

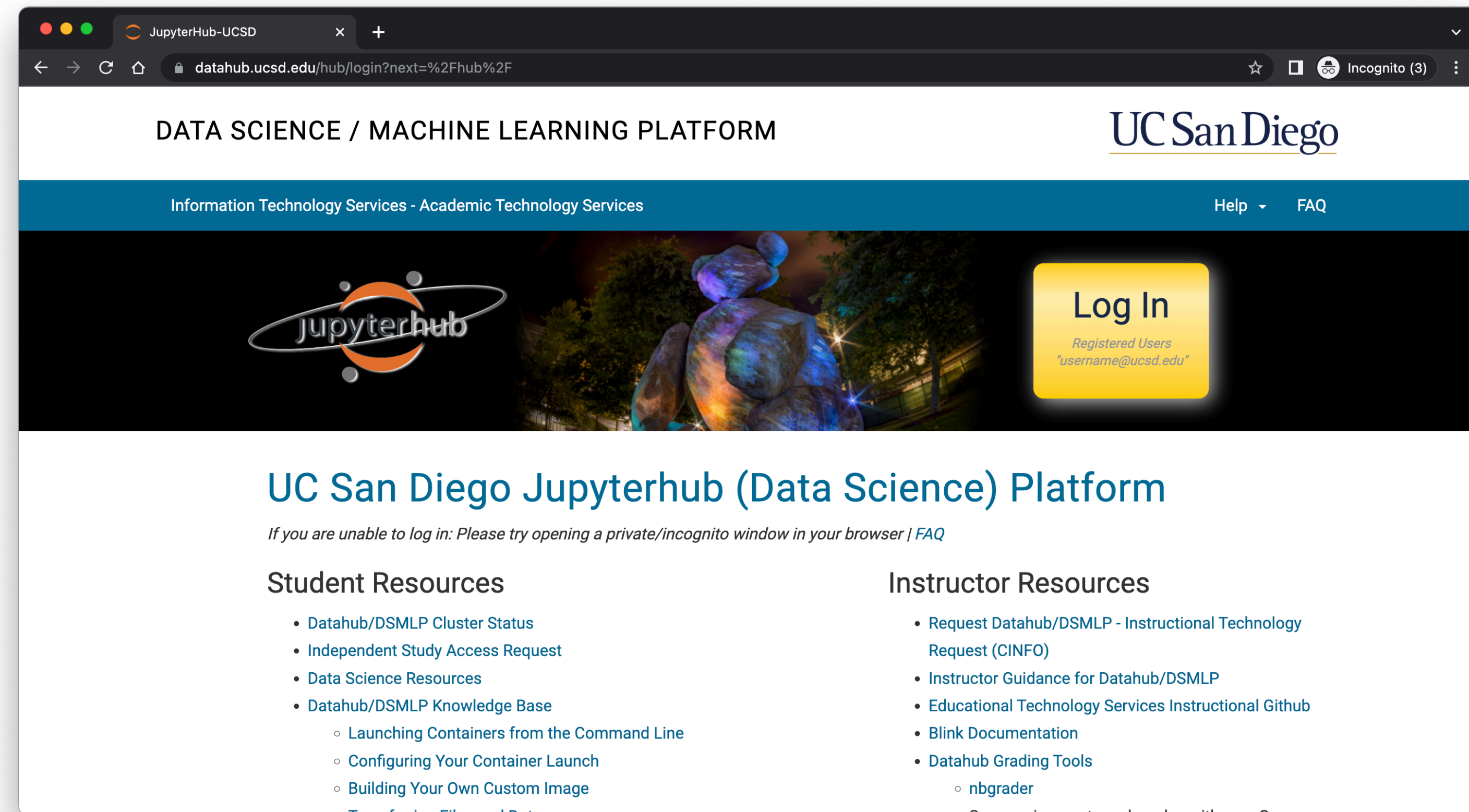
PHYS 142/242

Lab 01: DataHub, Jupyter, Python

Javier Duarte — January 9, 2023

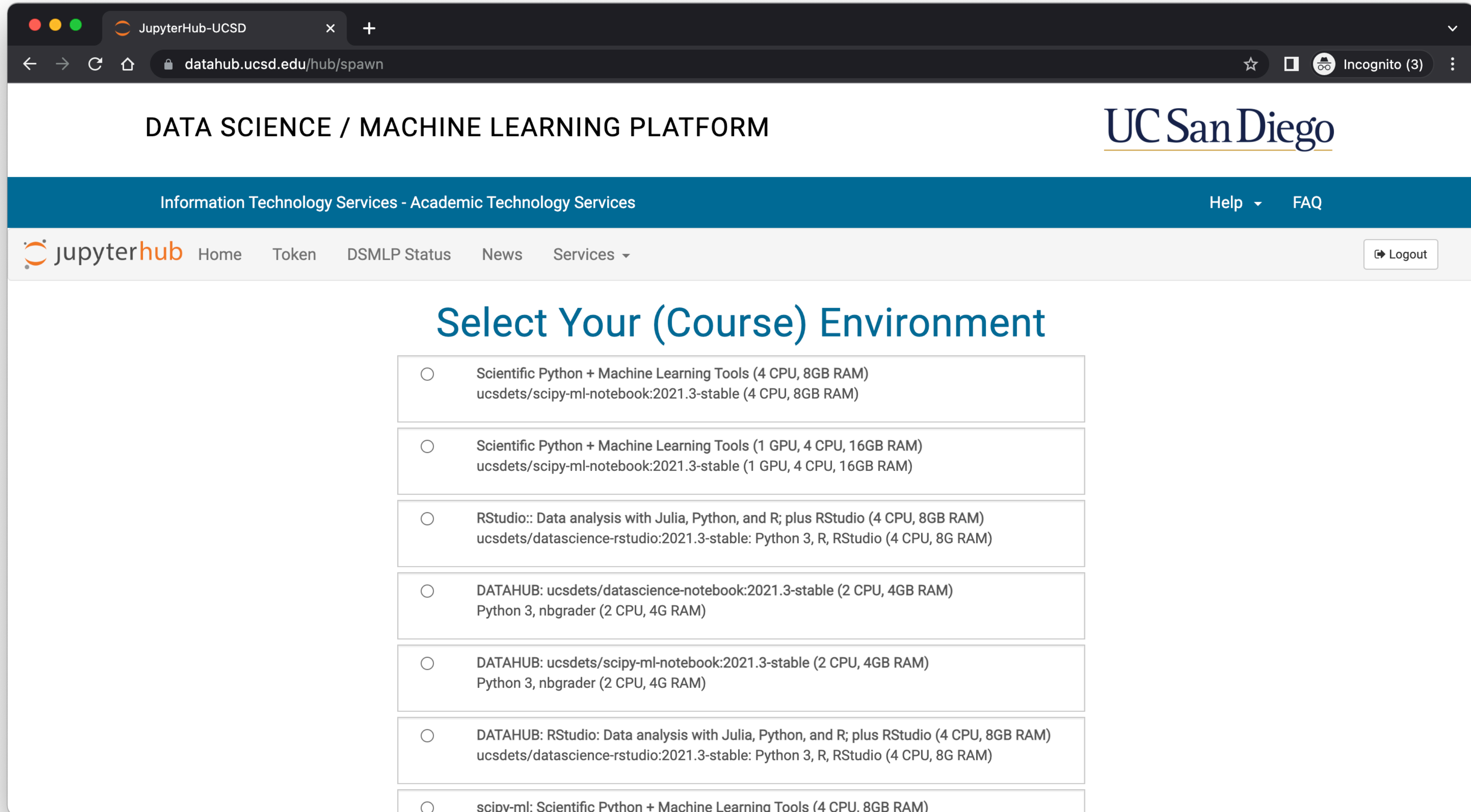
DataHub

- We will use DataHub for in-class hands-on portions
 - Recommend to use it for homework, final project, etc.
- Address: datahub.ucsd.edu
- Similar to public, free services Google Colab, but with access to better CPUs and GPUs and run by UCSD
- Provides a “Jupyter notebook” interface (Python-based but interactive coding like MATLAB/Mathematica)



Logging in

- After logging in, choose a course environment...
PHYS 141 PHYS 241 - Comp Phys I: Prob Models & Simu - Duarte [SP23]
jmduarte/phys141:latest (2 CPU, 4G RAM, 1 GPU)



