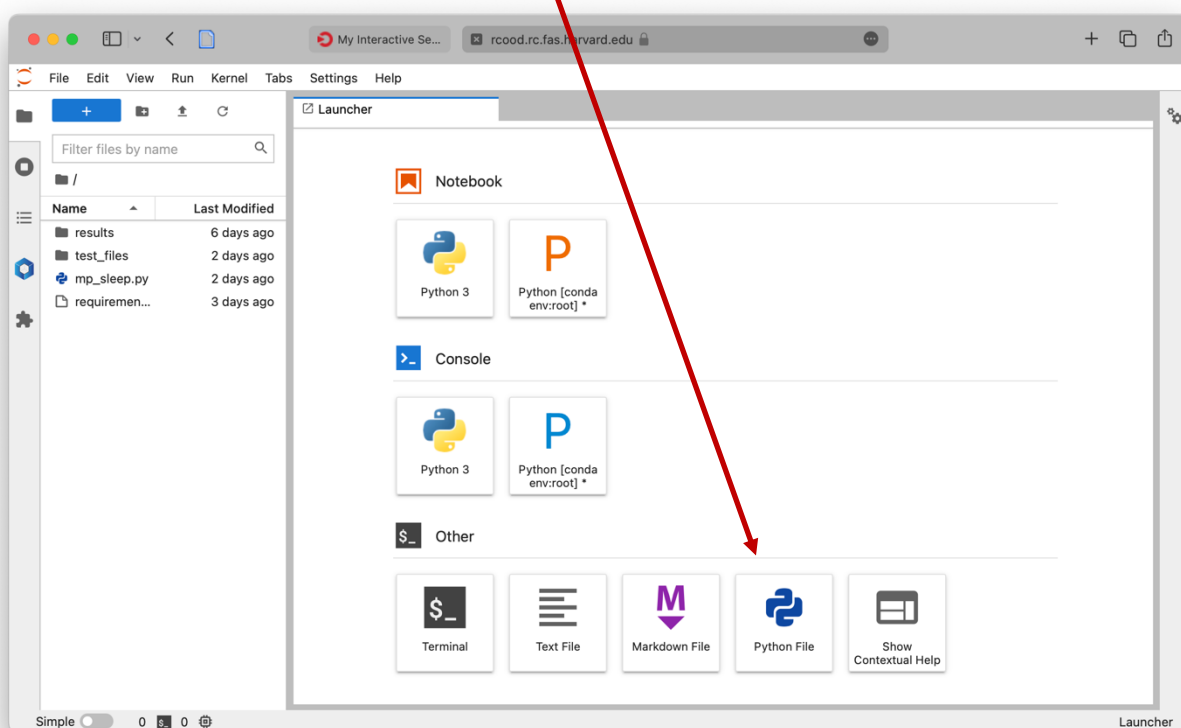


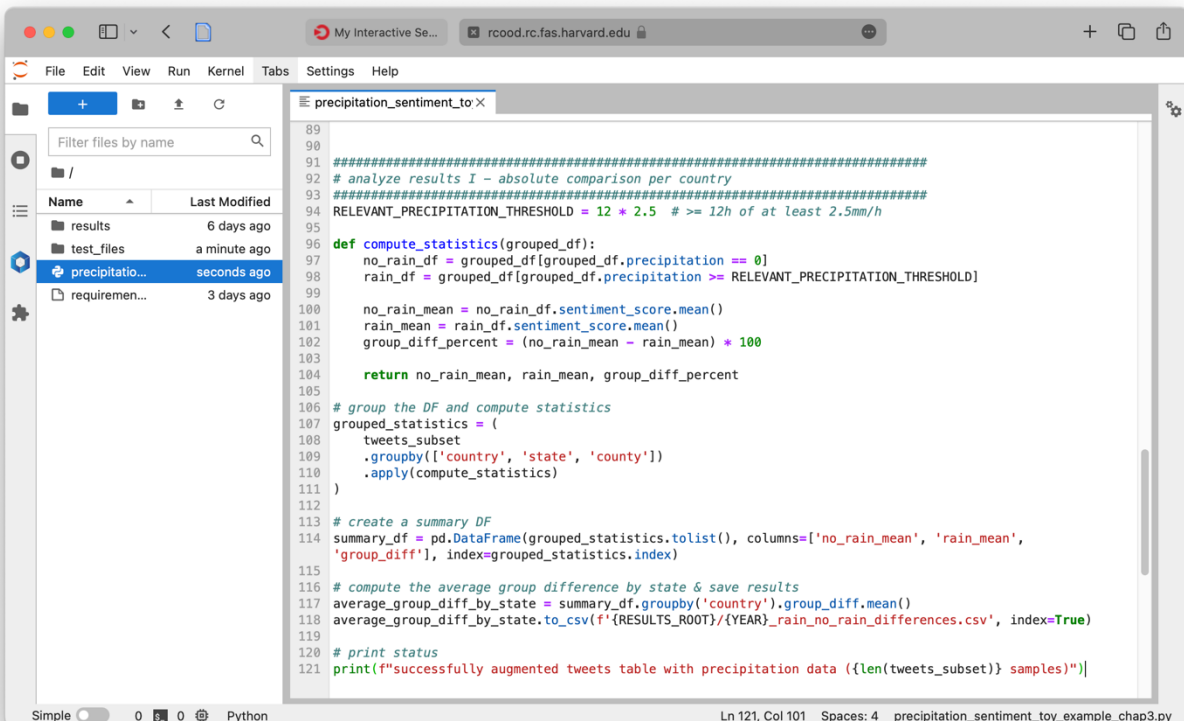
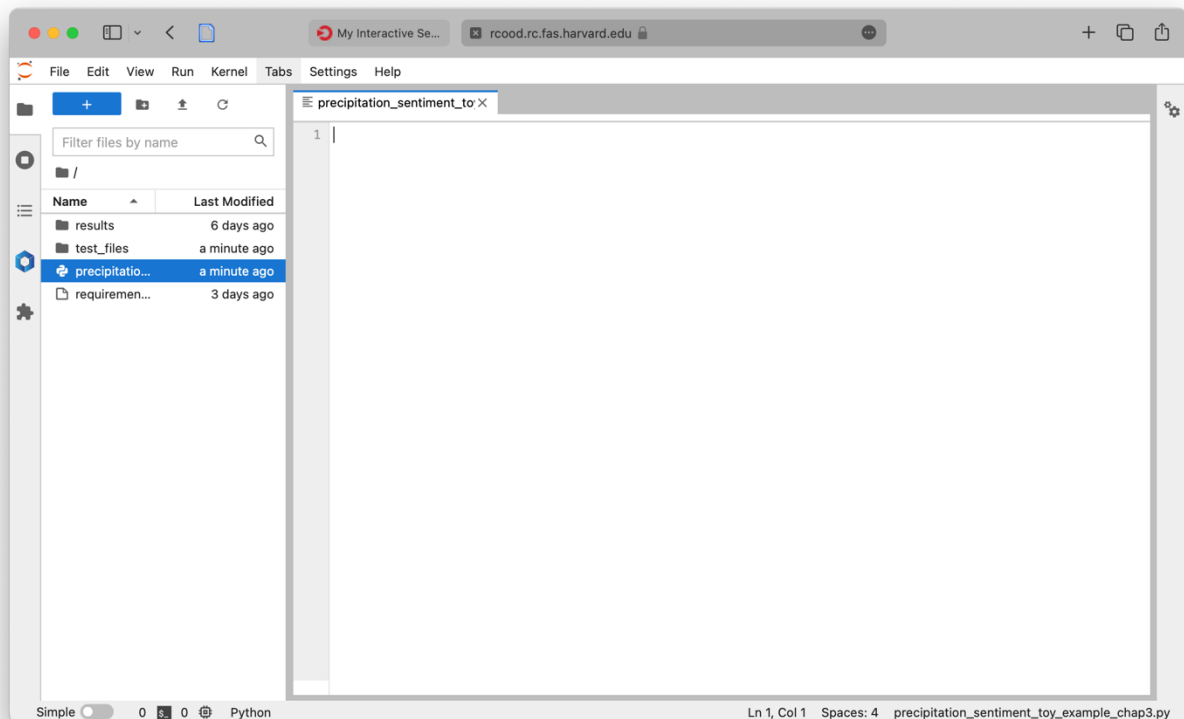
Exercise 3

1. Run your Python program on a compute-node

Start a new Jupyter session via the FASRC web interface, see Exercise 1 for help. In the “Launcher”, in section “Other”, select “Python file”:

