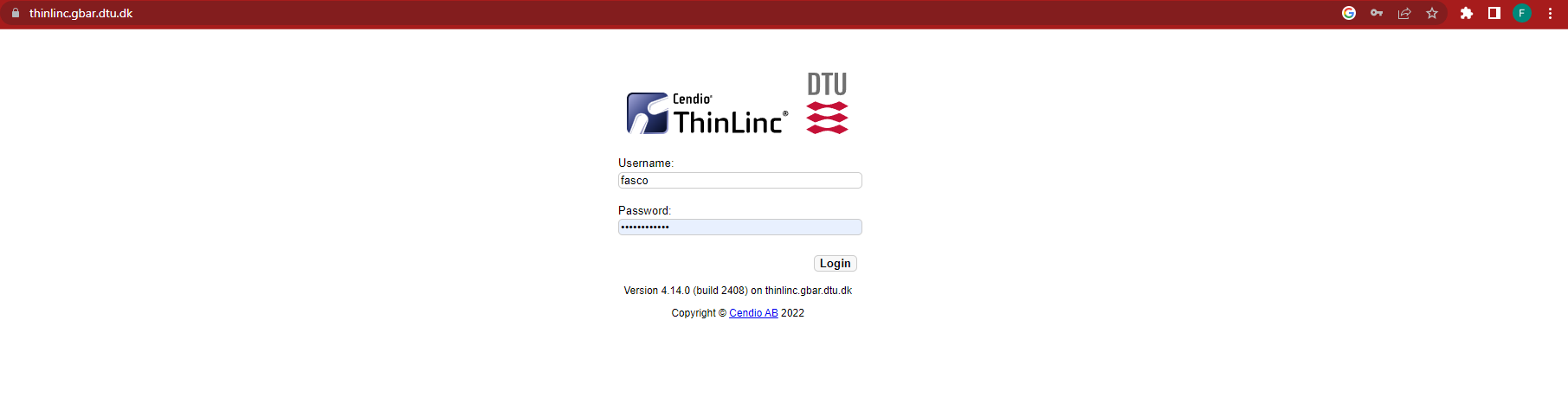
# Initial Set-up

## ThinLinc (GUI)

To access the HPC Cluster, use the following link in any browser:

<https://thinlinc.gbar.dtu.dk/>

The page should look something like this:



Log in using your credentials (without ‘dtu.dk’ at the end). Now the interface will look as follows:



If you want to use a Desktop app instead of a browser, Thinlinc can be downloaded at <https://www.cendio.com/thinlinc/download>, Log in:

To run a script on the High performance chips available, we first need to upload the script. This can either be done through copying and pasting, or, more easily through WinSCP.

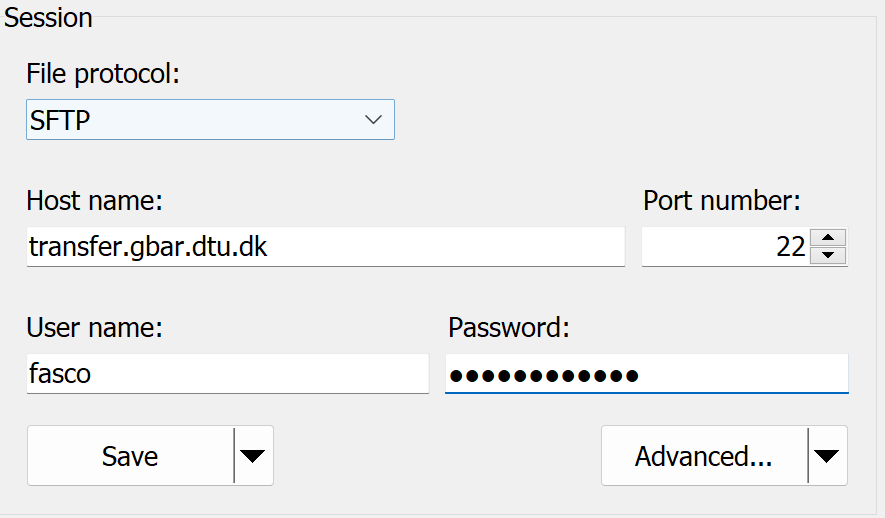
Go to:

<https://winscp.net/eng/download.php>

And press the green Download button:

