



Using Python

at the Center for High Performance Computing

Robben Migacz
Scientific Consultant

Based largely on work by Brett Milash



☰ Scope of this presentation

This presentation is an **overview of strategies for using Python on systems at the CHPC**. It is not an introduction to the Python language itself; for this, see the CHPC's *Introduction to Python* series.

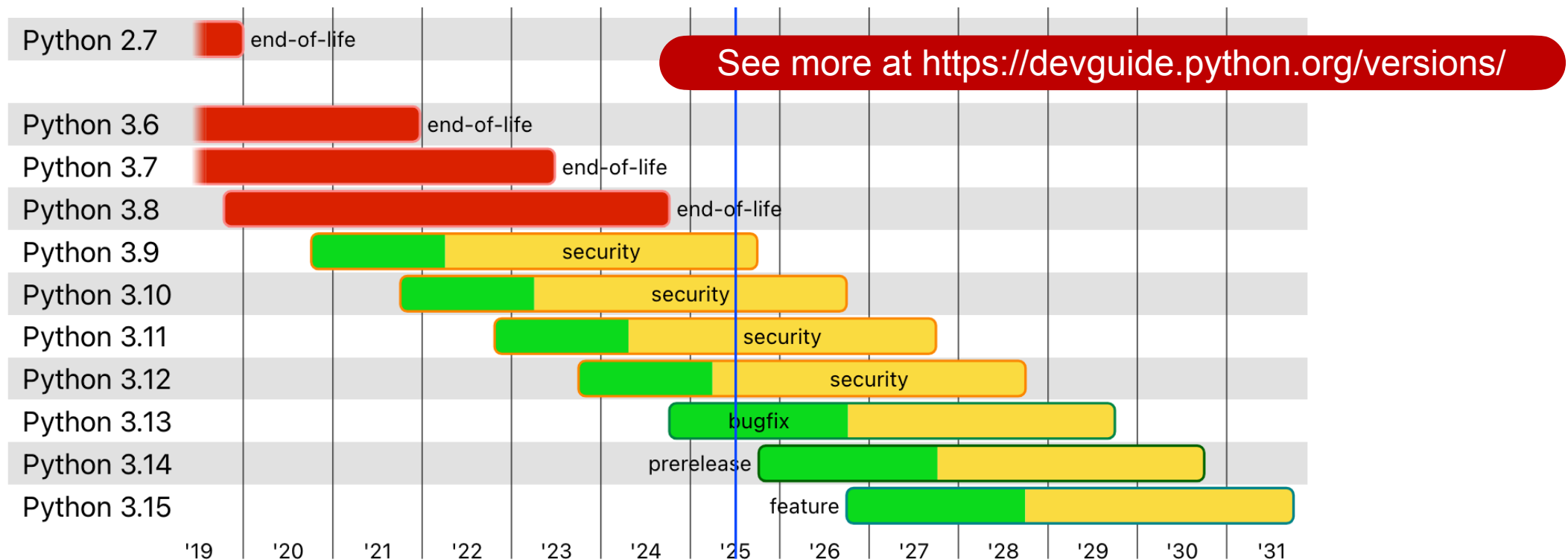
In this presentation, we'll focus on running Python on the Linux clusters, which represent the majority of computational resources at the CHPC.

Q1

**To use Python effectively at the CHPC,
which version should I use?**

Python versions

Python versions are generally supported for five years. We recommend using a recent version when you are using CHPC systems. (We'll discuss this soon.)



≡ Python versions

When you first log in to a CHPC system from the command line, the Python interpreters available to you are the system versions, which can be quite old. Please keep this in mind if you're running Python from the command line.

```
└─ python3 --version
```

Python 3.6.8



This version of Python is considered *end-of-life*

```
└─ python2 --version
```

Python 2.7.18



This version of Python is considered *end-of-life*

≡ Python versions

If you're running Python from the command line, *use a module!* If you're using Open OnDemand, you'll be able to select a version.

```
❏ module spider python
```

```
:
```

```
python/2.7.18  
python/3.6.8  
python/3.8.8  
python/3.9.15  
python/3.10.3  
python/3.11.3  
python/3.11.7-spack  
python/3.11.7  
python/3.12.4
```

```
:
```

```
❏ module load python/3.12.4
```

```
❏ python --version
```

```
Python 3.12.4
```

≡ Python versions

There are also Python modules with machine learning packages already installed and ready to use.

