Python Basics on the FASRC Clusters

Learning objectives

- Python using CLI (Command Line Interface)
  - Interactive
  - Sbatch
- Python Package installation
  - Interactive
  - Sbatch
- Python using OOD (Open On Demand)
- Jupyter Notebook
  - Create conda environment (i.e., jupyter kernel)

Python Programming Language [Python Programming Language – FASRC DOCS]

- High-level, general-purpose, and object-oriented programming language with emphasis on code readability and use of significant indentation.
- Ideal for scripting and rapid application development given its dynamic typing, elegant syntax, and automatic memory management (garbage collection).
- Has a comprehensive standard library. Also known as “batteries-included” language.
- Python’s implementation is mostly in C.
  - Python’s core interpreter, CPython, written in C.
- Interpreted language, hence slower than compiled languages, like C and Fortran.
  - Compiled generates executable
  - Interpreted executes instructions directly on the fly without compiling a program into machine language

Python using CLI - Interactive

```
# Login to Cannon
ssh <username>@login.rc.fas.harvard.edu

# Go to a compute node on the test partition:
salloc --partition test --nodes=1 --cpus-per-task=2 --mem=12GB --time=00:30:00

# Check Python modules available on Cannon:
module spider python

# Get detailed information on specific module, e.g.:
module spider python/3.10.13-fasrc01

# Load the latest (usually also the default) Python module:
module load python
```

CLI Training:

https://docs.rc.fas.harvard.edu/wp-content/uploads/2013/10/Getting-started-on-FASRC-clusters-with-CLI.pptx.pdf

Training Material

https://docs.rc.fas.harvard.edu/kb/training-materials/

```
# Check current location & change if desired for this training: pwd
cd <desired-location>

# Clone FASRC User Codes repository:
https://github.com/fasrc/User_Codes/tree/master
SSH - git clone git@github.com:fasrc/User_Codes.git
HTTPS - git clone https://github.com/fasrc/User_Codes.git

# Create a training folder & go to that folder:
mkdir python-training
cd python-training

# Copy Python folders from the User Codes directory:
cp -r ../User_Codes/Languages/Python .
cp -r ../User_Codes/Parallel_Computing/Python/Python-Multiprocessing-Tutorial .
```

Python using CLI - Interactive

- Check Python version: `python --version`
- Invoke Python interpreter: `python`
- Execute Python programming interactively:

```
def square(x):
    """square a number"""
    return x ** 2

for N in range(1, 4):
    print(N, "squared is", square(N))
```

- Exit Python: `exit()`
- Or run a python script interactively: `python myscript.py`