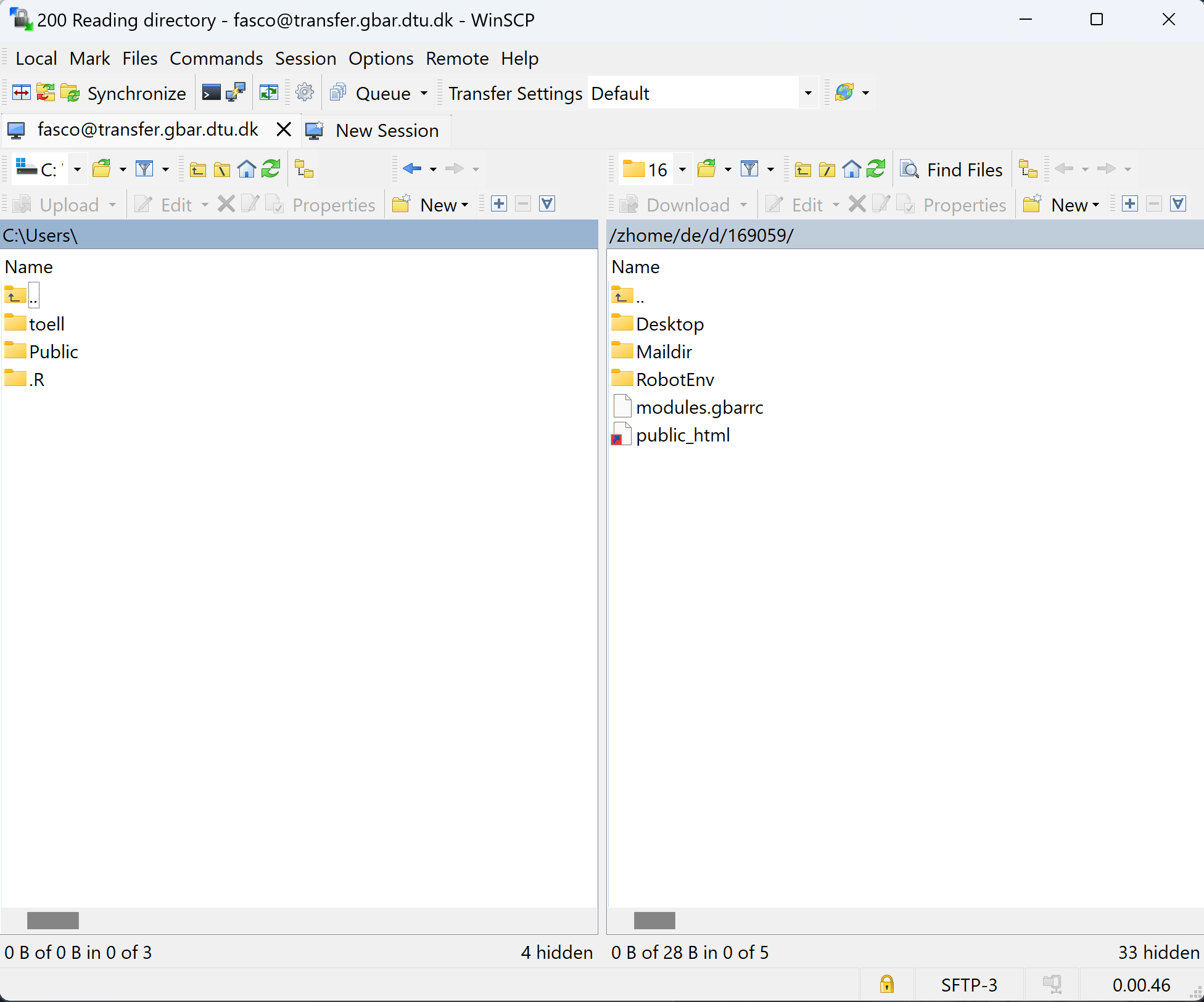


Now run the setup.exe file from the download and follow the instructions. Upon opening WinSCP after download, you will be faced with the following log in page:

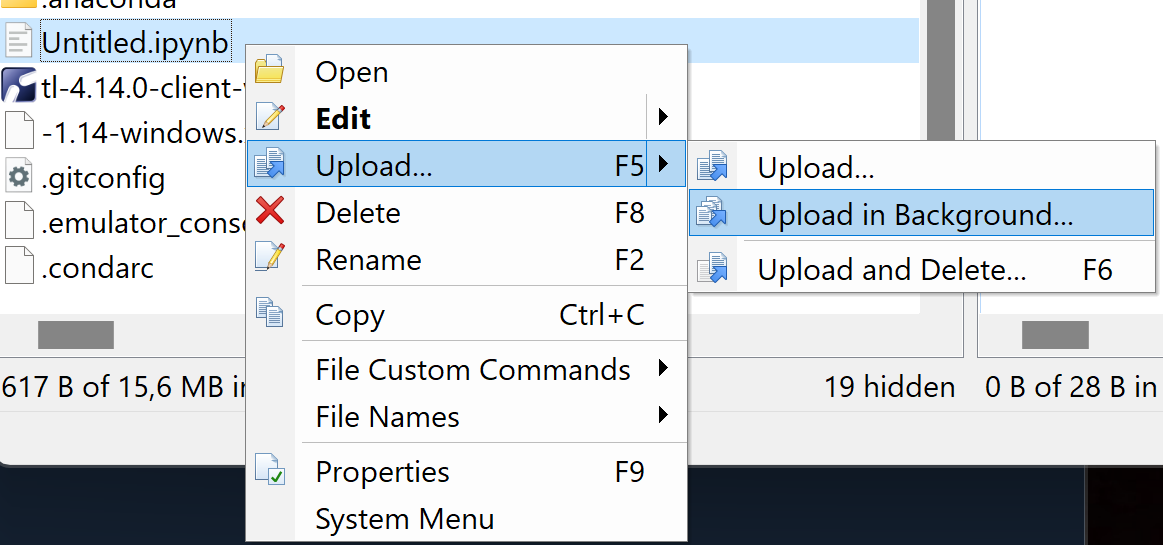


Use host name ‘transfer.gbar.dtu.dk’ and dtu credentials to start a session.

Upon a successful login, you should see a screen like the following:

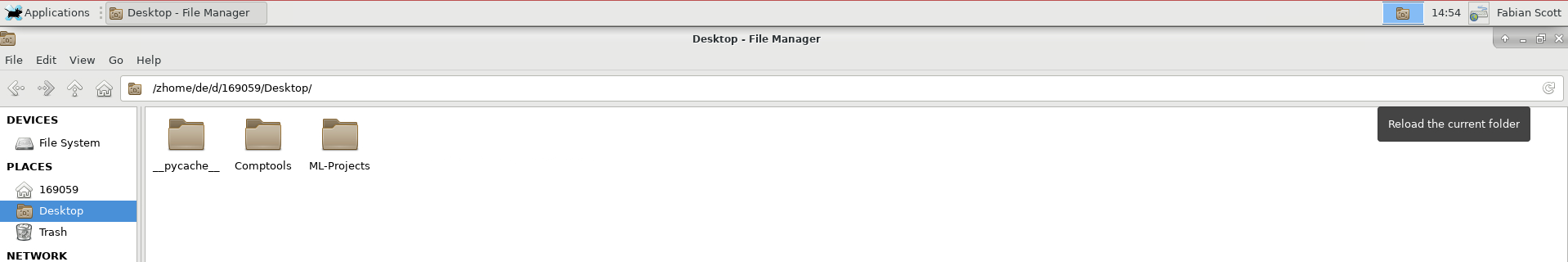


The folders on the left are from your PC and those on the right are your workspace on the DTU HPC, the exact same as found on ThinLinc. By navigating to a file/folder on your PC, you can upload it by right-clicking and choosing upload/upload in background:



I recommend ‘upload in background’, as default upload prevents you from doing anything else in the app until the upload is finished.

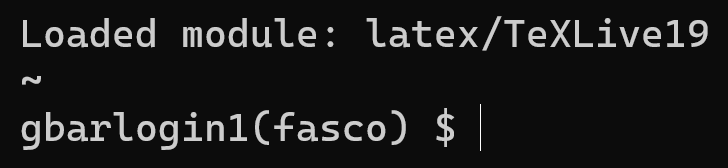
Once you have uploaded a file, it will appear in Thinlinc as soon as you reload the folder in which it was uploaded (Using the button in the top right of the file manager):



## SSH (Windows)

For those using a unix (MAC or Linux) system, you can probably get it working by skipping 1 and 2 as these steps are windows specific.

