



Using Python @ NSC

OS Python

- NSC's clusters have the Rocky 9 standard Python installed.

```
[x_abcde@tetralith]$ module purge
[x_abcde@tetralith]$ which python
/usr/bin/python
[x_abcde@tetralith]$ python --version
Python 3.9.21
```

- Recommendation:
 - don't use it
 - okej, maybe use it for simple python tasks that do not have any difficult dependencies.

Python 2: please try to
upgrade



Python modules



Python modules: 3 flavors

1. Python/<version>-**env**-hpcX-<toolchain>
2. Python/<version>-**bare**-hpcX-<toolchain>
3. “Miniforge/<version>-hpcX”

```
[x_abcde@tetralith2 ~]$ module avail python/
```

```
----- /software/sse2/tetralith_el9/modules -----  
Python/recommendation          (D)  Python/3.10.4-bare-hpc1-gcc-2022a-eb  
Python/2.7.18-bare-hpc1-gcc-2022a-eb  Python/3.10.4-env-hpc1-gcc-2022a-eb  
Python/3.10.4-env-hpc2-gcc-2022a-eb   Python/3.11.5-bare-hpc1-gcc-2023b-eb  
Python/3.11.5-env-hpc1-gcc-2023b-eb
```

Where:

D: Default Module

Python/<version>-env-hpcX-<toolchain>-eb

- After loading a Python module, you will have a new Python installation in your PATH
- Provides a specific python version and a fairly extensive range of scientific packages e.g. NumPy, SciPy, Matplotlib, Pandas, etc.

```
[x_abcde@tetralith]$ module load Python/3.11.5-env-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ which python
/software/sse2/tetralith_el9/easybuild/pure/software/Python/3.11.5-GCCcore-13.2.0/bin/python
[x_abcde@tetralith]$ python
Python 3.11.5 (main, Nov 18 2024, 16:41:22) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy
>>> numpy.linspace(0, 2, 9)
array([0.   , 0.25, 0.5  , 0.75, 1.   , 1.25, 1.5  , 1.75, 2.   ])
```

Python/<version>-env-hpcX-<toolchain>-eb

- Useful for python scripts that need extra packages but are not picky with the precise versions.
- Use “pip list” to give a listing of the installed packages (including versions)

```
[x_abcde@tetralith]$ module load Python/3.11.5-env-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ pip list --format=columns | grep -i scipy
SciPy                                1.11.4

[notice] A new release of pip is available: 23.2.1 -> 25.0.1
[notice] To update, run: pip install --upgrade pip
```

Managing your python environment



Managing your python environment

- Python **env** modules provide a set of common python packages.
- For technical reasons, we cannot install all the packages that everyone needs in the same module installation.
- Instead, we recommend that you install extra packages in your own user space using a **managed environment** .
- We support two options:
 1. Virtualenv
 2. Miniforge

Managing your python environment: virtualenv

- Recommend using e.g. `Python/3.11.5-bare-hpc1-gcc-2023b-eb` for managing environments using virtualenv

```
[x_abcde@tetralith]$ module load Python/3.11.5-bare-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ python -m venv myownvirtualenv
[x_abcde@tetralith]$ source myownvirtualenv/bin/activate
(myownvirtualenv)[x_abcde@tetralith]$ python -m pip install python-hostlist
...
(myownvirtualenv)[x_abcde@tetralith]$ python
Python 3.11.5 (main, Nov 18 2024, 16:41:22) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hostlist
>>> hostlist.__file__
'/home/x_abcde/myownvirtualenv/lib/python3.11/site-packages/hostlist.py'
```

Python/<version>-bare-hpcX-<toolchain>-eb

- Provide a specific python version without any preinstalled packages
- “bare” modules are primarily for use with Python venvs, so you can install packages using the pip command. This allows requesting precise versions of desired package(s).

```
[x_abcde@tetralith]$ module load Python/3.11.5-bare-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ python -m venv numpyscipy.venv
[x_abcde@tetralith]$ source numpyscipy.venv/bin/activate
(numpyscipy.venv) [x_abcde@tetralith]$ python -m pip install numpy==1.26.4 scipy==1.11.4
...
Installing collected packages: numpy, scipy
Successfully installed numpy-1.26.4 scipy-1.11.4
(numpyscipy.venv) [x_abcde@tetralith]$
```