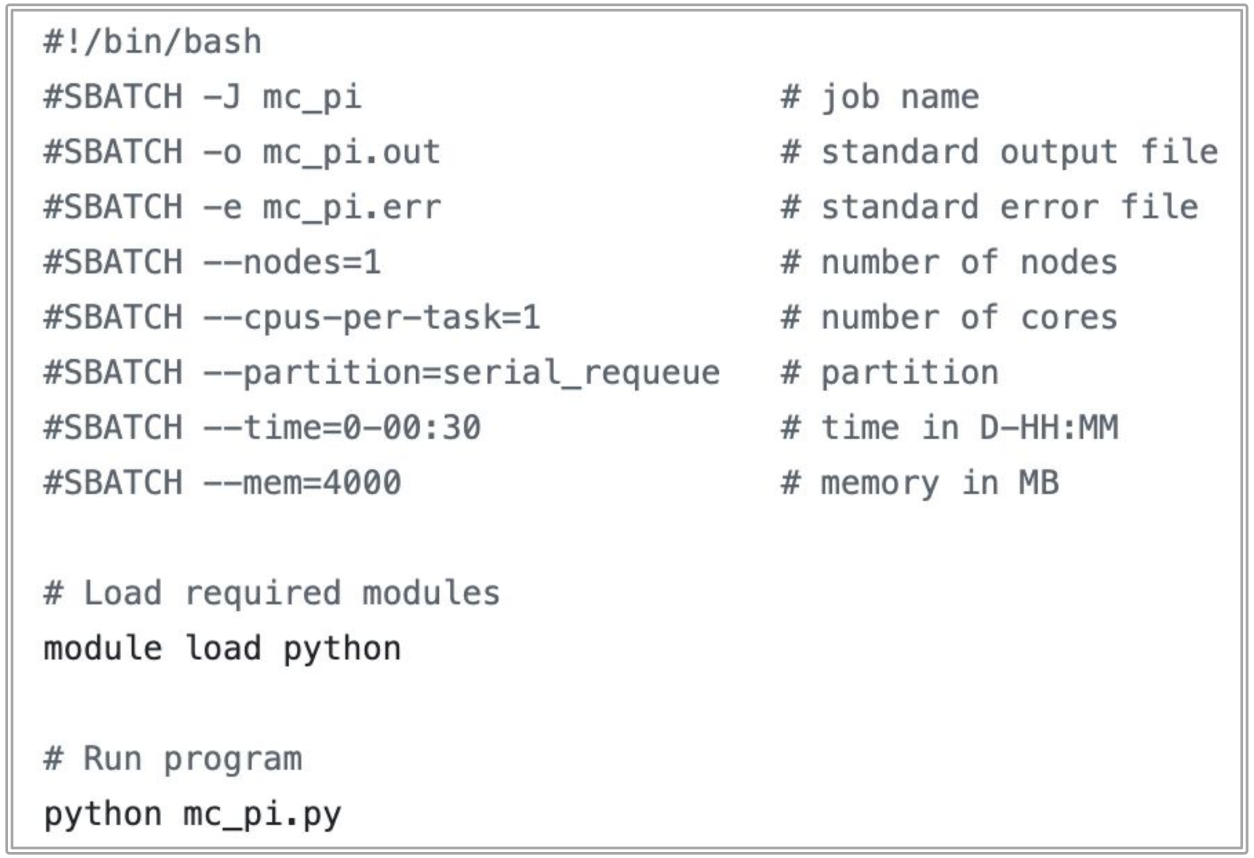
Python using CLI - sbatch; Example 1

https://github.com/fasrc/User_Codes/tree/master/Languages/Python/Example1

```
# Go to Example1 folder
cd Python/Example1
# Submit job
sbatch run.sbatch
```

run.sbatch: Batch-job submission script for queuing the job

mc_pi.py: Source code for calculating Pi using Monte-Carlo method



```
#!/bin/bash
#SBATCH -J mc_pi                            # job name
#SBATCH -o mc_pi.out                        # standard output file
#SBATCH -e mc_pi.err                        # standard error file
#SBATCH --nodes=1                           # number of nodes
#SBATCH --cpus-per-task=1                   # number of cores
#SBATCH --partition=serial_requeue          # partition
#SBATCH --time=0-00:30                      # time in D-HH:MM
#SBATCH --mem=4000                          # memory in MB

# Load required modules
module load python

# Run program
python mc_pi.py
```

Python Package Installation - Interactive

- Go to a compute node on the test partition:

```
salloc -p test --nodes=1 --cpus-per-task=2 --mem=12GB --time=01:00:00
```

- Create a vanilla mamba/conda environment (for multiprocessing exercise):

```
module load python
mamba create --prefix=/n/holylabs/LABS/<desired-folder>/multiproc_env
python=3.11 -y
```

- Alternatively, if default *$HOME* is desired, then do following instead:

```
module load python
conda create --name multiproc_env python=3.11 -y
```

- See Python Package Installation

Python Package Installation

- Activate conda/mamba environment:

```
mamba activate /n/holylabs/LABS/<desired-folder>/multiproc_env
```

- Or if $HOME used, then: `mamba activate multiproc_env`
- Install relevant python packages (Mamba recommended):

```
mamba install numpy pandas matplotlib -y
pip install jupyterlab swifter
```

