



Using Python @ NSC

**Warning: Python bites can
be very painful**



OS Python

- NSC's clusters have the CentOS standard Python 2.7.5 and Python 3.4.10 installed.

```
[x_abcde@tetralith]$ module purge
[x_abcde@tetralith]$ which python
/usr/bin/python
[x_abcde@tetralith]$ which python3
/usr/bin/python3
```

- Recommendation: don't use them

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/  
.  
.  
.  
Python/recommendation          (D)  
Python/2.7.14-anaconda-5.0.1-nsc1  
Python/2.7.14-nsc1-gcc-2018a-eb  
.  
.  
.  
Python/3.6.3-anaconda-5.0.1-nsc1  
Python/3.6.4-nsc1-intel-2018a-eb  
.  
.  
.
```

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/3.7.0-anaconda-5.3.0-extras-nsc1
$ python
Python 3.7.0 (default, Jun 28 2018, 13:15:42)
[GCC 7.2.0] :: Anaconda, Inc. 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

Three flavors of Python module

1. Python/x.y.z-nscX-toolchain
2. Python/x.y.z-env-nscX-toolchain
3. Python/x.y.z-anaconda-....

Some guidelines to help you choose a module:

- A. There are generally more packages included in the Anaconda installations
- B. If there are modules that only differ in the -nsc build/installation tag, then choose the one with the highest integer (e.g. nsc2 rather than nsc1)

Python modules

Check available packages in a module: Anaconda modules

To list the installed packages in an Anaconda Python installation, simply load the module and run: conda list.

If you are looking for a specific package, then pipe the output from `conda list` to `grep`:

```
[x_abcde@tetralith]$ module load Python/3.6.3-anaconda-5.0.1-nsc1
[x_abcde@tetralith]$ conda list | grep -i scipy
scipy                0.19.1                py36h9976243_3
```


Python modules

Check available packages in a module: NSC build modules

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/3.6.4-nsc2-intel-2018a-eb
[x_abcde@tetralith]$ pip list --format=legacy | grep -i scipy
scipy (1.0.0)
```

A hand holding a glass globe containing a small plant, symbolizing environmental care or a controlled environment. The globe is held in the center of the frame, with the hand's fingers visible around it. The background is a soft, out-of-focus green and blue bokeh, suggesting an outdoor setting with foliage and sky. The overall mood is one of care and attention to detail.

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 (NOT Python/x.y.z-anaconda-...)

```
[x_abcde@tetralith]$ module load Anaconda/2021.05-nsc1
[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 3.8.13 | packaged by conda-forge | (default, Mar 25 2022, 06:04:18)
[GCC 10.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas
>>> pandas.__version__
'1.4.2'
```

Managing your python environment: conda

- If you need to use the conda command for anything other than `conda list`, use an **Anaconda** module to create your own local conda environment.
- 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 <https://www.nsc.liu.se/software/python/>
- If you need to install a python package that requires **compiling**, then you should NOT use a conda environment! (try virtualenv instead).

Managing your python environment: virtualenv

- Use any of the **Python/x.y.z** modules for managing environments using virtualenv

```
[x_abcde@tetralith]$ module load Python/3.6.7-env-nsc1-gcc-2018a-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 3.6.7 (default, Nov 26 2018, 16:42:15)
[GCC 6.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import hostlist
>>> hostlist.__file__
'/home/x_abcde/myownvirtualenv/lib/python3.6/site-packages/hostlist.py'
```

Installing packages that require compiling

- Start from the gcc or intel Python and corresponding buildenv modules
- Create a virtual environment for building and adding packages using pip.

```
[x_abcde@tetralith]$ module load Python/3.6.7-env-nsc1-gcc-2018a-eb
[x_abcde@tetralith]$ module load buildenv-gcc/2018a-eb
[x_abcde@tetralith]$ virtualenv --system-site-packages myownvirtualenv
[x_abcde@tetralith]$ source myownvirtualenv/bin/activate
(myownvirtualenv) [x_abcde@tetralith]$ ...
```

jupyter notebooks

The Jupyter logo is a stylized orange smiley face with two grey circles for eyes. The word "jupyter" is written in a dark grey, lowercase, sans-serif font across the middle of the smiley face.

jupyter

Jupyter notebooks

- Jupyter notebooks can be run on with the login nodes or compute nodes.
- Jupyter packages are included in the `Python/x.y.z-anaconda-...-extras-nsc1` modules

```
[x_abcde@tetralith]$ module load Python/3.8.3-anaconda-2020.07-extras-nsc1
[x_abcde@tetralith]$ conda 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/3.8.3-anaconda-2020.07-extras-nsc1
[x_abcde@tetralith]$ jupyter-notebook
[x_abcde@tetralith]$ ...
```

- Jupyter notebooks can also be used via an ssh tunnel

mpi4py



mpi4py

- Python scripts that use mpi4py are now supported by the latest version of NSC's mpi launcher ([mpprun/4.3.0](#)).
 - E.g. interactive:

```
[x_abcde@tetralith]$ module load Python/3.6.7-env-nsc1-gcc-2018a-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 snic2022-x-yyy
#SBATCH -n 4
#SBATCH -t 01:00:00
#SBATCH -J jobname
```

```
module load Python/3.6.7-env-nsc1-gcc-2018a-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`
- [Mambaforge](#) modules are available on Tetralith/Sigma as a “drop-in” replacement to Anaconda
 - `Mambaforge/4.12.0-0-nsc1`