1. Open a terminal (with PowerShell) as an Administrator and install an shh client, for instance using:
   1. Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH\*'
2. If login in on a network outside of DTU, set up the VPN using this link:
   1. <https://www.hpc.dtu.dk/?page_id=4317>
3. To log into the HPC cluster use the command, where ‘userid’ is your DTU username:
   1. ssh [userid@login1.gbar.dtu.dk](mailto:userid@login1.gbar.dtu.dk)
4. Next type in your password and you should see something similar to the following in your input field:



1. Now you are in the terminal, where you can use the commands found in the following sections of this guide.

# Running Scripts

## Python

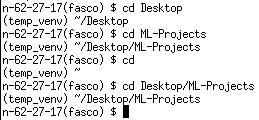
The following uses the terminal in ThinLinc, however the same commands work when you have used SSH to access the HPC.

### Set-up

For Python, it is recommended you use a virtual environment to run scripts in. To set up a virtual environment. Firstly, open the Terminal Emulator

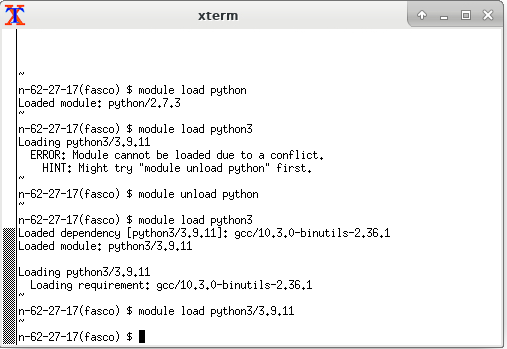


First thing to do is place yourself in the folder containing the files you’re working with:

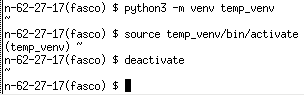


Load the python version you are working with

(note that the following screenshot also shows the necessity of specifying ‘python3’ instead of just ‘python’ and how to unload a module in the terminal):



After loading python3, specify the name of the virtual environment you want to create and activate it using the following commands. Notice how the ‘~’ between lines has now got the virtual environment name next to it. To deactivate the virtual environment, simply type ‘deactivate’:



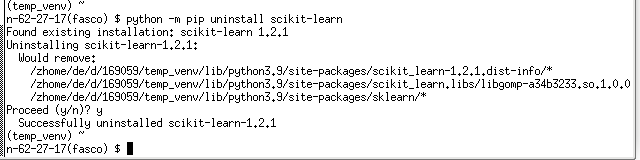
Tip: by pressing the upwards facing arrow key, you can go backwards through the commands you have written. Here I use it to reactivate the environment to continue the guide.

With the environment active, you can install Python packages using the command

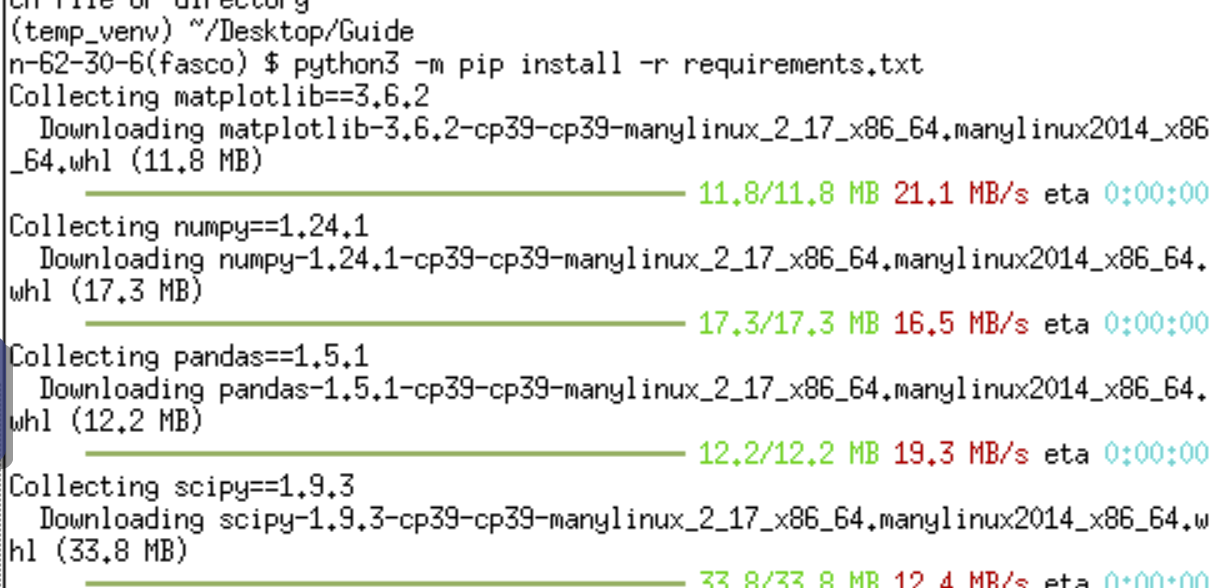
‘python -m pip install packagename’:



And the same command structure to uninstall, there is a prompt which will ask you to type ‘y’ to confirm:



In many projects, there will be a requirements.txt file. To install via this file do ‘python3 -m pip install -r requirements.txt’ This requires you to be in the same folder as the file:

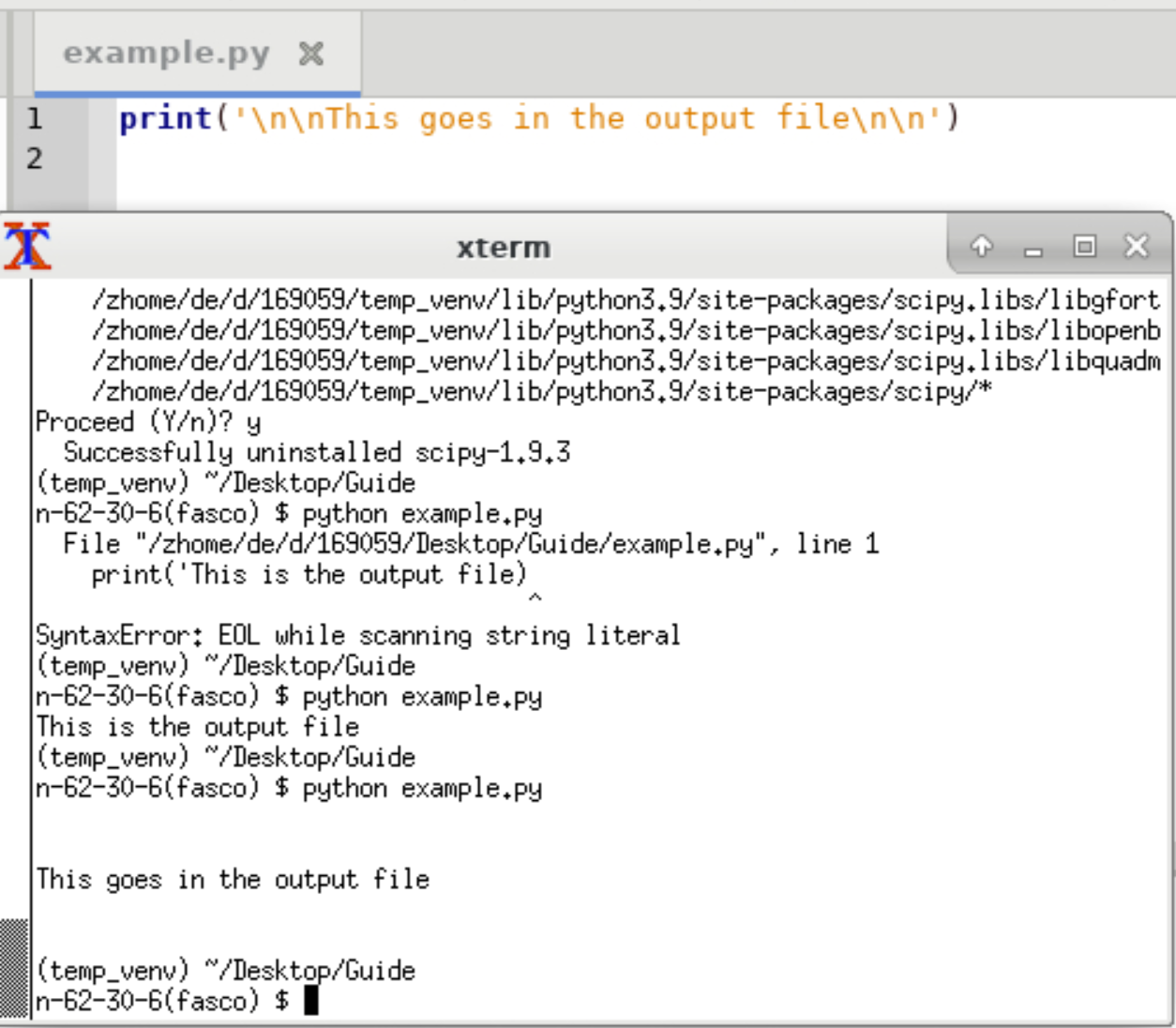


## 

## Do not load scipy as a way to save space, it may cause package conflicts….

### Run

Using the ‘cd’ command, place yourself in the directory where the script is located. Now you can use the command ‘python filename.py’ to run the script directly in the terminal:



Now it is time to make the .sh script which will execute the code on the HPC. The basic structure looks like this is found in the ‘submit.sh’ file in the folder this document is int. Here the default queue is used

### (GPU)

For an overview of the GPUs available see:

<https://www.hpc.dtu.dk/?page_id=2759>

If using pytorch install and load the appropriate modules by using the command found on this page:

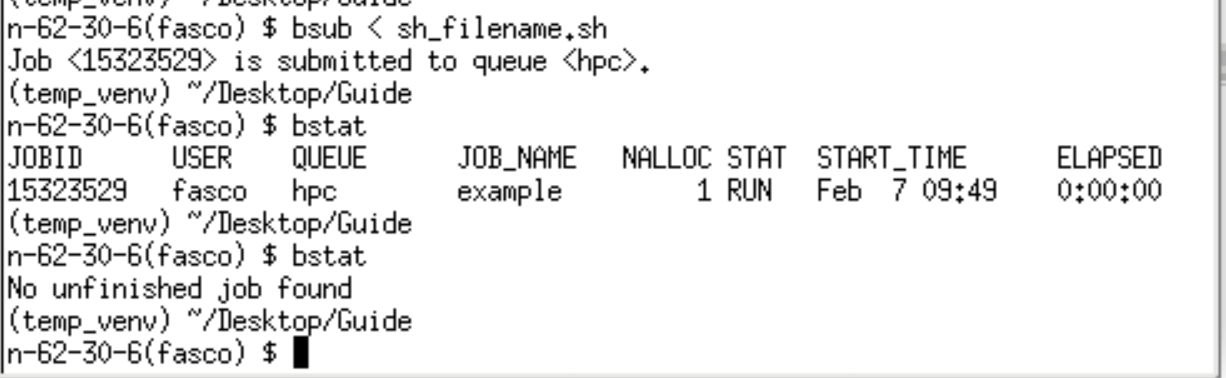
<https://pytorch.org/get-started/locally/>

Remember that the HPC uses Linux. The provided python file and requirements file should get a neural network to work and train on a GPU.

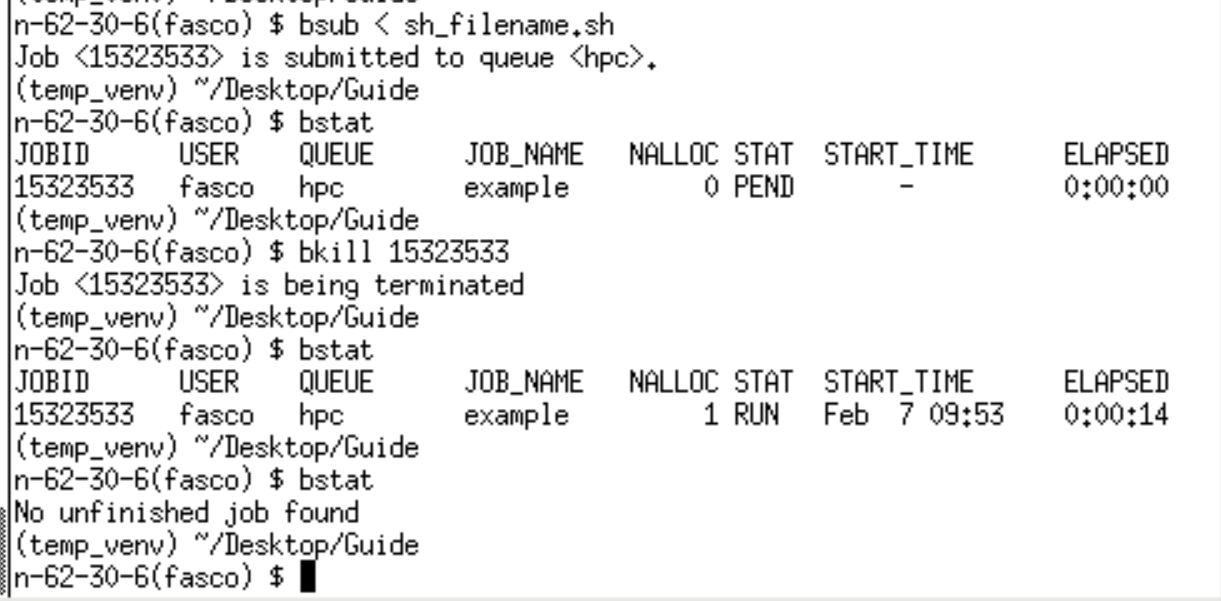
Once you have the .sh and .py in the same directory as you have placed yourself in in the terminal, use the command