<pre>└─ module spider deeplearning : deeplearning/2023.3 deeplearning/2024.1 deeplearning/2024.2.0 deeplearning/2025.4 :</pre>	<pre>└─ module load deeplearning/2025.4 └─ python --version Python 3.11.11</pre>
--	---

This module loads Python with common machine learning packages.

[Read more on the CHPC documentation →](#)



Q2

**I have specific software requirements;
how do I install packages?**



Python packages

Users can install their own packages on CHPC systems. We suggest using virtual environments or conda environments to install your own Python packages.

Photo by Luke Heibert (Unsplash License)



Python packages: virtual environments

Virtual environments (venv) allow users to install their own Python packages with pip. They are a built-in part of the Python language.

Use a virtual environment if *all* software can be installed with pip.

[Read more on the CHPC documentation →](#)

Python packages: conda environments

Conda environments allow users to install software in isolated environments. This includes versions of Python; Python packages; and other software, such as R dependencies commonly used in bioinformatics software.

If *any* dependency requires conda, use a conda environment for *all* dependencies.

[Read more on the CHPC documentation →](#)

[Advanced users: Self-installed conda →](#)

Q3

**To use Python effectively at the CHPC,
where should I run my scripts?**

☰ High-performance computing clusters

If you haven't seen the *Introduction to the CHPC* presentation, we recommend viewing a recording or reading through the slides.

Here, we'll briefly discuss the architecture of high-performance computing clusters, which is important to keep in mind when using CHPC systems.

[Introduction to the CHPC presentation materials →](#)



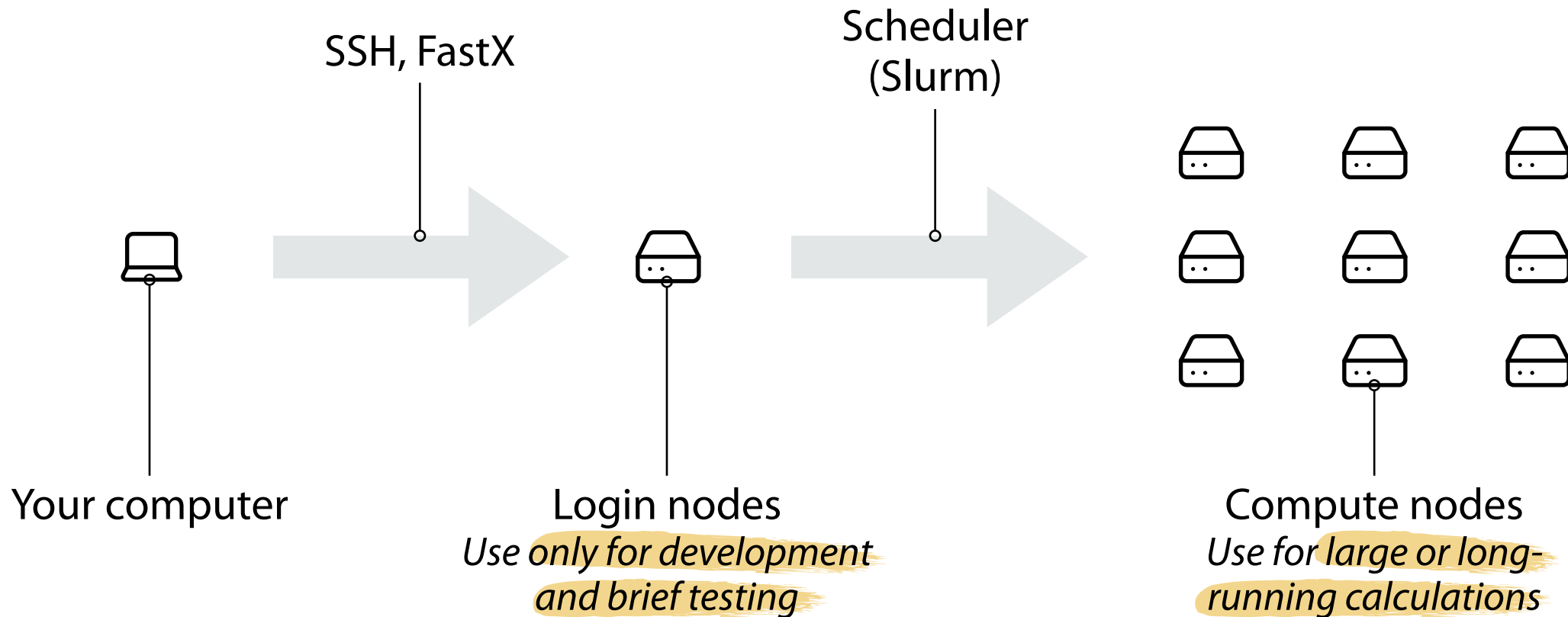
☰ High-performance computing clusters

High-performance computing clusters are *not* individual computers with lots of resources. They're composed of *many* computers, often called *nodes*.