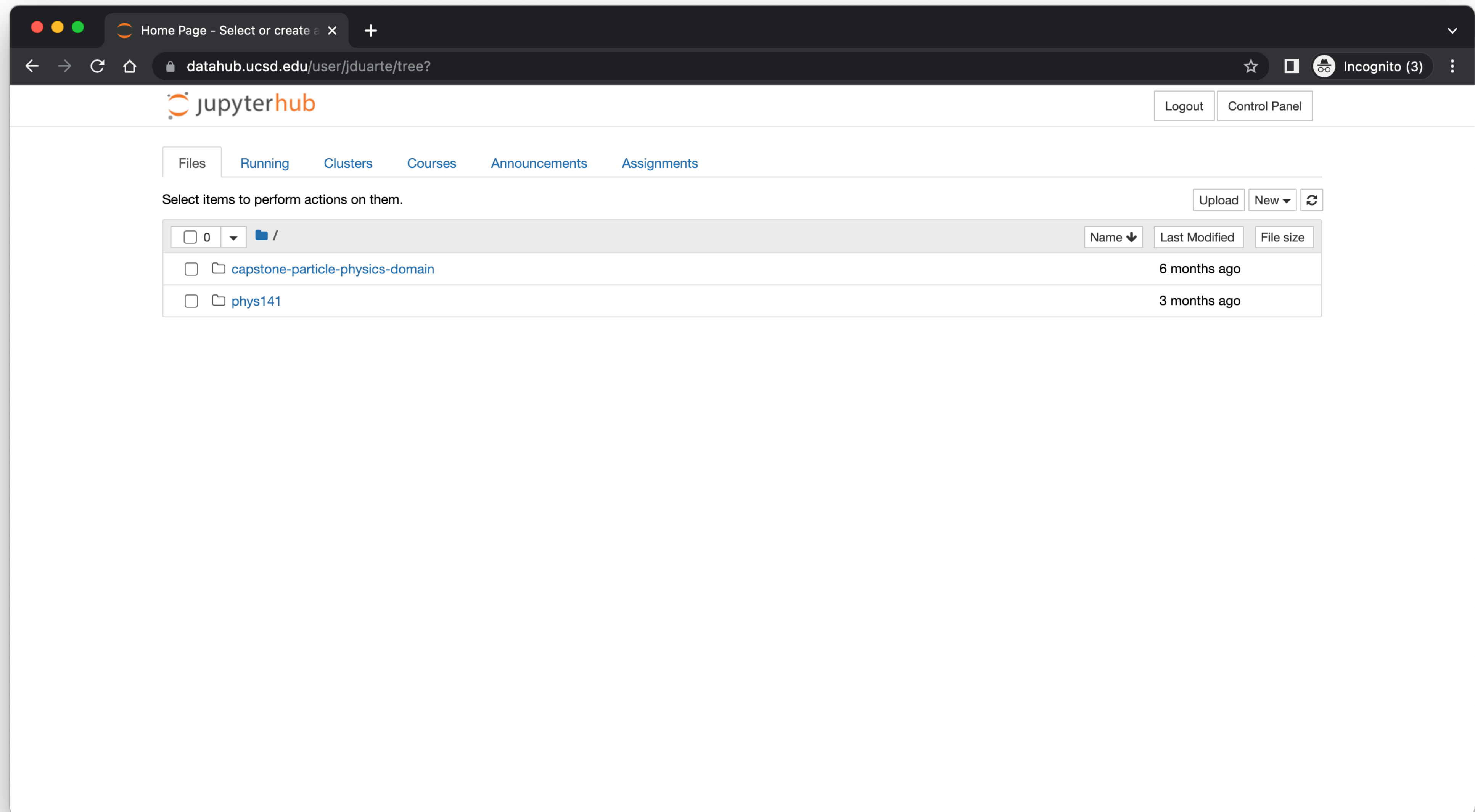
The screenshot shows a web browser window with the URL `datahub.ucsd.edu/hub/spawn`. The page header includes "DATA SCIENCE / MACHINE LEARNING PLATFORM" and the "UC San Diego" logo. A navigation bar contains "Information Technology Services - Academic Technology Services", "Help", and "FAQ". Below this is a JupyterHub navigation bar with links for "Home", "Token", "DSMLP Status", "News", and "Services", along with a "Logout" button.