bsub < sh\_filename.sh.

The code will now run and all outputs (mostly print statements) will appear in the output file, all error messages will appear in the error file. To check the status of you jobs use ‘bstat’:



To stop a job that is running, use ‘bkill jobid’. Notice that the jobid is found in the output from ‘bstat’. To keep an overview over different jobs, give them different, but short names, as the terminal can only display a limited number of characters.



The script is run in the same folder, so it is possible to read data and write data to files in the same way as normal. I recommend testing by running the script directly in the terminal, since if it works there it will (likely) work on the cluster as well.

### Run Jupyter Notebook

Install jupyter and notebook

python3 -m pip install jupyter

python3 -m pip install notebook

Then run the notebook with:

jupyter nbconvert –execute notebook\_name to notebook

This creates a copy of the notebook where it has tried to run all cells, notice that you should not specify the file format of ‘.ipynb’.

Module load matplotlib/3.5.1-numpy-1.22.2-python-3.9.11

### Clean up

Use the following command to remove the pip cache and save space:

rm -rvf ~/.cache/pip/http/\*

## R

### Installing packages

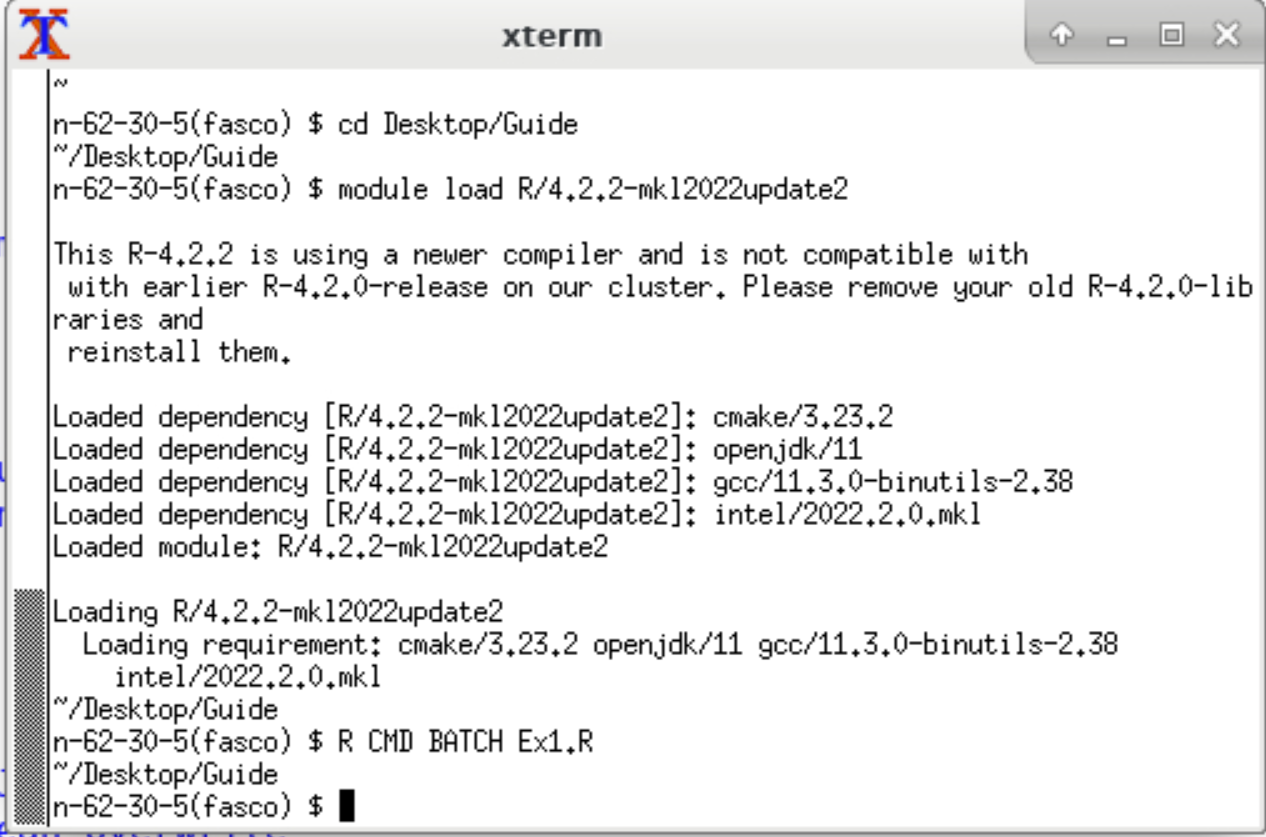
First step is to find the tar.gz files for the packages you wish to install. They will most likely end up in your downloads folder. With this file, load R using ‘module load R’ or specify the version you want from the list of available ones using ‘module avail R’.