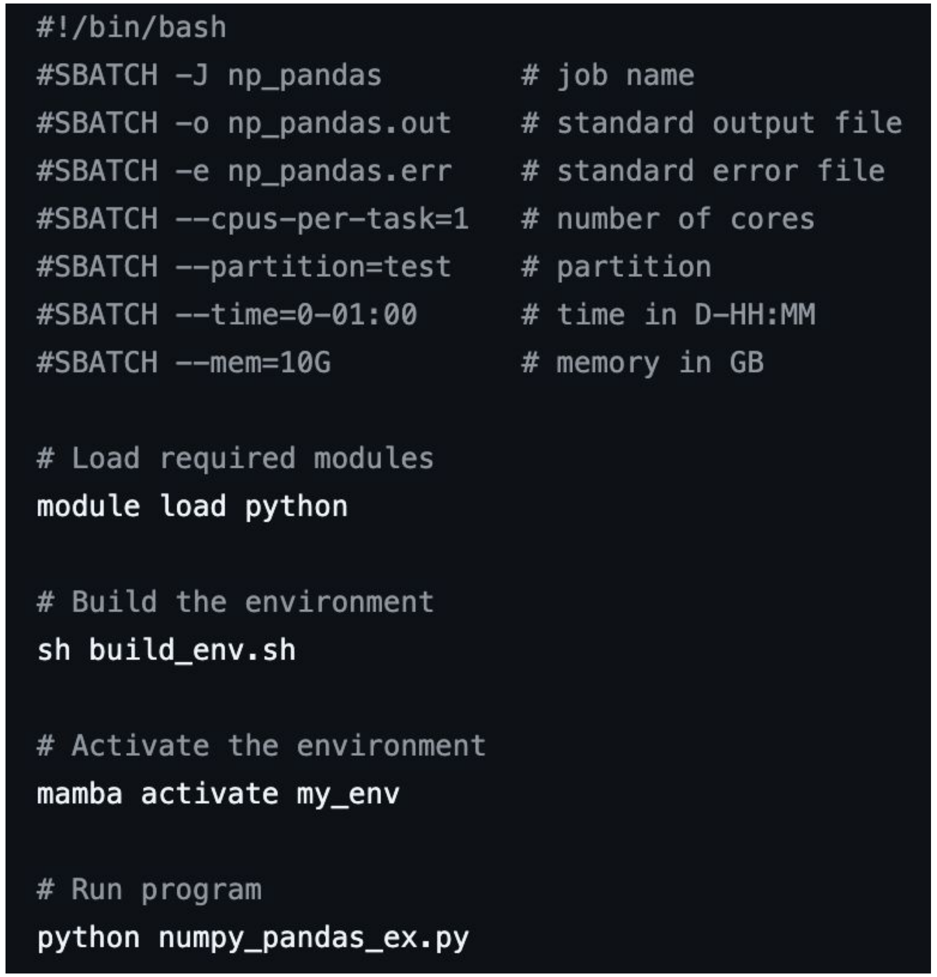
- Always pip install inside a conda environment to avoid package conflicts
- https://docs.rc.fas.harvard.edu/kb/python-package-installation/#Pip_Installs
- Deactivate the environment: `mamba deactivate`

Python Package Installation - sbatch

https://github.com/fasrc/User_Codes/tree/master/Languages/Python/Example2

```
# Go to Example2 folder
cd ../Python/Example2
# Submit job
sbatch run.sbatch
```

numpy_pandas_ex.py: source code for generating a dataframe utilizing a mamba environment



```
#!/bin/bash
#SBATCH -J np_pandas            # job name
#SBATCH -o np_pandas.out        # standard output file
#SBATCH -e np_pandas.err        # standard error file
#SBATCH --cpus-per-task=1       # number of cores
#SBATCH --partition=test        # partition
#SBATCH --time=0-01:00          # time in D-HH:MM
#SBATCH --mem=10G               # memory in GB

# Load required modules
module load python

# Build the environment
sh build_env.sh

# Activate the environment
mamba activate my_env

# Run program
python numpy_pandas_ex.py
```

Python Using Open OnDemand (OOD)