Installing packages that require compiling

- If you need to install a pip package that requires **compiling** , then we strongly recommend using a virtualenv.
- Load the corresponding buildenv module (e.g. buildenv-gcc/2023b-eb)
- Create a virtual environment for building and adding packages.

```
[x_abcde@tetralith]$ module load Python/3.11.5-bare-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ module load buildenv-gcc/2023b-eb
[x_abcde@tetralith]$ python -m venv myownvirtualenv
[x_abcde@tetralith]$ source myownvirtualenv/bin/activate
(myownvirtualenv) [x_abcde@tetralith]$ ...
```

- For more details (section “Pip packages that require compilation”)

Managing your python environment: conda/Miniforge

- Use a Miniforge module for managing your conda environments

```
[x_abcde@tetralith]$ module load Miniforge/24.7.1-2-hpc1
[x_abcde@tetralith]$ conda create -n myownenv python=3.8 pandas seaborn
...
[x_abcde@tetralith]$ conda activate myownenv
(myownenv)[x_abcde@tetralith]$ which python
~/.conda/envs/myownenv/bin/python
(myownenv)[x_abcde@tetralith]$ python
Python 3.8.20 | packaged by conda-forge | (default, Sep 30 2024, 17:52:49)
[GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
>>> pandas.__version__
'2.0.3'
```

Managing your python environment: conda/Miniforge

- Miniforge modules are available as a “drop-in” replacement for Anaconda.
- The `mamba` command can be used in place of `conda` (and in general performs better)
- By default, miniforge uses the conda-forge community-driven library of packages instead of channels offered by the Anaconda company
- By default, your conda environments are installed in `${HOME}/.conda`. If you have multiple conda environments here there is a risk of filling up your `${HOME}` space.
 - There are ways to install conda environments outside `${HOME}`, see: Python @ NSC
- More detailed information: Anaconda @ NSC

jupyter notebooks

The Jupyter logo is centered on the slide. It features the word "jupyter" in a dark grey, lowercase, sans-serif font. The text is enclosed within a large, stylized orange smiley face. The smiley face is composed of two thick, curved orange lines forming the upper and lower arcs, and two solid grey circles representing the eyes. The entire logo is set against a white background.