The “front doors” to HPC clusters are *login nodes*, which are not suitable for large or long-running computations.

Photo by Sam Liston (CHPC)

High-performance computing clusters



There are many ways to use Python at the Center for High Performance Computing.
In this presentation, we'll focus on the most effective strategies.



Command line (SSH)



FastX



Open OnDemand



Running Python on login nodes



Open OnDemand



Command line (salloc)

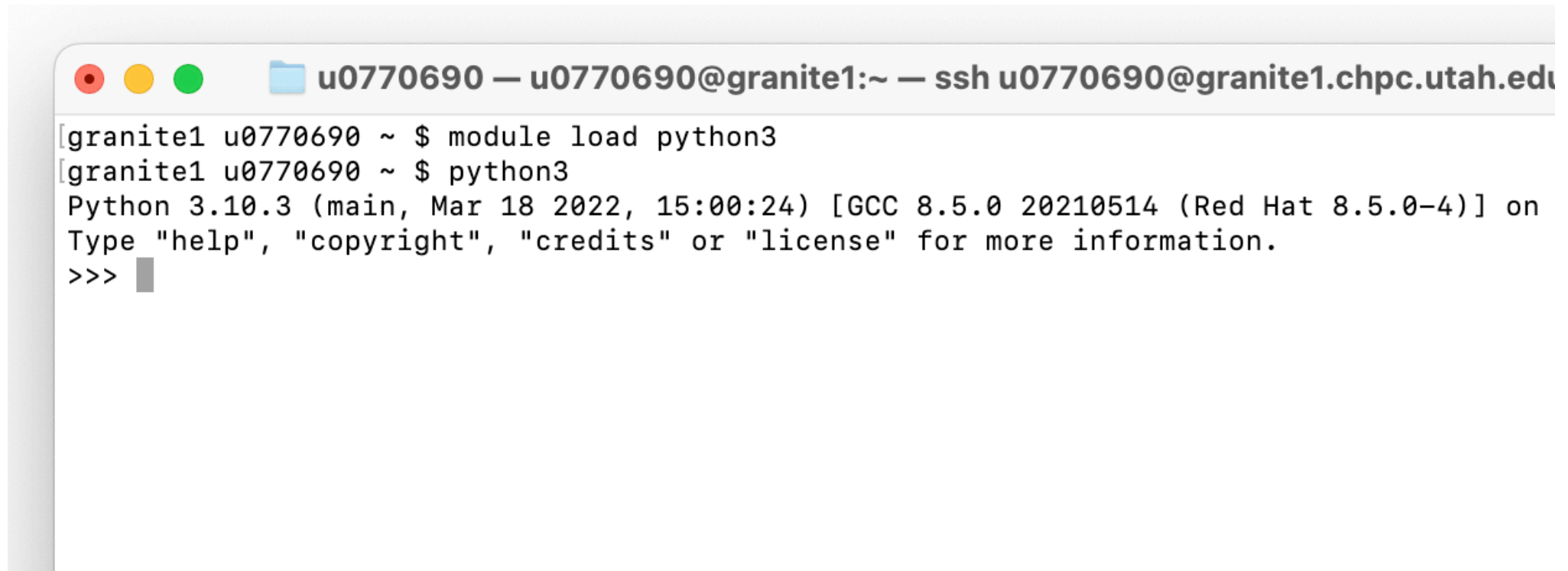


Batch scripts (sbatch)