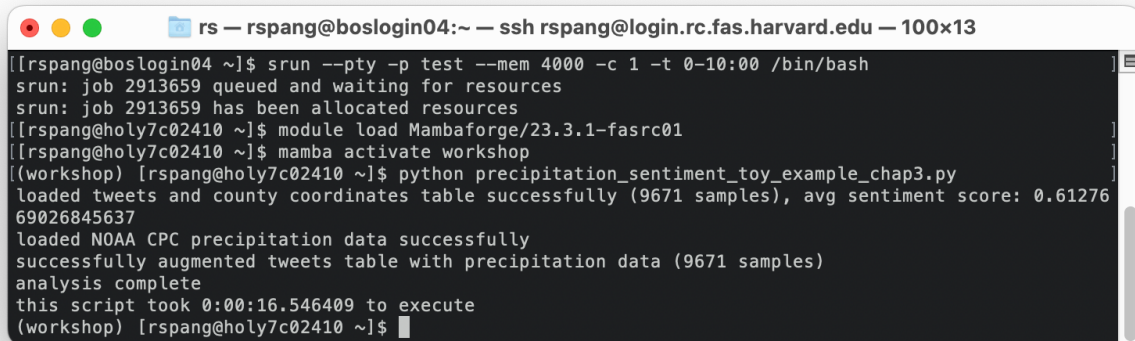


Rename the new file and copy your modified code into the window:



This is a convenient way to update your code quickly. Make sure you update the paths to the datasets: `/n/holyscratch01/cga/python-workshop-gis-big-data`

To run the code, use a SSH connection to a login-node and request a new interactive session. Since we now work with real data, make sure to allocate enough memory. For the example, 4GB are sufficient. `srun --pty -p test --mem 4000 -c 1 -t 0-10:00 /bin/bash`



```
rs — rspang@boslogin04:~ — ssh rspang@login.rc.fas.harvard.edu — 100x13
[[rspang@boslogin04 ~]$ srun --pty -p test --mem 4000 -c 1 -t 0-10:00 /bin/bash
srun: job 2913659 queued and waiting for resources
srun: job 2913659 has been allocated resources
[[rspang@holy7c02410 ~]$ module load Mambaforge/23.3.1-fasrc01
[[rspang@holy7c02410 ~]$ mamba activate workshop
((workshop) [rspang@holy7c02410 ~]$ python precipitation_sentiment_toy_example_chap3.py
loaded tweets and county coordinates table successfully (9671 samples), avg sentiment score: 0.61276
69026845637
loaded NOAA CPC precipitation data successfully
successfully augmented tweets table with precipitation data (9671 samples)
analysis complete
this script took 0:00:16.546409 to execute
(workshop) [rspang@holy7c02410 ~]$
```

1.2 Add time tracking

To monitor how long the execution of a segment in your code takes, you can take the time before the segment in question (or the entire code), and after. Then, you subtract the first timestamp from the second one to compute the difference. For example:

```
from datetime import datetime

# take start time
start_execution_timestamp = datetime.now()

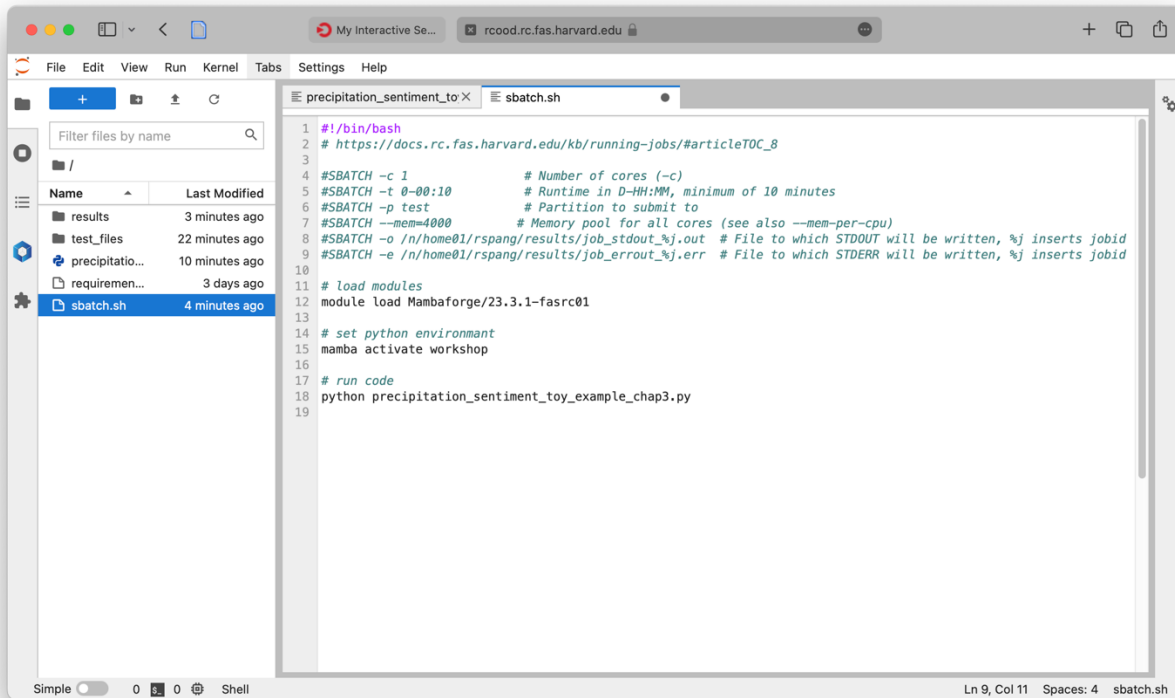
a_long_running_task() # replace this with your actual code

# take end time
end_execution_timestamp = datetime.now()
time_diff = end_execution_timestamp - start_execution_timestamp

# print status
print(f"this script took {time_diff}sec to execute")
```