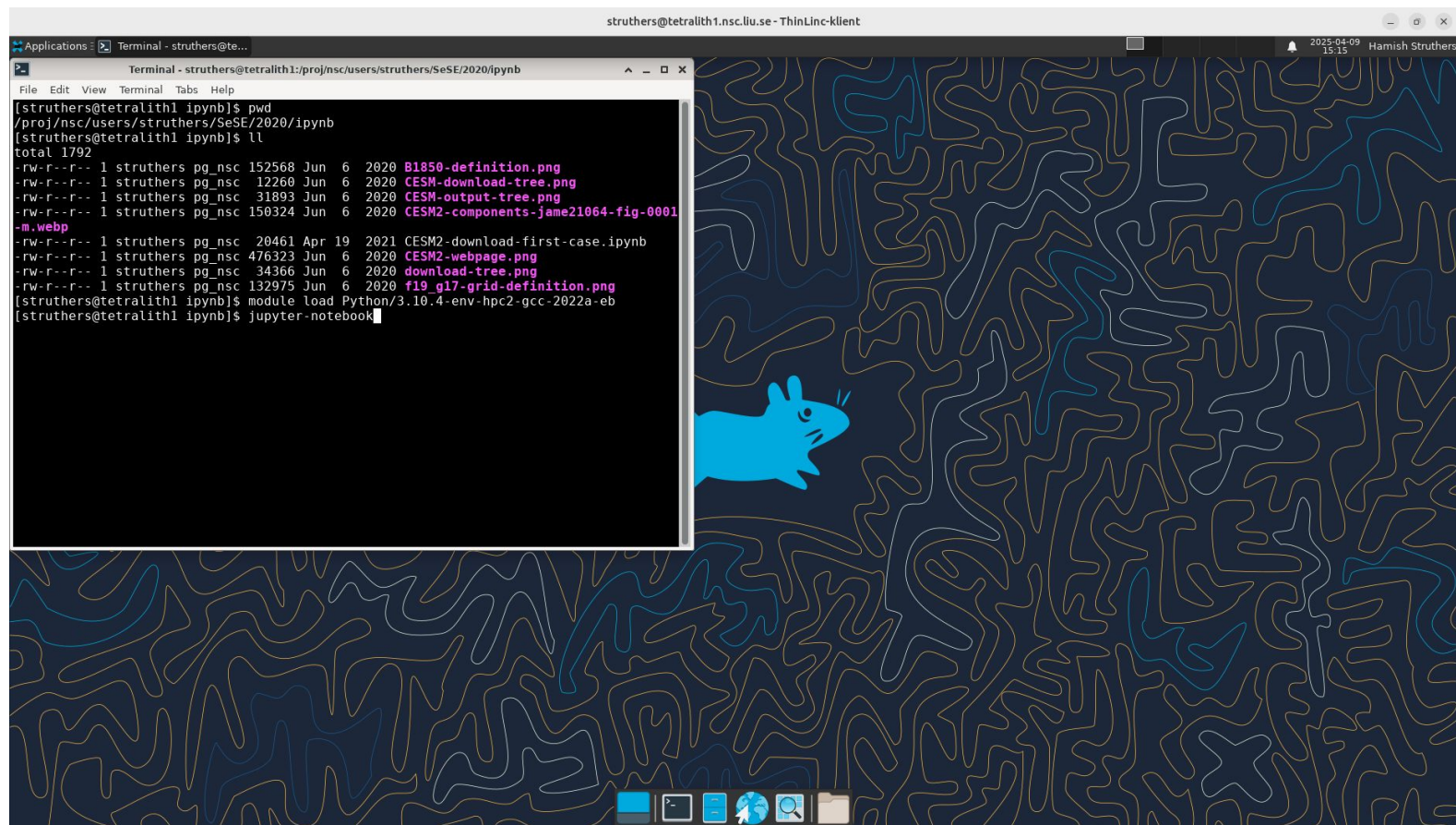
jupyter

Jupyter notebooks

- Jupyter notebooks can be run on with the login nodes or compute nodes.
- Jupyter packages are included in the `Python/3.11.5-env-hpc1-gcc-2023b-eb` module

```
[x_abcde@tetralith]$ module load Python/3.11.5-env-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ pip list | grep jupyter
...
```

- (or you can create and manage your own python environment that includes jupyter)



For light calculations, use jupyter in combination with thinlinc (on a login node).

The screenshot shows a desktop environment with a terminal window and a web browser displaying the Jupyter notebook interface.

Terminal Window: The terminal shows the user `struthers` at `tetralith1.nsc.liu.se` in a `ThinLinc-klient` session. The user is in a directory `struthers@tetralith1:~/proj/nsc/users/struthers/SeSE/2020/ipyb`. The terminal output shows the user running `module load jupyter-notebook` and `jupyter-notebook`. The Jupyter notebook server is started on port 8889. The terminal also shows the user's IP address `127.0.0.1` and the server's IP address `127.0.0.1`. The user is prompted to enter a token to access the notebook.

Web Browser: The browser shows the Jupyter notebook interface. The title bar indicates the user is logged in as `struthers` at `tetralith1.nsc.liu.se`. The browser address bar shows `localhost:8889/tree`. The Jupyter interface displays a list of files in the `/` directory:

Name	Last Modified	File size
☐ <code>CESM2-download-first-case.ipynb</code>	4 years ago	20.5 kB
☐ <code>B1850-definition.png</code>	5 years ago	153 kB
☐ <code>CESM-download-tree.png</code>	5 years ago	12.3 kB
☐ <code>CESM-output-tree.png</code>	5 years ago	31.9 kB
☐ <code>CESM2-components-jame21064-fig-0001-m.webp</code>	5 years ago	150 kB
☐ <code>CESM2-webpage.png</code>	5 years ago	476 kB
☐ <code>download-tree.png</code>	5 years ago	34.4 kB
☐ <code>ft19_g17-grid-definition.png</code>	5 years ago	133 kB

For light calculations, use jupyter in combination with thinlinc (on a login node).

Jupyter notebooks

- For more heavy jupyter notebook calculations: recommend using jupyter on a compute node (interactive session) and connecting via an ssh tunnel

mpi4py

- Python scripts that use mpi4py are supported by NSC's mpi launcher mpprun.
 - E.g. interactive:

```
[x_abcde@tetralith]$ module load Python/3.11.5-env-hpc1-gcc-2023b-eb
[x_abcde@tetralith]$ interactive -n 4 -A <my-project> -t 01:00:00
...
[x_abcde@n1234]$ mpprun python mpi4py-pythonscript.py
...
```

- where `mpi4py-pythonscript.py` includes, e.g.:
`import mpi4py as MPI`

mpi4py

- You can also run in batch mode using a script something like:

```
#!/bin/bash
#SBATCH -A naiss2025-x-yyy
#SBATCH -n 4
#SBATCH -t 01:00:00
#SBATCH -J jobname
```

```
module load Python/3.11.5-env-hpc1-gcc-2023b-eb
mpprun python mpi4py-pythonscript.py
```

Final notes:

- Maintain separate python environments for separate work tasks
- Please do NOT use `pip install --local`
- Remove `conda init` from your `.bashrc` (and try to keep changes to `.bashrc` to a minimum)