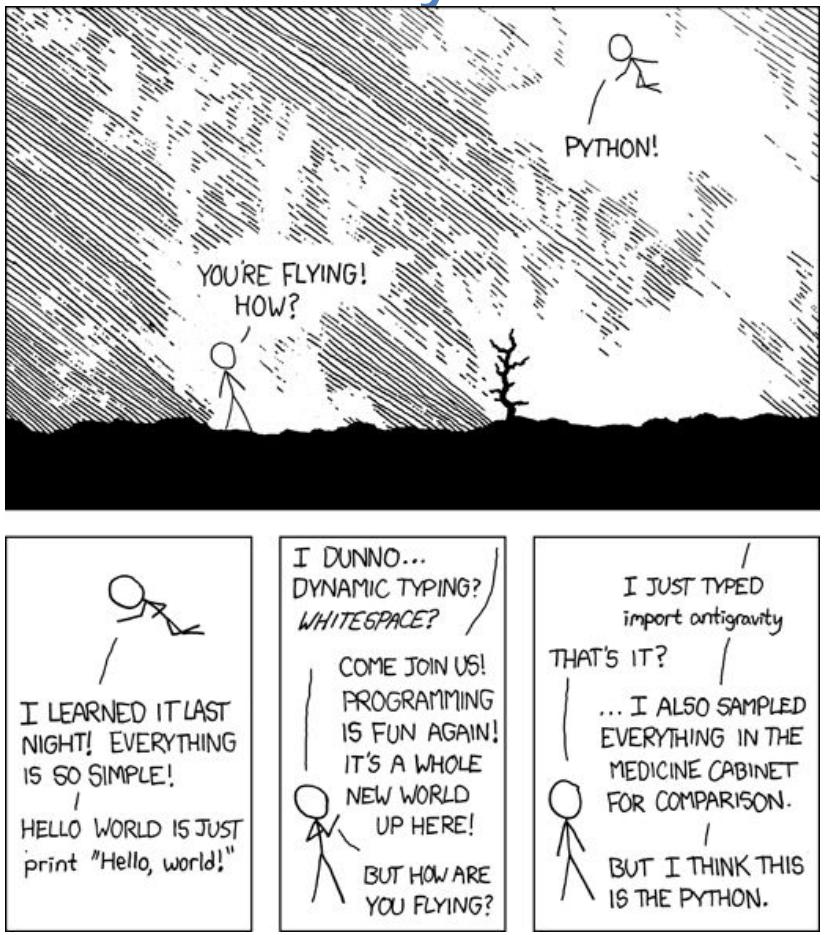
Python at NERSC

New User Training
June 16, 2020

Laurie Stephey
Data Analytics and Services Group
lastephey@lbl.gov

What is Python?

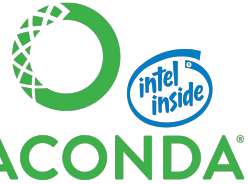


- We probably don't have to tell you about Python
- Extremely popular interpreted language, continuing to grow
- Libraries like NumPy, SciPy, scikit-learn commonly used for scientific analysis
- Machine learning tools covered in M. Mustafa's talk later today

https://xkcd.com/353/

Can I Use Python at NERSC? Yes!

- In the “old days” languages like C and Fortran were the only options at HPC/supercomputing centers
- Now NERSC has 7000+ users, most of whom use Python for some part of their work, and this number is growing
- Python is fully supported at NERSC- we use Anaconda Python to provide pre-built environments and the ability for users to create their own environments
- We are here to help enable your Python scientific software at both small and large scale



https://docs.anaconda.com/anaconda/

Two Major Options for Python at NERSC

Pre-built Python environment

```
module load python
```

Best for basic things like NumPy, SciPy, mpi4py

Build your own conda environment

```
module load python
conda create -n
myenv…
```

Best for using custom packages (from github, for example) or libraries not in the default environment

Making Your Own Python Conda Environment

```
module load python
conda create -n myenv python=3.7
source activate myenv
conda (or pip) install yourfavoritepackage
###import antigravity
source deactivate myenv
```

- That's it!
- You can install anything you need
- Easy to convert into a Jupyter kernel
- If something goes wrong, just delete it and make a new one

How to Scale Up Your Python Code?

Since you're here at NERSC, chances are you want to run your Python code in parallel. There are many options, including:

- Multiprocessing
  - Single node only, process parallelism via a pool of workers
- Dask
  - Single or many nodes, framework to create a group of workers that execute tasks coordinated by a scheduler, nice visualization tools
- mpi4py
  - Single or many nodes, best performance when used together with a container (Docker/Shifter)
- See our docs for more info:
  https://docs.nersc.gov/development/languages/python/scaling-up/

Most Common Python Problems at NERSC

Here are the most common* reasons people submit Python tickets:

1) **Trouble with installing and scaling mpi4py**
2) **Trouble as a result of settings in dotfiles**

Let's talk about how you can avoid these problems!