Select Your (Course) Environment

☐	Scientific Python + Machine Learning Tools (4 CPU, 8GB RAM) ucsdets/scipy-ml-notebook:2021.3-stable (4 CPU, 8GB RAM)
☐	Scientific Python + Machine Learning Tools (1 GPU, 4 CPU, 16GB RAM) ucsdets/scipy-ml-notebook:2021.3-stable (1 GPU, 4 CPU, 16GB RAM)
☐	RStudio:: Data analysis with Julia, Python, and R; plus RStudio (4 CPU, 8GB RAM) ucsdets/datascience-rstudio:2021.3-stable: Python 3, R, RStudio (4 CPU, 8G RAM)
☐	DATAHUB: ucsdets/datascience-notebook:2021.3-stable (2 CPU, 4GB RAM) Python 3, nbgrader (2 CPU, 4G RAM)
☐	DATAHUB: ucsdets/scipy-ml-notebook:2021.3-stable (2 CPU, 4GB RAM) Python 3, nbgrader (2 CPU, 4G RAM)
☐	DATAHUB: RStudio: Data analysis with Julia, Python, and R; plus RStudio (4 CPU, 8GB RAM) ucsdets/datascience-rstudio:2021.3-stable: Python 3, R, RStudio (4 CPU, 8G RAM)
☐	scipy-ml: Scientific Python + Machine Learning Tools (4 CPU, 8GB RAM)

Jupyter interface

- Spawns a “JupyterHub” (let’s go step by step through all the buttons)



Start coding!

- Coding in Jupyter notebooks

Home Page - Select or create a... phys241_test - Jupyter Notebo... 1.6 Appendix A. Complex numb... python - Does Numpy automati... cupy version - Google Search

datahub.ucsd.edu/user/jduarte/notebooks/phys241_test.ipynb

jupyterhub phys241_test Last Checkpoint: a few seconds ago (unsaved changes) Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (clean)

Run Code Show Usage Validate

```
In [1]: print("hello world")
hello world

In [2]: import numpy as np

In [3]: # matrix multiplication
a = np.array([[1, 2, 3],
              [2, 3, 2],
              [1, 1, 2],
              [2, 2, 2]])
a.shape

Out[3]: (4, 3)

In [4]: b = np.array([[1],
                     [2],
                     [1]])
b.shape

Out[4]: (3, 1)

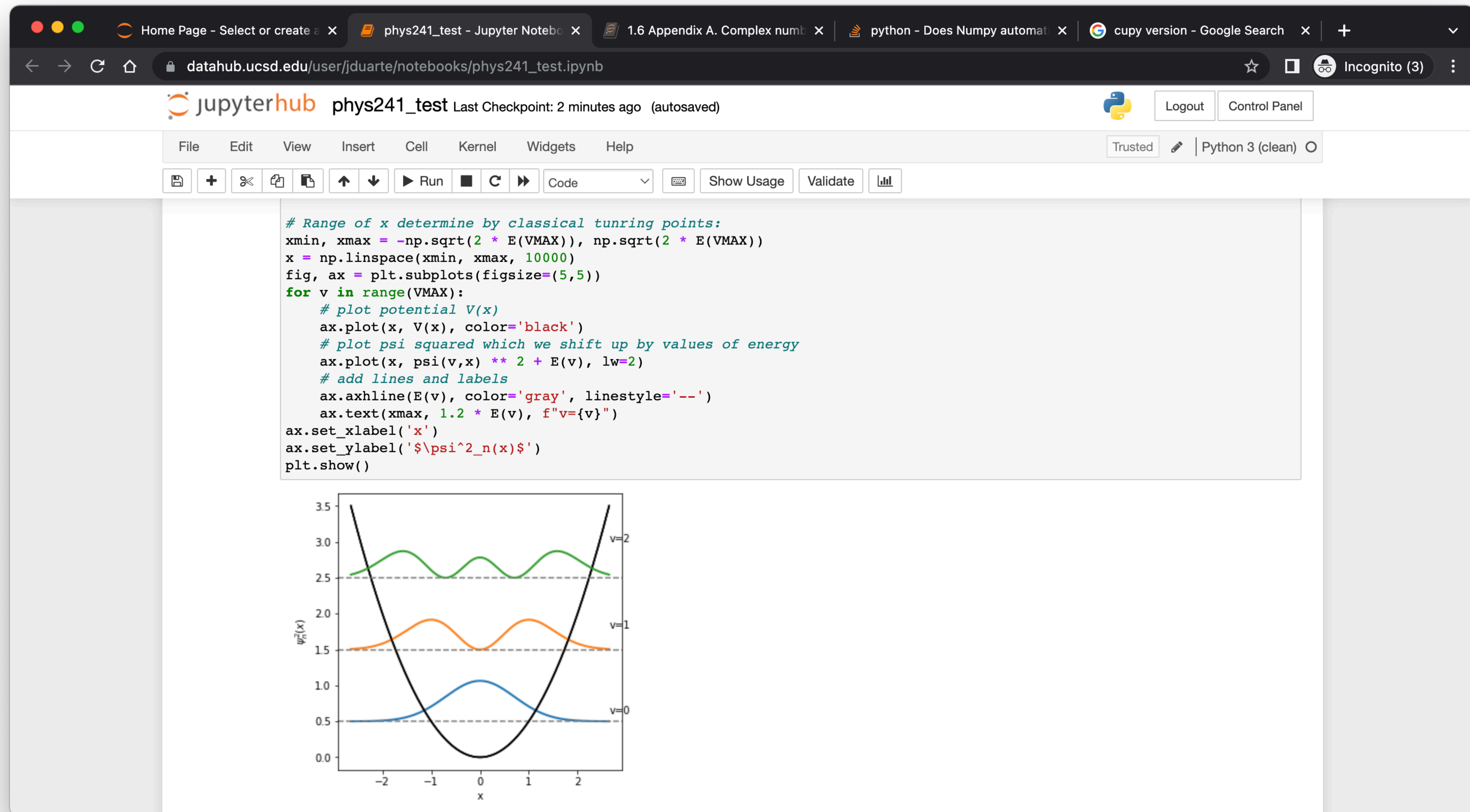
In [5]: a@b

Out[5]: array([[ 8],
               [10],
               [ 5],
               [ 8]])

In [6]: from scipy.special import hermite
from math import factorial
```