- Open-source web portal to access clusters
- Web-based, no software needs be installed on your local laptop/desktop (except for a modern browser like Google Chrome, Mozilla Firefox)
- Easy to learn and simple to use
- Very similar to desktop applications
- The easiest way to run GUI applications remotely on a cluster
- Safari is not recommended for OOD
- OOD Training: https://docs.rc.fas.harvard.edu/wp-content/uploads/2013/10/Getting-started-on-FASRC-clusters-with-OOD-May2024.pdf

How to access OOD on FASRC Clusters

- Accessing OOD from Cannon
  - Connect to FASRC VPN - Virtual Desktop (VDI) through Open OnDemand – FASRC DOCS
  - Then go to https://rcood.rc.fas.harvard.edu

- Accessing OOD from FASSE
  - Connect to FASSE VPN - FASSE VDI Apps – FASRC DOCS
  - Then go to https://fasseood.rc.fas.harvard.edu

FASSE proxy

Documentation: FASSE Proxy Settings – FASRC DOCS

- You may need to set FASSE proxy on
  - Firefox (web browsing)
  - Jupyter Notebook
  - Access Github
  - (Basically, anything outside of FASSE)

Filling a form to launch an app

- Request the resources that you need
  (If you don't know for a first trial run, use similar resources as your laptop/desktop)

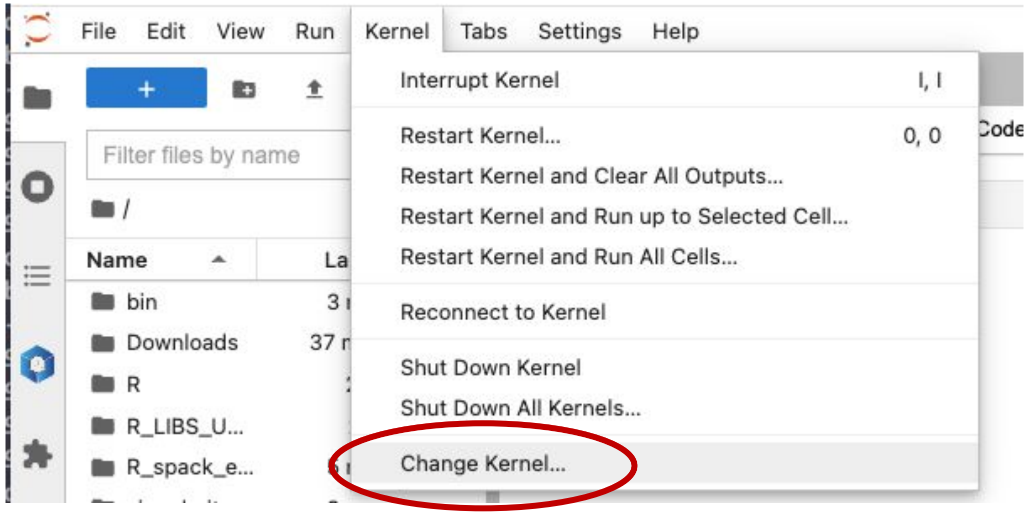
  - Partition (Name): depends on [Cannon](URL) (URL) vs [FASSE](URL) (URL)
  - Memory (RAM): amount of memory in GB
  - Number of cores: recommended at least 2
  - Number of GPUs: if >= 1, make sure you **select** a gpu partition
  - Allocated time: time you would like your session to run

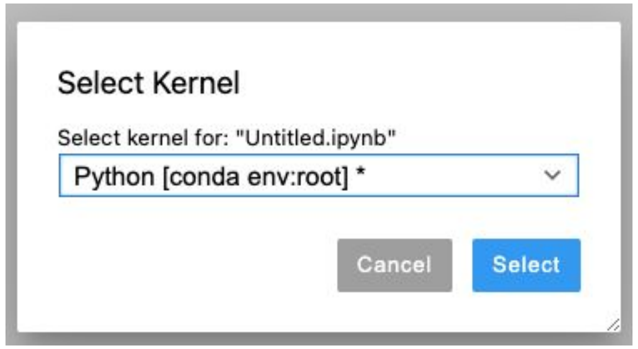
  the minimum and/or maximum values of each field depends on the selected partition