**** My unscientific opinion after answering Python tickets for ~ 1 year***

mpi4py Installation at NERSC

- mpi4py is our most complex Python package in terms of how it is installed
- Do not `pip install mpi4py` or `conda install mpi4py`! This works on your laptop but not on our complex supercomputer!
- Your options:
  - If you module load python, mpi4py is already there and you are ready to go
  - If you build your own conda environment, you'll need to install mpi4py inside following our recipe: https://docs.nersc.gov/development/languages/python/mpi4py/#mpi4py-in-your-custom-conda-environment
- Note that MPI is disabled on all NERSC login nodes. You'll need to be on a compute node to use mpi4py.

Containers Best for mpi4py Scaling

You try to launch your mpi4py job on 1000 nodes and suddenly it's really slow. What happened?!?

Every time you `import numpy` inside your Python script, it has to move data across our filesystems. For 1000 nodes, this can be really slow!

If you plan to run mpi4py at large scale, we *strongly* recommend that you build your Python inside a Docker container and run via Shifter. This will make your startup much faster and resilient to filesystem congestion. See our docs for more info:

https://docs.nersc.gov/development/languages/python/scaling-up/#shifter-the-best-way-to-run-python-at-scale

Python Problems from Dotfiles

Dotfiles = .bashrc, .bashrc.ext, .bash_profile, bash_profile.ext ...

These files let you customize your NERSC environment at startup.

But! it's easy to put things in these files and forget about them.

When users submit Python tickets, usually the first thing we do is check your dotfiles.

User: Python is broken
NERSC staff: Python is fine, your .bashrc.ext just has some bad settings
User:

Here Is a Bad .bashrc.ext Example

```
[redacted] Python setup
#export PATH=/project/projectdirs/polar/software/edison/shared_gnu/bin:$PATH
#export LD_LIBRARY_PATH=/project/projectdirs/polar/software/edison/shared_gnu/lib:$PATH:$LD_LIBRARY_PATH
#export PYTHONPATH=/project/projectdirs/polar/software/edison/lib/python2.7/site-packages/:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]FORECAST/xForecast/codes:$PYTHONPATH
#export PYINST=/project/projectdirs/polar/software/edison/lib/python2.7
#export EM_DIR=/global/homes/d/[redacted]'cmb/explicit_mapmaking
export PYTHONPATH=$PYTHONPATH:$EM_DIR/pytools

export ABPATH=/global/homes/[redacted]'POLARBEAR/AnalysisBackend
export PYTHONPATH=/global/homes/[redacted]/POLARBEAR/:$ABPATH:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]/SOFTWARES/:$ABPATH:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]'FORECAST/self_consistent_forecast/codes:$PYTHONPATH

[redacted]
#module load gcc/4.9.3
#module load cray-hdf5
#module load allineatools
#module load boost
#module load fftw
#module load ipm
#export EM_DIR=/global/homes/[redacted]POLARBEAR/cmb/explicit_mapmaking

#export MIRAMARE=edison
#export PATH=/global/homes/[redacted]SOFTWARES/wmap_likelihood_v5:$PATH
#export PATH=/global/homes/[redacted]SOFTWARES/:$PATH
#export PATH=/global/homes/[redacted]'SOFTWARES/cosmomc:$PATH
#module load cmb
#module load lmfit-hpcp
#module load h5py-hpcp
#export PYTHONPATH=/global/homes/[redacted]/FORECAST/self_consistent_forecast/codes:$PYTHONPATH
#export PYTHONPATH=/global/homes/[redacted]'FORECAST/xForecast/codes:$PYTHONPATH
#export PYTHONPATH=/global/homes/[redacted]/SOFTWARES:$PYTHONPATH
#PATH=${PATH}:~/.local/bin
#export PYTHONPATH=/project/projectdirs/polar/software/edison/lib/python2.7/site-packages/:$PYTHONPATH

export PYTHONPATH=/global/homes/[redacted]'FORECAST/xForecast/codes:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]/FORECAST/xForecast/sim_exploitation:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]/SIMONS_OBS:$PYTHONPATH
export PYTHONPATH=/global/homes/[redacted]/SOFTWARES/PySM_public:$PYTHONPATH
export PYTHONPATH=/project/projectdirs/polar/software/edison:$PYTHONPATH

export LD_LIBRARY_PATH=/global/homes/[redacted]SOFTWARES/libsharp/auto/lib/:$LD_LIBRARY_PATH
export LDFLAGS="-L/usr/common/software/gsl/2.1/intel/lib -L/usr/common/software/cfitsio/3.370-reentrant/hsw/intel/lib -L/global.homes/
[redacted]'SOFTWARES/libsharp/auto/lib":$LD_LIBRARY_PATH
export CPPFLAGS="-I/usr/common/software/gsl/2.1/intel/include -I/usr/common/software/cfitsio/3.370-reentrant/hsw/intel/include -I/globa
l.homes/[redacted]/SOFTWARES/libsharp/auto/include"
export CC=cc
```