Plotting

- Plotting can be done easily with Matplotlib



Installing a missing library

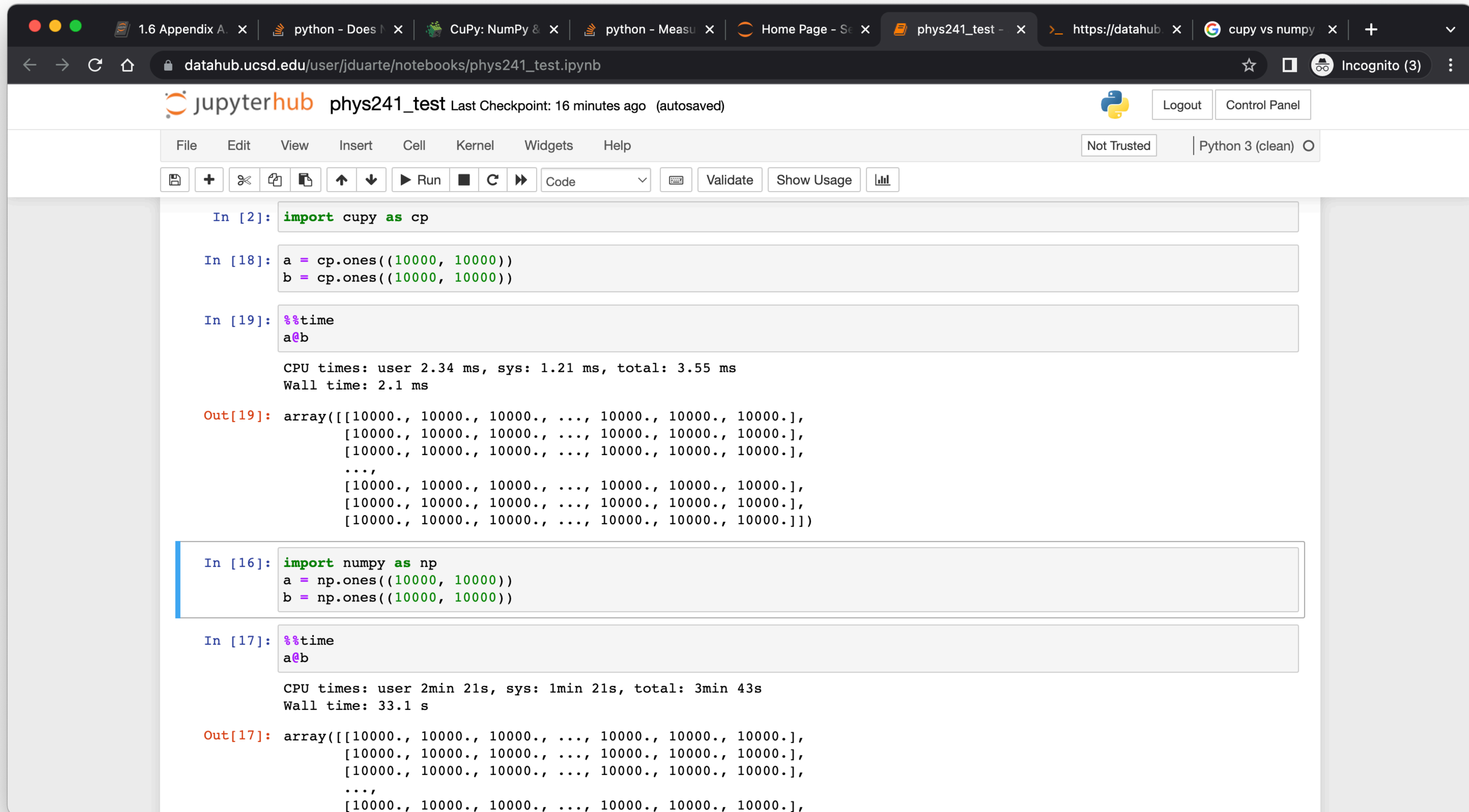
- Not all libraries are preloaded, but it's easy to install a new one
- Note: restart your “kernel” after doing this

```
In [1]: !pip install cupy-cuda112 --user
```

```
Collecting cupy-cuda112
  Using cached cupy_cuda112-10.3.1-cp39-cp39-manylinux1_x86_64.whl (78.9 MB)
Collecting fastrlock>=0.5
  Using cached fastrlock-0.8-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (49 kB)
Requirement already satisfied: numpy<1.25,>=1.18 in /opt/conda/lib/python3.9/site-packages (from cupy-cuda112) (1.19.5)
Installing collected packages: fastrlock, cupy-cuda112
Successfully installed cupy-cuda112-10.3.1 fastrlock-0.8
```