2. Run your Python program on a compute-node as a job

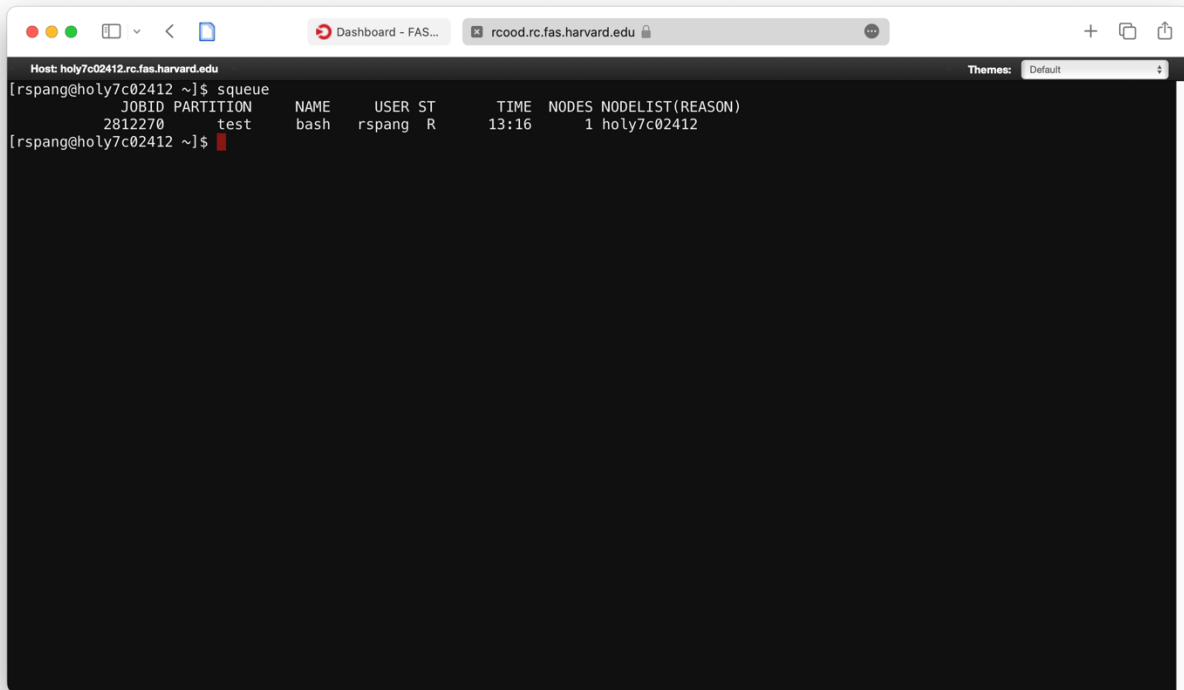
Create a sbatch script to run your Python program. This sbatch script can then be submitted as a SLURM job. The GitHub repo provides you with a template, the cheat sheet also explains the structure of a sbatch script.