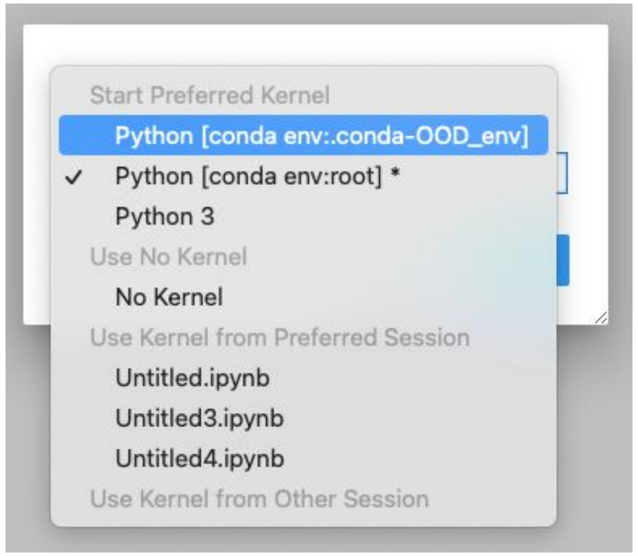
  - Email for status notification: to know when job starts, ends
  - Reservation: if you have a special reservation (this requires approval from FASRC)
  - Account: use this if you have more than one PI_lab affiliation

Jupyter Notebook

- Launch **new** Jupyter Notebook session (existing session will not work!)
- Select newly created conda environment as the kernel
  a. Open a notebook
  b. On the top menu, click Kernel -> Select Kernel -> Click on OOD_env
  c. Note: kernels is the same as conda, python, mamba environment




Select Kernel
Select kernel for: "Untitled.ipynb"
Python [conda env:root] *
Cancel
Select


Start Preferred Kernel
Python [conda env:.conda-OOD_env]
✓ Python [conda env:root] *
Python 3
Use No Kernel
No Kernel
Use Kernel from Preferred Session
Untitled.ipynb
Untitled3.ipynb
Untitled4.ipynb
Use Kernel from Other Session

Closing running OOD windows/tabs

- In most OOD apps, you can close the browser tab while the code is running, and the code will continue to run on the background
- Jupyter Notebook will not! The cell that is running will lose the data and output files will not be written
  - Solution: run Remote Desktop app and launch Jupyter Notebook from within Remote Desktop
  - Documentation: [Open OnDemand (OOD/VDI) Remote Desktop: How to open software – FASRC DOCS]()

FASRC documentation

- FASRC docs: https://docs.rc.fas.harvard.edu/
- FASRC Python docs:
  - https://docs.rc.fas.harvard.edu/kb/python/
  - https://docs.rc.fas.harvard.edu/kb/python-package-installation/
- GitHub User_codes: https://github.com/fasrc/User_Codes/
- Getting help
  - Office hours: https://www.rc.fas.harvard.edu/training/office-hours/
  - Ticket
    - Portal: http://portal.rc.fas.harvard.edu/rcrt/submit_ticket (requires login)
    - Email: rchelp@rc.fas.harvard.edu

FASRC Upcoming Trainings

Training calendar: https://www.rc.fas.harvard.edu/upcoming-training/

Python Multiprocessing on the FASRC cluster

Training is focused on some of the techniques to accelerate Python programming with emphasis on utilizing multiprocessing with numpy arrays.

Audience: Users who are familiar with basic Python, command line, HPC systems, and have attended our Python Basics on FASRC clusters training.

Note: All topics below are a brief overview to utilizing multiprocessing on FASRC clusters.

Objectives:
1. Understanding Multiprocessing
2. Executing Multiprocessing on FASRC clusters

Survey

Please, fill out our course survey. Your feedback is essential for us to improve our trainings!!

http://tinyurl.com/FASRCsurvey

Thank you :)

FAS Research Computing