Speed up numerical calculations with the GPU

- Many libraries to do this: TensorFlow, CuPy, PyTorch, ...



The screenshot shows a Jupyter Notebook interface in a web browser. The browser's address bar displays `datahub.ucsd.edu/user/jduarte/notebooks/phys241_test.ipynb`. The notebook's title bar indicates the name `phys241_test` and notes the last checkpoint was 16 minutes ago. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for saving, adding cells, and running code. The notebook content consists of several code cells. The first cell imports `cupy` as `cp`. The second cell creates two 10,000x10,000 matrices of ones using `cp.ones`. The third cell uses `%%time` to time the matrix multiplication `a@b`, which completes in approximately 3.55 ms. The output shows a large array of ones. The fourth cell imports `numpy` as `np` and creates similar matrices. The fifth cell times the `a@b` multiplication, which takes 33.1 seconds. The output is the same large array of ones. This comparison demonstrates the significant performance advantage of CuPy over NumPy for GPU-accelerated numerical calculations.

```
In [2]: import cupy as cp

In [18]: a = cp.ones((10000, 10000))
         b = cp.ones((10000, 10000))

In [19]: %%time
         a@b

CPU times: user 2.34 ms, sys: 1.21 ms, total: 3.55 ms
Wall time: 2.1 ms

Out[19]: array([[10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               ...,
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.]])