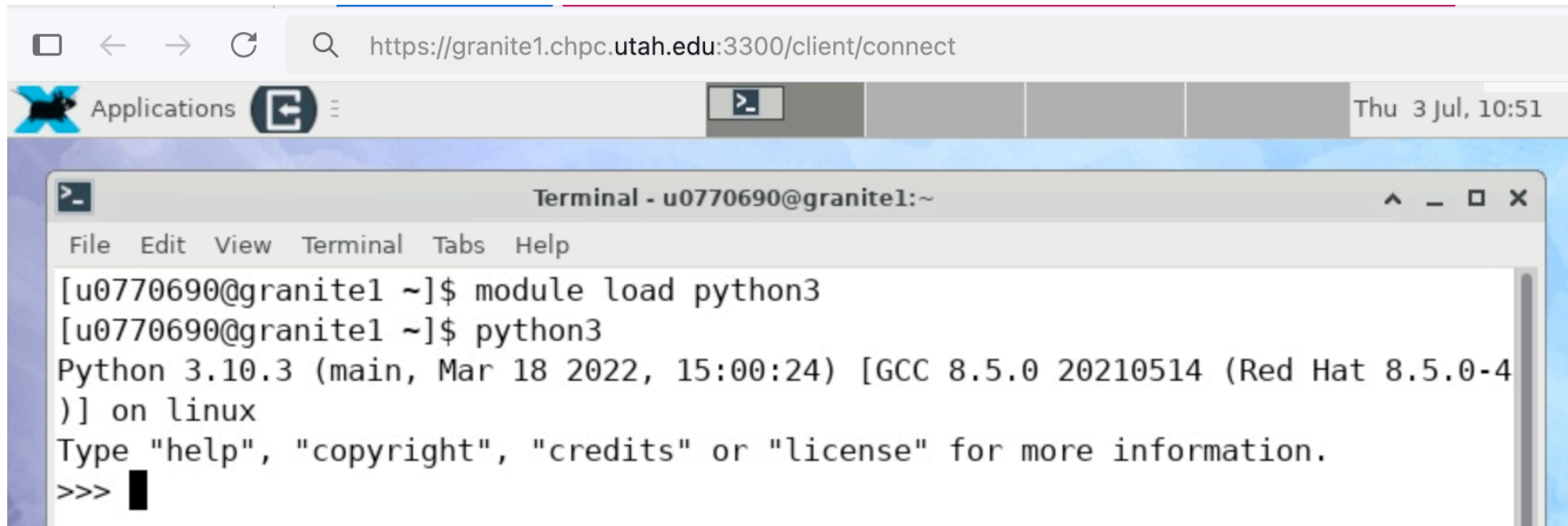
Running Python on compute nodes

You can connect to a login node with SSH, load a Python module, and run the Python interpreter to work interactively or run scripts.

