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 3.9.18
```

- Recommendation: don’t use it

Python modules

Python modules

- Python modules provide access to a recent version of Python + scientific libraries: e.g. NumPy, SciPy, Matplotlib, Pandas etc

```
[x_abcde@tetralith]$ module avail Python/
-------------------------------------------------
/software/sse2/tetralith_el9/modules
-------------------------------------------------
   Python/recommendation (D)  Python/2.7.18-bare-hpc1-gcc-2022a-eb
Python/3.10.4-env-hpc1-gcc-2022a-eb

  Where:
   D:  Default Module
```

Python 2: please try to upgrade

Python modules

- After loading a Python module, you will have a new Python installation in your PATH
- A range of scientific packages are also made available e.g. NumPy

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ which python
/software/sse2/tetralith_el9/easybuild/pure/software/Python/3.10.4-GCCcore-11.3.0/bin/python
[x_abcde@tetralith]$ python
     Python 3.10.4 (main, Oct  6 2023, 16:37:55) [GCC 11.3.0] 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 modules

Check available packages in a Python module

- To list the installed packages in a NSC build installation, simply load the module and run: `pip list`.
- If you are looking for a specific package, then pipe the output from `pip list` to `grep`:

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ pip list --format=columns | grep -i scipy
    SciPy                           1.8.1
```

Managing your python environment

Managing your python environment

- Python 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. anaconda
  2. virtualenv

Managing your python environment: conda

- Use an **Anaconda** module for managing your conda environments

```
[x_abcde@tetralith]$ module load Anaconda/2023.09-0-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 3.8.19 | packaged by conda-forge | (default, Mar 20 2024, 12:47:35)
     [GCC 12.3.0] on linux
     Type "help", "copyright", "credits" or "license" for more information.
     >>> import pandas
     >>> pandas.__version__
     '2.0.3'
```

Managing your python environment: conda

- 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
- Mambaforge modules are available on Tetralith/Sigma as a “drop-in” replacement to Anaconda

  `Mambaforge/23.3.1-1-hpc1`

Managing your python environment: virtualenv

- Use the **Python/3.10.4-env-hpc1-gcc-2022a-eb** module for managing environments using virtualenv

