Using Python at CHPC

Brett Milash
Center for High Performance Computing
06/28/2023
Overview

• Selecting a version of python
• Installing python
• Installing python packages
• Running python
  • scripts
  • Jupyter notebook
  • VSCode IDE
Which python?

$ python
bash: python: command not found...
$ /usr/bin/python3 --version
Python 3.6.8
$ /usr/bin/python2 --version
Python 2.7.18

• Python release cycle: https://python-release-cycle.glitch.me/
• Don’t use the system-provided pythons
  • They are really old!
  • You don’t control when it is updated
• Alternatives:
  • Use a CHPC-installed python module
  • Install your own
CHPC-installed python modules

$ module spider python
------------------------------------------
python:
------------------------------------------
  Versions:
    python/2.7.18
    python/3.6.8
    python/3.8.8
    python/3.9.15
    python/3.10.3
$ module load python/3.10.3
$ python --version
Python 3.10.3
$ which python
/uufs/chpc.utah.edu/sys/installdir/r8/python/3.10.3/bin/python
Install your own python

• User-installed python at CHPC:
  • [https://www.chpc.utah.edu/documentation/software/python-anaconda.php](https://www.chpc.utah.edu/documentation/software/python-anaconda.php)

• Miniconda: [https://docs.conda.io/en/latest/miniconda.html](https://docs.conda.io/en/latest/miniconda.html)
  • Requires 510 MB, ~ 1% of a 50 GB disk space quota

$ mkdir -p Software/MyPython
$ cd Software/MyPython
$ wget https://repo.anaconda.com/...
$ bash Miniconda3-py311_23.5.2-0-Linux-x86_64.sh -h
$ bash ./Miniconda3-py311_23.5.2-0-Linux-x86_64.sh -b -f -p -s
$ cd bin
$ export PATH=$(pwd):$PATH

• Can also create your own python module
Installing packages

• You can install packages yourself, using your own Python or CHPC’s
• You can install precompiled packages, or install from source
• You can install packages where you want
• You can install using “pip” or from source code
Installing packages with pip

- [https://docs.python.org/3/installing/index.html](https://docs.python.org/3/installing/index.html)
- Installs packages from the Python Package: [https://pypi.org/](https://pypi.org/)
- Pros:
  - “pip is the preferred installer program”
  - installs / upgrades prerequisites
  - can uninstall too
  - --dry-run option (added with python version 3.10)
  - --user (default if main library location is un-writable)
- Cons:
  - have been addressed in recent releases - older versions can be dangerous
Installing packages from source with setup.py

- Python packages include a script named “setup.py”
  ```bash
  python setup.py install --help
  ```
- Useful if package isn’t found on pypi.org
- Allows you to examine prerequisites in requirements.txt
Where do the packages go?

• By default:
  `<main_python_location>/lib/python<version>/site-packages`

• User directory:
  `$HOME/.local/lib/python<version>/site-packages`

• If using `--prefix=`
  `<prefix_location>/lib/python<version>/site-packages`
  • Need to add this location to PYTHONPATH
## Methods to access resources at CHPC

<table>
<thead>
<tr>
<th>Method</th>
<th>Attributes</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh to interactive node</td>
<td>command line or GUI</td>
<td>interactive node</td>
</tr>
<tr>
<td>Fastx to interactive node</td>
<td>command line or GUI, persistence</td>
<td>interactive node</td>
</tr>
<tr>
<td>OnDemand cluster shell access</td>
<td>convenient, like ssh, command line only</td>
<td>interactive node</td>
</tr>
<tr>
<td>SLURM sbatch command</td>
<td>non-interactive (batch mode)</td>
<td>compute node(s)</td>
</tr>
<tr>
<td>SLURM salloc command</td>
<td>interactive command-line or GUI</td>
<td>compute node(s)</td>
</tr>
<tr>
<td>OnDemand system installed applications</td>
<td>web-based access</td>
<td>compute node(s)</td>
</tr>
</tbody>
</table>

Remember the appropriate uses for interactive and compute nodes:

- Interactive nodes: writing code, installing code, small-scale testing, debugging, managing SLURM jobs
- Compute nodes: heavy-duty computing (simulations, stats, data visualization) whether interactive or not
## Methods of using Python at CHPC

<table>
<thead>
<tr>
<th>Command</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>python, python scriptname.py ...</td>
<td>Interactive, command line style</td>
</tr>
<tr>
<td>python scriptname.py ...</td>
<td>Batch jobs, script</td>
</tr>
<tr>
<td>Jupyter Lab or Notebook</td>
<td>Interactive, document based, web browser GUI</td>
</tr>
<tr>
<td>vscode, pycharm</td>
<td>Integrated Development Environment, GUI</td>
</tr>
<tr>
<td>Workflows, e.g. snakemake, nextflow</td>
<td>Non-interactive workflow tools</td>
</tr>
</tbody>
</table>
Python use methods vs. CHPC access methods