A terminal window with a title bar showing a folder icon and the text "u0770690 — u0770690@granite1:~ — ssh u0770690@granite1.chpc.utah.edu". The terminal content shows a user logging in, loading the python3 module, and starting the Python 3.10.3 interpreter. The prompt changes from a shell prompt to a Python prompt. The window has standard macOS-style window controls (red, yellow, green buttons) on the left.

```
[granite1 u0770690 ~ $ module load python3
[granite1 u0770690 ~ $ python3
Python 3.10.3 (main, Mar 18 2022, 15:00:24) [GCC 8.5.0 20210514 (Red Hat 8.5.0-4)] on
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

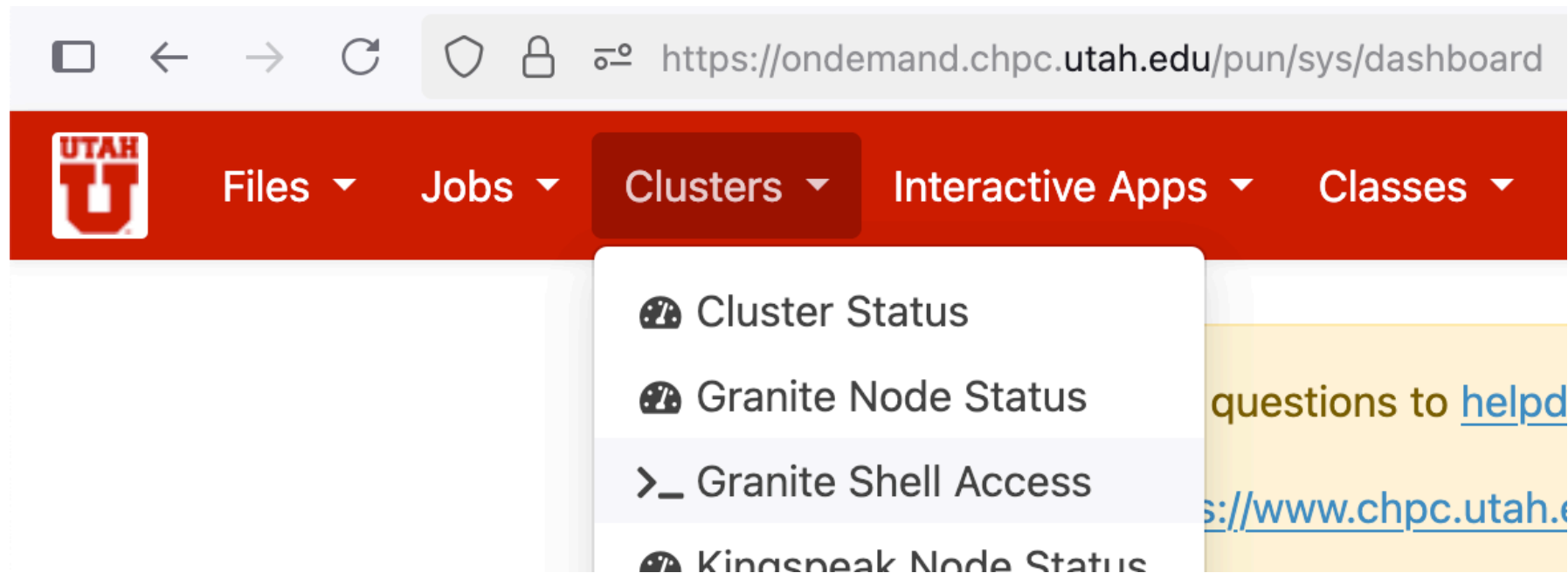
You can also use FastX to connect to a login node. This will allow you to **work with a desktop environment** and **maintain persistent sessions**. You can use FastX from a web browser or a client you install on your own computer.

A screenshot of a web browser window displaying a FastX terminal session. The browser's address bar shows the URL `https://granite1.chpc.utah.edu:3300/client/connect`. The browser's top bar includes navigation icons, the text "Applications", and a system clock showing "Thu 3 Jul, 10:51". The terminal window, titled "Terminal - u0770690@granite1:~", contains the following text:

```
[u0770690@granite1 ~]$ module load python3
[u0770690@granite1 ~]$ python3
Python 3.10.3 (main, Mar 18 2022, 15:00:24) [GCC 8.5.0 20210514 (Red Hat 8.5.0-4)] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Running Python on login nodes

You can use a web browser to quickly access a terminal on a login node through “Shell Access” on Open OnDemand.



