In your own research, do you use...

SciPy Stack?
NumPy, SciPy, Matplotlib, Pandas, SymPy, IPython/Jupyter

Analytics Frameworks?
PySpark, Dask

Machine/Deep Learning?
scikit-learn, TensorFlow, Keras, PyTorch, etc.

Parallelism/Performance/Performance IO Libraries?
NumExpr, Numba, mpi4py, H5py

Other Python Visualization Tools
Bokeh, bqplot, Seaborn, yt
Python Usage Statistics

Module Load Python [Non-Staff]: 167,290
Unique Users [Non-Staff]: 245
Good docs advise on how to use Python at NERSC.

Updates are ~continuous. [Main page.](#)

Frequently Asked Questions [FAQ page.](#)
Suggest new questions!

Advice/gotchas for KNL users [KNL page.](#)
Use Environment Modules

Environment modules:
Environment modules project:
http://modules.sourceforge.net/

Always* “module load python”
Do not use /usr/bin/python.
Using #!/usr/bin/env python is OK!

What is there?
module avail python

* Unless you install your own somehow.
(Totally fine, see later in the talk.)
NERSC’s builds of Python on Cori/Edison have been retired for a while.

Modules Now Leverage Anaconda Python

- Distro for large-scale data analytics and scientific computing.
- Handy package management and deployment (conda tool).
- Conda environments replace virtualenv.

Hundreds of useful packages

- Threaded Intel MKL comes for free.
- Now with some ML tools too.
- Additional “channels” and you can use pip still.

Monolithic modules, some add-on modules (h5py-parallel).

https://docs.anaconda.com/anaconda/
Recommended environment modules at NERSC on Cori or Edison for Python users:

```
module load python/2.7-anaconda-4.4
module load python/3.6-anaconda-4.4
```

Default Python is 2.7 up to no later than 21 months from now:

```
module load python
[= module load python/2.7-anaconda-4.4]
```

https://pythonclock.org/
Doing Things Yourself

Your own Anaconda or Miniconda installation, maybe at 
/global/common/software/<project-name>

module unload python
unset PYTHONSTARTUP

wget https://repo.continuum.io/miniconda/Miniconda2-latest-Linux-x86_64.sh
/bin/bash Miniconda2-latest-Linux-x86_64.sh -b -p $PREFIX
source $PREFIX/bin/activate
  <or export PATH=$PREFIX/bin:$PATH>
conda install basemap yt...
Building your own mpi4py or parallel h5py?  
Do not conda install ...  
Do not pip install ...  
Link to Cray MPICH, using compiler wrappers

wget https://bitbucket.org/mpi4py/mpi4py/downloads/mpi4py-2.0.0.tar.gz  
tar zxvf mpi4py-2.0.0.tar.gz  
cd mpi4py-2.0.0  
module swap PrgEnv-intel PrgEnv-gnu  
python setup.py build --mpicc=$(which cc)  
python setup.py install
Parallelism with Python

Within a node:
Use OpenMP-threaded math libs. Multiprocessing is OK too.

Multi-node parallelism:
Best supported by mpi4py. Dask, PySpark work too.

Hybrid parallelism:
Best route is mpi4py + threaded math libs.
Ye olde stand-by, `print()`!

```
srun -u python -u <script-name> ...  
```

Unbuffered output hurts performance.
Can be a lot of messy output to parse.

Good for general exploration (standard lib):
- `cProfile` plus `snakeviz` or `gprof2dot`
- MPI processes? [see fbpic example here]

Good for a deeper dive on one function (package):
- `line_profiler`

High-performance instrumented timer (mixed-language, MPI, package):
- `TiMemory`

High-performance tools (mixed-language, MPI):
- Intel VTune (some collection methods) on Intel Python
Use Shifter to Scale Up Python

Python’s “import” statement is file metadata intensive (.py, .pyc, .so open/stat calls).
=> Very slow times to just start Python applications at larger concurrency (MPI).

BEST POSSIBLE PERFORMANCE IS THROUGH SHIFTER:
Eliminates metadata calls off the compute nodes.
Paths to .so libraries can be cached via ldconfig.

Other approaches:
Pack up software to compute nodes (python-mpi-bcast).
Install software to $SCRATCH or /global/common/software.
Using Jupyter at NERSC

Code, text, equations, viz in a narrative.