A screenshot of a web-based code editor interface. The left sidebar shows a file explorer with a tree view containing folders like 'results' and 'test_files', and files like 'precipitation...', 'requiremen...', and 'sbatch.sh'. The 'sbatch.sh' file is selected. The main editor area displays the content of 'sbatch.sh', which is a SLURM batch script. The script includes comments for each line, explaining the purpose of the directives. The script starts with a shebang, followed by a URL to a documentation page. It then sets SLURM parameters: number of cores (-c 1), runtime (-t 0-00:10), partition (-p test), memory pool (--mem=4000), and output files (-o and -e). It then loads the 'Mambaforge/23.3.1-fasrc01' module, sets the Python environment with 'mamba activate workshop', and finally runs the Python script 'precipitation_sentiment_toy_example_chap3.py'. The status bar at the bottom indicates 'Ln 9, Col 11 Spaces: 4 sbatch.sh'.

Once created, the job can be submitted (on a login-node): `sbatch sbatch.sh`

3. Monitor jobs

The SLURM command `squeue` returns a list of all currently running jobs. This will return (at least) the one interactive session you used to start the python program.

A terminal window with a dark background. The title bar shows "Dashboard - FAS..." and "rcood.rc.fas.harvard.edu". The terminal content shows the output of the 'squeue' command, displaying a table of job information. The first row of the table is highlighted in red.

```
Host: holy7c02412.rc.fas.harvard.edu
[rsbang@holy7c02412 ~]$ squeue
      JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      2812270      test      bash     rsbang  R       13:16        1 holy7c02412
[rsbang@holy7c02412 ~]$
```

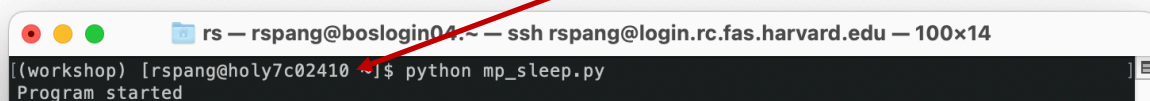
3.1 Monitor the execution of interactive shells

An interactive shell connection runs in your terminal; while a program is running, you cannot use the same terminal for different purposes. However, you can start a second terminal and start a second SSH connection. While this will allow you to run SLURM monitoring tools, it is not guaranteed that you will be connected to the same node if you request a second (interactive) session with SLURM.

However, there is a web-tool that allows you to connect to the same instance from your browser! Open the following link

`https://rcood.rc.fas.harvard.edu/pun/sys/shell/ssh/COMPUTE_NODE.rc.fas.harvard.edu`

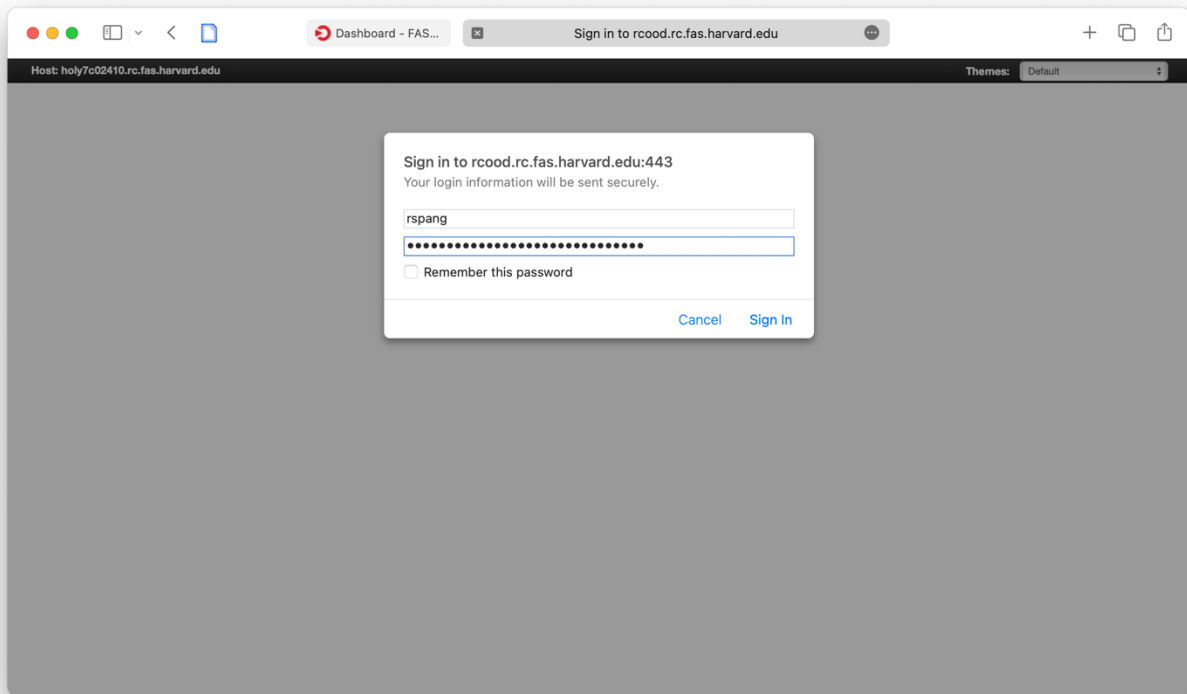
in a browser; replace “COMPUTE_NODE” with the node-ID you are connected to.