Running Python on compute nodes

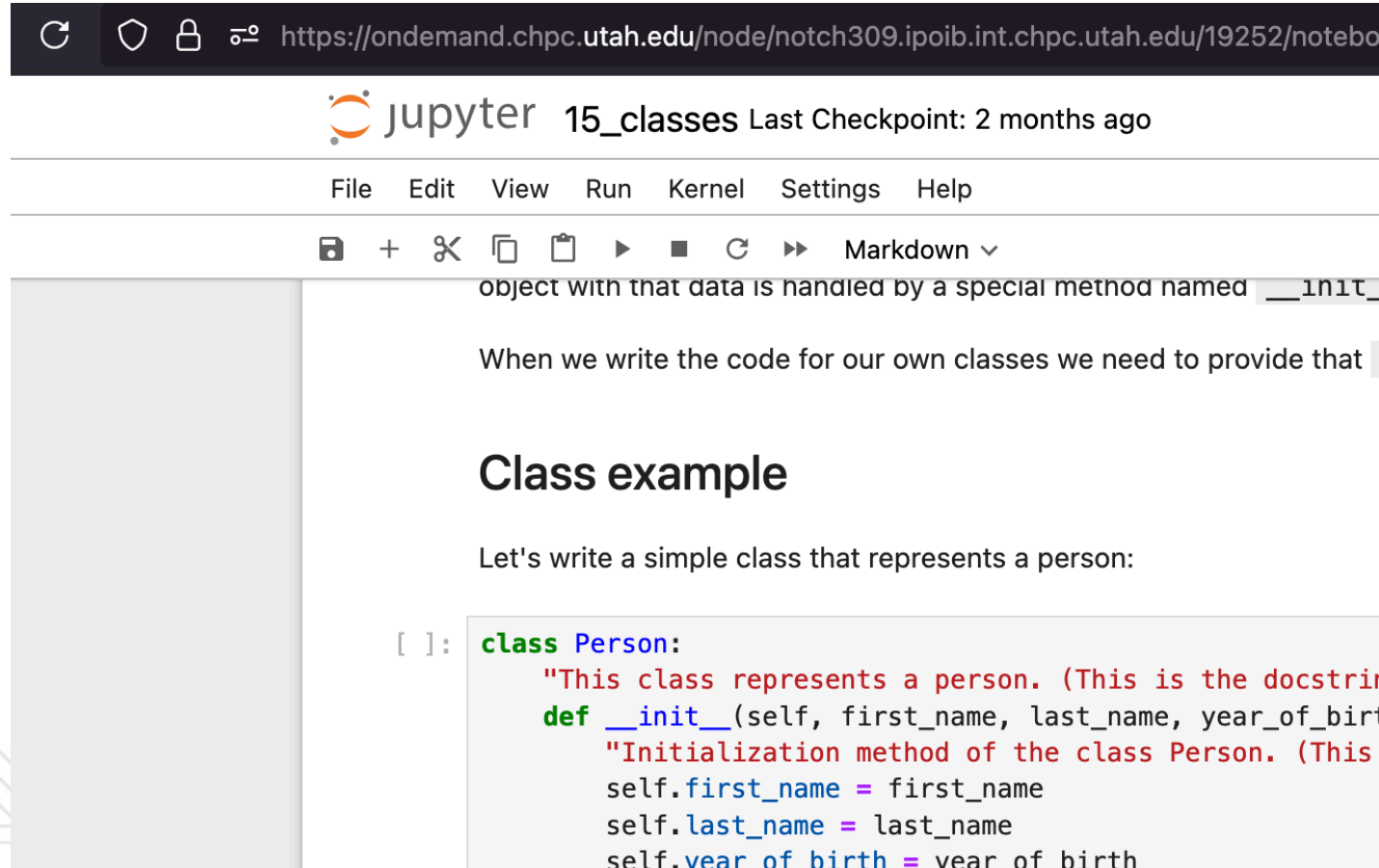
Running on compute nodes will require interaction with the scheduler, Slurm. If you're not familiar with Slurm, we recommend reading through our documentation and attending the introductory Slurm presentations.

[Read more on the CHPC documentation →](#)

[Information about presentations →](#)

Running Python on compute nodes

Open OnDemand allows you to start jobs on compute nodes from your web browser. We recommend using Jupyter Notebooks in Open OnDemand if you are using Python interactively.



object with that data is handled by a special method named `__init__`

When we write the code for our own classes we need to provide that

Class example

Let's write a simple class that represents a person:

```
[ ]: class Person:
    "This class represents a person. (This is the docstring)
    def __init__(self, first_name, last_name, year_of_birth)
        "Initialization method of the class Person. (This
        self.first_name = first_name
        self.last_name = last_name
        self.year of birth = year of birth
```


My Interactive Sessions / Jupyter

Jupyter

This app will launch a [Jupyter](#) Notebook or Lab server using [Python](#) on a [HPC cluster](#) or on a [Frisco node](#).

To start the job promptly, use notchpeak-shared-short account and partition on the [Notchpeak cluster](#).

[GPU specification](#) is optional for the clusters and partitions that have them.

Jupyter interface

Notebook

This defines the interface of Jupyter you want to start (Notebook or Lab).

Jupyter Python version

Custom (Environment Setup below)

This defines the Python distribution of Jupyter you want to start.

Environment Setup for Custom Python

Enter commands (module load, source activate, etc) to create your desired Jupyter environment; jupyter MUST be on your path and either notebook or jupyterlab installed in your Python environment. For instructions how to install your own Python using Miniconda see <https://www.chpc.utah.edu/documentation/software/python-anaconda.php>.

Cluster

Open OnDemand allows you to start a Slurm job by selecting parameters from a menu.

If you are using a virtual environment or conda environment with a Jupyter Notebook, make sure you choose the “Custom” Python version. Enter the commands you’d use to access your environment in “Environment Setup.”

The “jupyter” package must be installed in your environment for this to work.