Two URLs for Jupyter at NERSC.

**jupyter.nersc.gov:**
- External to Edison/Cori
- Can’t see $SCRATCH
- But can see NGF

**jupyter-dev.nersc.gov:**
- Spawns notebooks on Cori
- Can see Cori $SCRATCH
- Same Python env as ssh login
- Can submit jobs via `%sbatch`
Most common Jupyter question from beginners:

“How do I take a conda environment and turn it into a Jupyter kernel?”

Several ways to accomplish this, here’s the easy one.

```
$ module load python
$ conda create -n myenv python=3.6
$ source activate myenv
(myenv) $ conda install ipykernel <other-packages>...
(myenv) $ python -m ipykernel install --user --name myenv-jupyter
```

Point your browser to jupyter-dev.nersc.gov. (You may need to restart your notebook server via control panel). Kernel “myenv-jupyter” should be present in the kernel dropdown.
The kernelspec File

```
(myenv) rthomas@cori01:~> cat \n    $HOME/.local/share/jupyter/kernels/myenv-jupyter/kernel.json

{
    "argv": [ 
        "/global/homes/r/rthomas/.conda/envs/myenv/bin/python",
        "-m",
        "ipykernel_launcher",
        "-f",
        "{connection_file}" 
    ],
    "display_name": "myenv-jupyter",
    "language": "python"
}
```
Additional Customization

```json
{
  "argv": ["/global/homes/r/rthomas/.conda/envs/myenv/bin/python", "-m", "ipykernel_launcher", "-f", "{connection_file}"],
  "display_name": "myenv-jupyter",
  "language": "python",
  "env": {
    "PATH": ...
  }
}
```
{  
  "argv": [  
    "/global/homes/r/rthomas/jupyter-helper.sh",
    "-f",
    "{connection_file}"  
  ],  
  "display_name": "myenv-jupyter2",
  "language": "python",
}

Meanwhile, in jupyter-helper.sh:

```bash
#!/bin/bash
export SOMETHING=123
module load texlive
exec python -m ipykernel "$@
```
Debugging Jupyter-dev Stuff

(myenv) rthomas@cori01:~> cat ~/jupyter.log

[I 2018-03-19 16:00:08.175 SingleUserNotebookApp manager:40] [nb_conda_kernels] enabled, 5 kernels found
[I 2018-03-19 16:00:08.248 SingleUserNotebookApp extension:54] JupyterLab application directory is /global/common/cori/software/python/3.6-anaconda-4.4/share/jupyter/lab
[I 2018-03-19 16:00:09.123 SingleUserNotebookApp handlers:73] [nb_ancondacloud] enabled
[I 2018-03-19 16:00:09.129 SingleUserNotebookApp handlers:292] [nb_conda] enabled
[I 2018-03-19 16:00:09.181 SingleUserNotebookApp __init__:35] ✓ npresent HTML export ENABLED
[I 2018-03-19 16:00:09.186 SingleUserNotebookApp singleuser:365] Starting jupyterhub-singleuser server version 0.8.0.rc1
[I 2018-03-19 16:00:09.194 SingleUserNotebookApp notebookapp:1445] 0 active kernels
[I 2018-03-19 16:00:09.194 SingleUserNotebookApp notebookapp:1445] The Jupyter Notebook is running at:
[I 2018-03-19 16:00:09.194 SingleUserNotebookApp notebookapp:1445] http://0.0.0.0:56901/user/rthomas/
[I 2018-03-19 16:00:09.194 SingleUserNotebookApp notebookapp:1446] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[I 2018-03-19 16:00:09.236 SingleUserNotebookApp log:122] 302 GET /user/rthomas/ → /user/rthomas/tree/global/homes/r/rthomas? (@::ffff:10.42.245.15) 0.39ms
Python and Jupyter at NERSC

Python & Jupyter: integral elements of NERSC’s Data Intensive Science portfolio.

We want users to have a:

- **familiar** Python environment
- **productive** Python experience
- **performant** Python software stack

Always looking for:

New ways to empower Python & data users.

Feedback, advice, and even help:

consult@nersc.gov or https://help.nersc.gov/
rcthomas@lbl.gov
National Energy Research Scientific Computing Center