A terminal window with a light background. The title bar shows "rs — rsbang@boslogin04.~ — ssh rsbang@login.rc.fas.harvard.edu — 100x14". The terminal content shows the user running a python script.

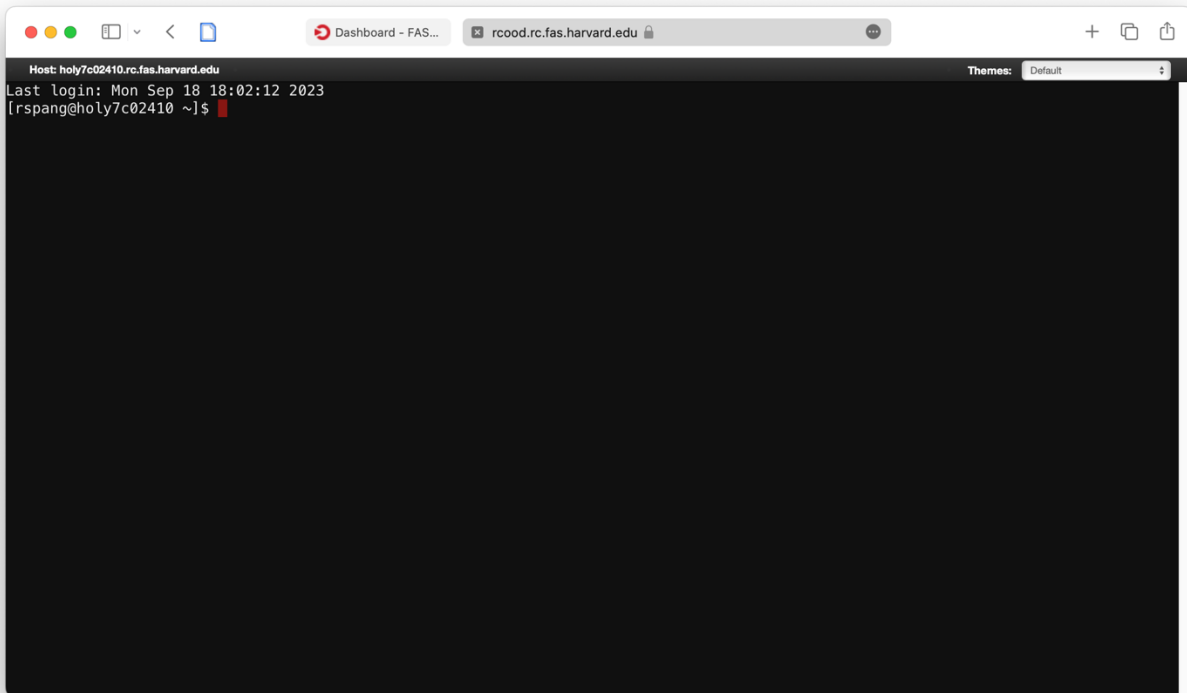
```
rs — rsbang@boslogin04.~ — ssh rsbang@login.rc.fas.harvard.edu — 100x14
[(workshop) [rsbang@holy7c02410 ~]$ python mp_sleep.py
Program started
```

In this example, the node-ID is be “holy7c02410”.