If this user encounters a problem, all these settings make it really difficult to find the actual issue

Avoid problems by keeping your dotfiles clean:

- **Add only what you really need**
- **Periodically check to make sure the settings are up-to-date**
- **This .bashrc.ext also had settings from Hopper, Carver, and Edison...**

Should I Use Python 2 or Python 3?

Python 3!

Python 2 retired on January 1, 2020. Developers are no longer supporting Python 2. See our docs for more info: https://docs.nersc.gov/development/languages/python/#end-of-life-for-python-2

We still provide some pre-built Python 2 environments, but we don't promise to keep these forever.

We will *not* support Python 2 on Perlmutter so please transition your code now.

What About Perlmutter and GPUs?!?!

- Good question! There is no easy `import gpu` in Python.
- Python GPU frameworks come in two major categories:
  - Drop-in replacement for NumPy/SciPy
  - Custom GPU kernel
- You may actually need a combination of frameworks to fully port your code
- NVIDIA RAPIDS is another option for pandas/scikit-learn, but we we won't cover that in detail here. See our docs for more info: https://docs-dev.nersc.gov/cgpu/examples/#nvidia-rapids

Drop-in Replacements for NumPy/SciPy

- CuPy
  - Very easy to use, looks very much like NumPy
  - Backend is CUDA, will only work on NVIDIA GPUs
  - Also has its own method for user defined kernels
- JAX
  - Developed by Google, backend uses XLA compiler
  - Harder to use, may require some code changes
  - Will work on many kinds of hardware, both CPUs and GPUs
  - Also has JIT compiler for custom kernels
- Legate, Bohrium, GrumPy, Weld…

Custom GPU Kernel Options

- Numba CUDA
  - JIT compiles Python code to GPU code
  - Numba kernels look more like CUDA than traditional Python
  - Will only work on NVIDIA GPUs (Numba AMD also exists)
- PyCUDA
  - CUDA kernels wrapped in Python, more powerful than Numba
  - User has to know and understand CUDA
  - Will only work on NVIDIA GPUs
- PyOpenCL
  - OpenCL kernels wrapped in Python
  - User has to know and understand OpenCL
  - Portable-- will run on any CPU and GPU

Getting Started Using Python on GPUs

General recommendation: Choose 1 NumPy replacement + 1 custom kernel framework

Quickstart: CuPy + Numba CUDA

- Pros: easier to use, minor code changes required
- Cons: tied to NVIDIA GPUs

Portable: JAX + PyOpenCL

- Pros: will run on any hardware
- Cons: harder to use, have to understand OpenCL

See our docs for more info:
https://docs.nersc.gov/performance/readiness/#python

tl;dr

- Welcome to NERSC!
- We work hard to have clear and helpful Python documentation. Please read it! It will save you time and trouble: https://docs.nersc.gov/development/languages/python/
- Documentation suggestions (or contributions!) welcome
- Avoid common Python problems:
  - Be careful when installing mpi4py and consider using a container for better mpi4py performance
  - Make your life easy and keep your dotfiles clean
- If you have questions, submit a ticket at help.nersc.gov

Thank You and Welcome to NERSC!

`source activate` vs. `conda activate`

After creating a conda environment you will see:

```
# To activate this environment, use
#     $ conda activate myenv
# To deactivate an active environment, use
#     $ conda deactivate
```

You can use `source activate` OR `conda activate`

- `source activate` doesn't make any changes to your setup
- `conda activate` will make changes to your .bashrc file
- See our docs for more info: https://docs.nersc.gov/development/languages/python/#creating-conda-environments

Helpful Python Commands at NERSC

`module load python` → loads the default Python environment

`module avail python` → shows all of our pre-built Python environment options

`conda list` → shows you which libraries are in the environment

`conda create -n myenv` → make a new environment

`source activate myenv/conda activate myenv` → start your custom environment

`source deactivate/conda deactivate` → exit your custom environment

`conda clean -a` → clean up your .conda directory, including packages left over from upgraded/deleted environments, important for keeping your $HOME directory under quota!