Once loaded, use the following command to install the package into the directory of you choice:

R CMD INSTALL -l PATH/TO/DESTINATION /zhome/de/d/PATH/TO/DOWNLOADS

The dependencies of R packages may require some time to ensure that the package you want is actually installed.

To test whether your script will run on the HPC, load the version of R you are currently using ‘module load R/a.b.c-mkl-yyyyupdatec’, place yourself in the correct folder and use the command ‘R CMD BATCH script\_name.R’ to run the script:



This will create a file called ‘script\_name.Rout’ with the output along with a file called ‘Rplots.pdf’. These contain the output and the plots respectively. To run the same script on the HPC, use a similar script to that for Python, simply replacing the final commands with those for running R scripts in the terminal:

#!/bin/sh

### General options

### -- specify queue --

#BSUB -q hpc

### -- set the job Name --

#BSUB -J Job\_Name

### -- ask for 1 core --

#BSUB -n 1

### -- specify that we need 2GB of memory per core/slot --

#BSUB -R "rusage[mem=2GB]"

### -- specify that we want the job to get killed if it exceeds 3 GB per core/slot --

#BSUB -M 3GB

### -- set walltime limit: hh:mm --

#BSUB -W 24:00

### -- set the email address --

# please uncomment the following line and put in your e-mail address,

# if you want to receive e-mail notifications on a non-default address

##BSUB -u your\_email\_address

### -- send notification at start --

#BSUB -B

### -- send notification at completion --

#BSUB -N

### -- Specify the output and error file. %J is the job-id --

### -- -o and -e mean append, -oo and -eo mean overwrite --

#BSUB -o Output\_%J.out

#BSUB -e Output\_%J.err

module load R/4.2.2-mkl2022update2

R CMD BATCH My\_serial\_script.R

This will run the script in the same way as in the terminal, creating the same output files.

## Matlab

With matlab, you have to manually save all the plots you wish to produce along with the data as it does not do this automatically.

To run a script, the command is ‘matlab -nodisplay -batch script\_name’.

NOTE: Do not include the .m at the end of the filename.

Test if you can run it directly from the terminal:



If it works, simply put this line into the .sh script and ‘bsub’ it in the same manner as the other languages:

#!/bin/sh

### General options

### -- specify queue --

#BSUB -q hpc

### -- set the job Name --

#BSUB -J Job\_Name

### -- ask for 1 core --

#BSUB -n 1

### -- specify that we need 2GB of memory per core/slot --

#BSUB -R "rusage[mem=2GB]"

### -- specify that we want the job to get killed if it exceeds 3 GB per core/slot --

#BSUB -M 3GB

### -- set walltime limit: hh:mm --

#BSUB -W 24:00

### -- set the email address --

# please uncomment the following line and put in your e-mail address,

# if you want to receive e-mail notifications on a non-default address

##BSUB -u your\_email\_address

### -- send notification at start --

#BSUB -B

### -- send notification at completion --

#BSUB -N

### -- Specify the output and error file. %J is the job-id --

### -- -o and -e mean append, -oo and -eo mean overwrite --

#BSUB -o Output\_%J.out

#BSUB -e Output\_%J.err

Matlab -nodisplay -batch script\_name