<table>
<thead>
<tr>
<th>Method</th>
<th>python, interactive</th>
<th>python, script</th>
<th>Jupyter</th>
<th>VSCode, Pycharm</th>
<th>Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh (to interactive node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fastx (to interactive node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnDemand cluster shell access (runs on interactive node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLURM sbatch (compute node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLURM salloc (compute node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnDemand system installed applications (runs on compute nodes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
ssh, Fastx, and OnDemand shell access demo

• These methods provide access to interactive nodes
• Graphics (whether GUI or graphical output) requires X-forwarding
  • On Mac use "ssh -Y username@hostname"
  • On Windows use Xming (https://xming.en.softonic.com/)
• X-forwarding can be slow without some help
# Python use methods vs. CHPC access methods

<table>
<thead>
<tr>
<th>Access Method</th>
<th>python, interactive</th>
<th>python, script</th>
<th>Jupyter</th>
<th>VSCode, PyCharm</th>
<th>Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh (to interactive node)</td>
<td>✅ good for testing</td>
<td>✅ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ❌ PyCharm ✓</td>
<td>✅ good for testing</td>
</tr>
<tr>
<td>Fastx (to interactive node)</td>
<td>✅ good for testing</td>
<td>✅ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ❌ PyCharm ✓</td>
<td>✅ good for testing</td>
</tr>
<tr>
<td>OnDemand cluster shell access (on interactive node)</td>
<td>✅ good for testing, but no graphics</td>
<td>✅ good for testing, but no graphics</td>
<td>❌ - requires X windows</td>
<td>❌ - requires X windows</td>
<td>✅ good for testing, but no graphics</td>
</tr>
<tr>
<td>SLURM sbatch (compute node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SLURM salloc (compute node)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OnDemand system installed applications (on compute nodes)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“✅ good for testing” means software works well within computing limits of interactive node
SLURM sbatch and salloc demo

• Both methods provide access to compute nodes
• sbatch is batch oriented - therefore non-interactive
• salloc starts an interactive shell session on a compute node
## Python use methods vs. CHPC access methods

<table>
<thead>
<tr>
<th>Method</th>
<th>python, interactive</th>
<th>python, script</th>
<th>Jupyter</th>
<th>VSCode, PyCharm</th>
<th>Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh (to interactive node)</td>
<td>✓ good for testing</td>
<td>✓ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✗ PyCharm ✓</td>
<td>✓ good for testing</td>
</tr>
<tr>
<td>Fastx (to interactive node)</td>
<td>✓ good for testing</td>
<td>✓ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✗ PyCharm ✓</td>
<td>✓ good for testing</td>
</tr>
<tr>
<td>OnDemand cluster shell access (on interactive node)</td>
<td>✓ good for testing, but no graphics</td>
<td>✓ good for testing, but no graphics</td>
<td>✗ - requires X windows</td>
<td>✗ - requires X windows</td>
<td>✓ good for testing, but no graphics</td>
</tr>
<tr>
<td>SLURM sbatch (compute node)</td>
<td>✗</td>
<td>🟢🟢🟢🟢🟢</td>
<td>✗</td>
<td>✗</td>
<td>🟢🟢🟢🟢🟢</td>
</tr>
<tr>
<td>SLURM salloc (compute node)</td>
<td>✓</td>
<td>✓</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✓ PyCharm ✓</td>
<td>✓</td>
</tr>
<tr>
<td>OnDemand system installed applications (on compute nodes)</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

“ ✓ good for testing” means software works well within computing limits of interactive node
OnDemand demo

• Web portal
• Access to compute nodes
• Very good for web and GUI applications
## Python use methods vs. CHPC access methods

<table>
<thead>
<tr>
<th>Method</th>
<th>python, interactive</th>
<th>python, script</th>
<th>Jupyter</th>
<th>VSCode, PyCharm</th>
<th>Workflows</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssh (to interactive node)</td>
<td>✅ good for testing</td>
<td>✅ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✗ PyCharm ✓</td>
<td>✅ good for testing</td>
</tr>
<tr>
<td>Fastx (to interactive node)</td>
<td>✅ good for testing</td>
<td>✅ good for testing</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✗ PyCharm ✓</td>
<td>✅ good for testing</td>
</tr>
<tr>
<td>OnDemand cluster shell access (on interactive node)</td>
<td>✅ good for testing, but no graphics</td>
<td>✅ good for testing, but no graphics</td>
<td>✋ - requires X windows</td>
<td>✋ - requires X windows</td>
<td>✅ good for testing, but no graphics</td>
</tr>
<tr>
<td>SLURM sbatch (compute node)</td>
<td>✗</td>
<td>★★★★★</td>
<td>✗</td>
<td>✗</td>
<td>★★★★★</td>
</tr>
<tr>
<td>SLURM salloc (compute node)</td>
<td>✓</td>
<td>✓</td>
<td>Inefficient - not recommended</td>
<td>VSCode ✓ PyCharm ✓</td>
<td>✓</td>
</tr>
<tr>
<td>OnDemand system installed applications (on compute nodes)</td>
<td>✗</td>
<td>✓</td>
<td>★★★★★</td>
<td>VSCode ★★★★ PyCharm ✓</td>
<td>✗</td>
</tr>
</tbody>
</table>

“✅ good for testing” means software works well within computing limits of interactive node