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ virtualenv --system-site-packages myownvirtualenv
[x_abcde@tetralith]$ source myownvirtualenv/bin/activate
(myownvirtualenv)[x_abcde@tetralith]$ pip install python-hostlist
(myownvirtualenv)[x_abcde@tetralith]$ python
     Python 3.10.4 (main, Oct  6 2023, 16:37:55) [GCC 11.3.0] on linux
     Type "help", "copyright", "credits" or "license" for more information.
     >>> import hostlist
     >>> hostlist.__file__
     '/home/x_abcde/myownvirtualenv/lib/python3.10/site-packages/hostlist.py'
```

Installing packages that require compiling

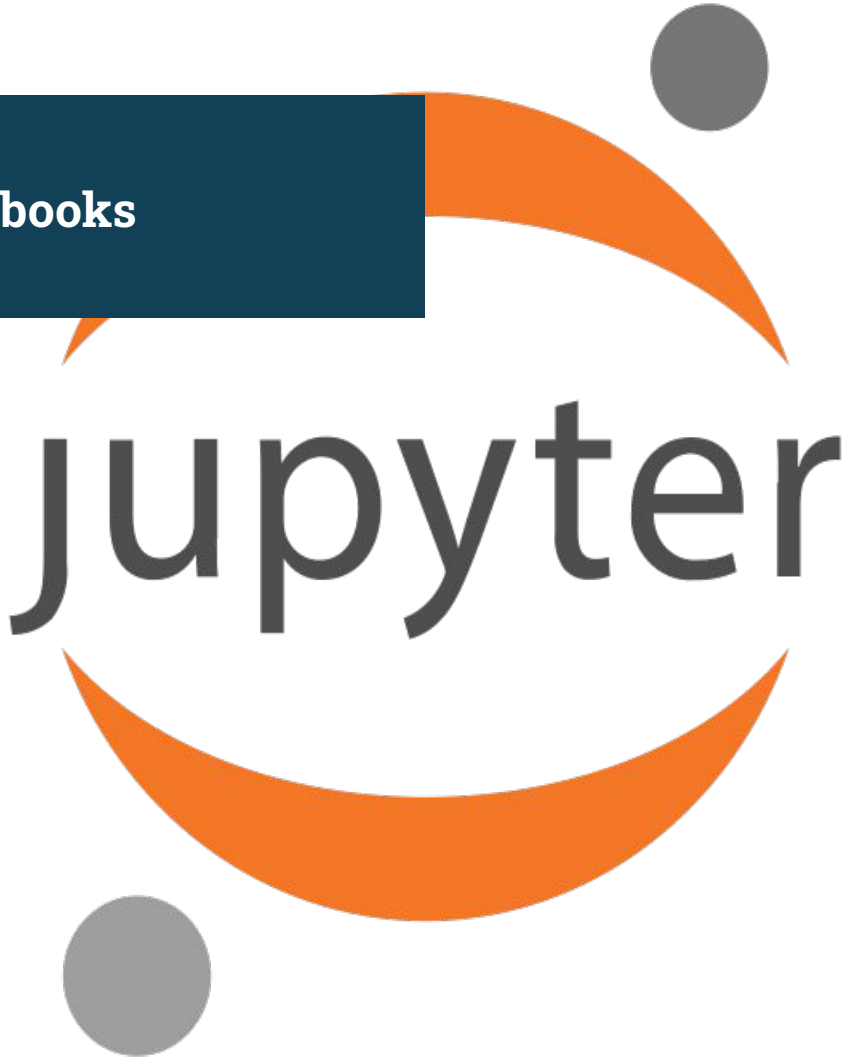
- If you need to install a python package that requires *compiling*, then you shouldn't use a conda environment!
- Use Python module with the corresponding buildenv modules
- Create a virtual environment for building and adding packages.

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ module load buildenv-gcc/2022a-eb
[x_abcde@tetralith]$ virtualenv --system-site-packages myownvirtualenv
[x_abcde@tetralith]$ source myownvirtualenv/bin/activate
(myownvirtualenv)[x_abcde@tetralith]$ …
```

Mixing conda and software compiled from source code or via pip

- It is possible to compile software in relation to Anaconda: either using the NSC-provided compilers or conda-forge package compilers – **but be careful!**
- Please read: Anaconda @ NSC, section “Mixing conda and software compiled from source code or via pip“

jupyter notebooks



Jupyter notebooks

- Jupyter notebooks can be run on with the login nodes or compute nodes.
- Jupyter packages are included in the `Python/3.10.4-env-hpc1-gcc-2022a-eb` module

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ pip list | grep jupyter
…
```

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

Jupyter notebooks

- Recommendation: Use jupyter in combination with thinlinc.
  - E.g. on a login node, in a thinlinc terminal:

```
[x_abcde@tetralith]$ module load Python/3.10.4-env-hpc1-gcc-2022a-eb
[x_abcde@tetralith]$ jupyter-notebook
[x_abcde@tetralith]$ …
```

- Jupyter notebooks can also be used via an ssh tunnel

mpi4py

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

```
[x_abcde@tetralith]$ module load  Python/3.10.4-env-hpc1-gcc-2022a-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 naiss2023-x-yyy
#SBATCH -n 4
#SBATCH -t 01:00:00
#SBATCH -J jobname

module load Python/3.10.4-env-hpc1-gcc-2022a-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)
- It is possible to use `conda install` and `pip install` in the same environment
  1. Create a conda environment using an `Anaconda` module
  2. `conda install pip`
  3. `pip install mypipmodule`