In [16]: import numpy as np
         a = np.ones((10000, 10000))
         b = np.ones((10000, 10000))

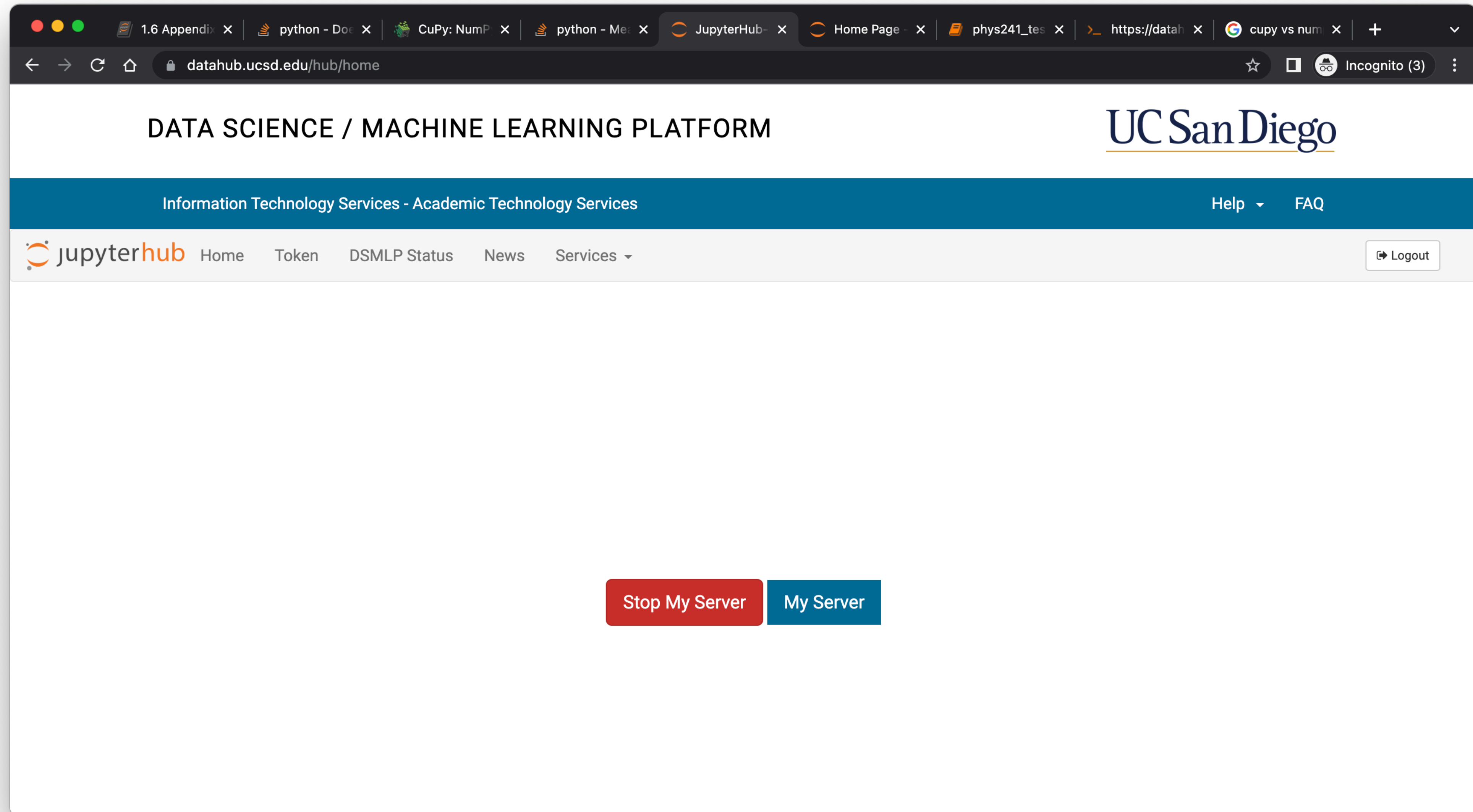
In [17]: %%time
         a@b

CPU times: user 2min 21s, sys: 1min 21s, total: 3min 43s
Wall time: 33.1 s

Out[17]: array([[10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               [10000., 10000., 10000., ..., 10000., 10000., 10000.],
               ...,
               [10000., 10000., 10000., ..., 10000., 10000., 10000.]])
```


Exiting / shutting down server

- When you're done; hit control panel and "stop your server"
When you start it again, all your data will still be there



Data Science & Machine Learning Platform (DSMLP)

- UCSD Data Science & Machine Learning Platform (DSMLP)
 - Based on docker and **Kubernetes** (K8s): open-source system for automating deployment, scaling, and management of containerized applications



- Overview:
 - You log in to a remote "login" node
 - From that login node, you can launch a "pod" running a container
 - The pod automatically starts a "JupyterHub" web server (only accessible from campus so you must VPN if off campus)
 - You are also automatically logged into that pod so you can run interactive terminal commands, etc.

More Resources

- UCSD DSMLP Cluster
 - [Using the DSMLP server](#)
 - [Customizing the DSMLP Containers](#)
- Terminal and Command-Line Interface
 - [Bash Scripting Reference](#)
 - [Git\(Hub\) Resources](#)