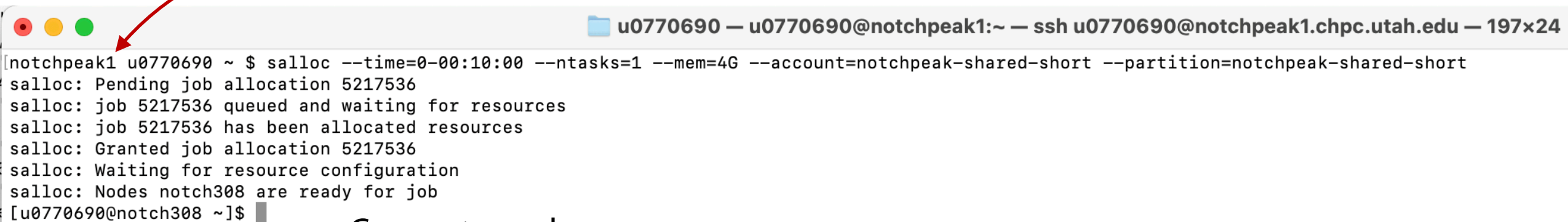


Running Python on compute nodes

You can use the “salloc” command to request computational resources from the command line on a login node.

Watch for a change in your prompt when resources are allocated.

Login node



```
u0770690 — u0770690@notchpeak1:~ — ssh u0770690@notchpeak1.chpc.utah.edu — 197x24
[notchpeak1 u0770690 ~ $ salloc --time=0-00:10:00 --ntasks=1 --mem=4G --account=notchpeak-shared-short --partition=notchpeak-shared-short
salloc: Pending job allocation 5217536
salloc: job 5217536 queued and waiting for resources
salloc: job 5217536 has been allocated resources
salloc: Granted job allocation 5217536
salloc: Waiting for resource configuration
salloc: Nodes notch308 are ready for job
[u0770690@notch308 ~]$
```

Compute node

Once you are on a compute node, you can load and run Python without worrying about resource consumption

Running Python on compute nodes

You can use the “sbatch” command to submit a script that will run on a compute node once resources are available. This is the best way to run Python scripts that do not require user interaction.

Slurm job
parameters

slurm_script.sh

```
#!/usr/bin/env bash

#SBATCH --time=0-00:10:00
#SBATCH --ntasks=1
#SBATCH --mem=4G
#SBATCH --clusters=notchpeak
#SBATCH --account=notchpeak-shared-short
#SBATCH --partition=notchpeak-shared-short

module load python3
python3 my_script.py
```

```
>- sbatch slurm_script.sh
```



Summary of strategies

for using Python effectively at the CHPC

Strategies for running Python on login nodes

	Mechanism for running Python				
Mechanism for accessing CHPC resources	Interactive (command line)	Script	Jupyter Notebook	Remote Editing (VSCode and others)	Workflows (Snakemake and others)
SSH (graphics support with X forwarding)	◆ Ideal	◆ Ideal	✗ Not recommended	✗ Not recommended	◆ Ideal
FastX (graphics support in desktop sessions)	◆ Ideal	◆ Ideal	✓ Possible (in web browser on remote host)	✗ Not recommended	◆ Ideal
Open OnDemand (Shell Access)	✓ Possible (lacks support for graphics)	✓ Possible (lacks support for graphics)	✗ Not recommended	✗ Not recommended	✓ Possible (lacks support for graphics)

Strategies for running Python on compute nodes

	Mechanism for running Python				
Mechanism for accessing CHPC resources	Interactive (command line)	Script	Jupyter Notebook	Remote Editing (VSCode and others)	Workflows (Snakemake and others)
Batch jobs: sbatch (from login node)	✗ Not recommended	♦ Ideal	✗ Not recommended	✗ Not recommended	♦ Ideal
Interactive jobs: salloc (from login node)	♦ Ideal	✓ Possible (if no user interaction is necessary, use a script instead)	✗ Not recommended	✓ Possible (start a job, then connect to the compute node from your local computer ¹)	✓ Possible (workflows often work well as scripts, for which a batch job may be more appropriate)
Open OnDemand (Interactive Apps or Job Composer)	✓ Possible (use Interactive Apps → Interactive Desktop, then run Python)	✓ Possible (use Jobs → Job Composer to submit a job as you would from the command line)	♦ Ideal (Interactive Apps → Jupyter)	✓ Possible (start a job with Jobs → VSCode Server, then connect to the compute node from your local computer ¹)	✗ Not recommended

¹This will require using an SSH tunnel, as described on the Visual Studio Code documentation on the CHPC website

Do you have any questions?

If we don't have time to answer your question, or if you think of any questions after the presentation, please reach out to us! helpdesk@chpc.utah.edu