If you are asked to login first, provide your FASRC credentials:



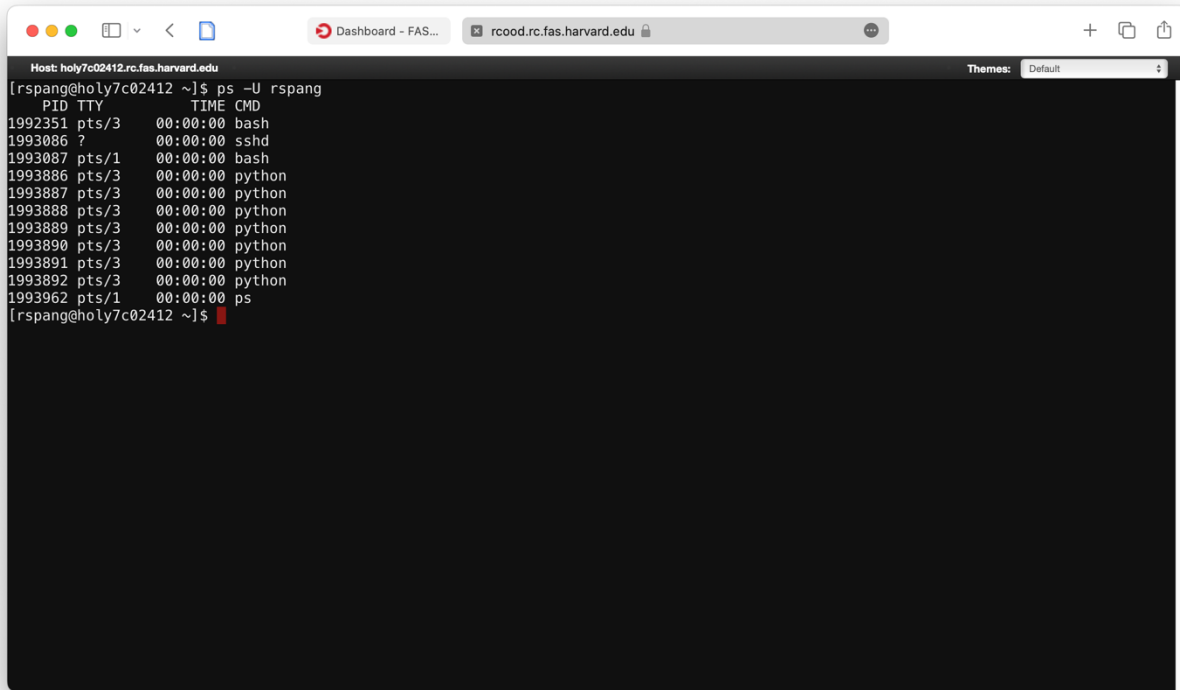
Then, a terminal-like web page shows up.



Here, you can run all commands as you would in a normal terminal. For example, you can monitor how many processes you are currently running. Use the following command to obtain a list of all processes in your name:

```
ps -U username
```

Replace “username” with your FASRC username.



The screenshot shows a terminal window with a dark background. The prompt is [rspang@holy7c02412 ~]\$. The command ps -U rspang has been executed, resulting in a table of processes. The table has four columns: PID, TTY, TIME, and CMD. The processes listed are: bash (PID 1992351), sshd (PID 1993086), bash (PID 1993087), python (PID 1993886), python (PID 1993887), python (PID 1993888), python (PID 1993889), python (PID 1993890), python (PID 1993891), python (PID 1993892), and ps (PID 1993962). The terminal window is titled 'Host: holy7c02412.rc.fas.harvard.edu' and 'Themes: Default'.

```
Host: holy7c02412.rc.fas.harvard.edu
[rspang@holy7c02412 ~]$ ps -U rspang
  PID TTY          TIME CMD
1992351 pts/3    00:00:00 bash
1993086 ?        00:00:00 sshd
1993087 pts/1    00:00:00 bash
1993886 pts/3    00:00:00 python
1993887 pts/3    00:00:00 python
1993888 pts/3    00:00:00 python
1993889 pts/3    00:00:00 python
1993890 pts/3    00:00:00 python
1993891 pts/3    00:00:00 python
1993892 pts/3    00:00:00 python
1993962 pts/1    00:00:00 ps
[rspang@holy7c02412 ~]$
```

What you learned in this exercise:

- How to create a python script using the Jupyter web interface
- How to create a sbatch script and how to submit a SLURM job
- How to start a browser-based shell to connect to an ongoing (interactive) session
- How to monitor processes and jobs while running