Intro to JupyterHub on the National Research Platform

Presented by Cal Poly Humboldt - ITS

Brian Campbell - Academic Technology Specialist

Ravi Chalasani - ITS Research Support Technician

Cal Poly Humboldt.

Library 308 - 6/26/2025

National Research Platform

What is the National Research Platform?

- **Free** to access subject to accepting Acceptable Use Policy
- Collection of computing resources housed under one umbrella
- Funded by NSF
- Grew out of the former Pacific Research Platform
- Distributed network of over 400 servers with massive computing power - https://dash.nrp-nautilus.io/
- More computing power compared to running locally
- Instantly scalable to higher workloads

H.

National Research Platform

- Do Not Mine crypto on NSF funded machines
  - federal crime
- Violators will get caught
- Violators will be reported to the FBI
- Cluster users may not use the computers to enrich themselves financially
  - Faculty/PIs: Please share this with your students

Accessing NRP Nautilus Portal

- URL for NRP Nautilus Portal: https://nrp.ai/viz/resources/
- We use the portal to create Namespaces for your projects
- We can add you and your students to your Namespace
- You can check for available resources
- NRP access is provided via Cyber Infrastructure CILogon and Authentik

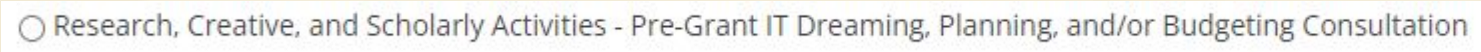
CILogon

NRP Support on Campus

- Our process:
  1. Initial consultation about your project
     - Team based help with accessing NRP
     - Uploading any preexisting code/Notebooks to JupyterHub
  2. Troubleshooting, custom packages for specific needs
     - e.g. Otter-grader
  3. Uploading/downloading data to/from NRP
  4. Collaborate to adapt and improve processes
  5. Continued consultation until your project is successful

› How to Contact Us

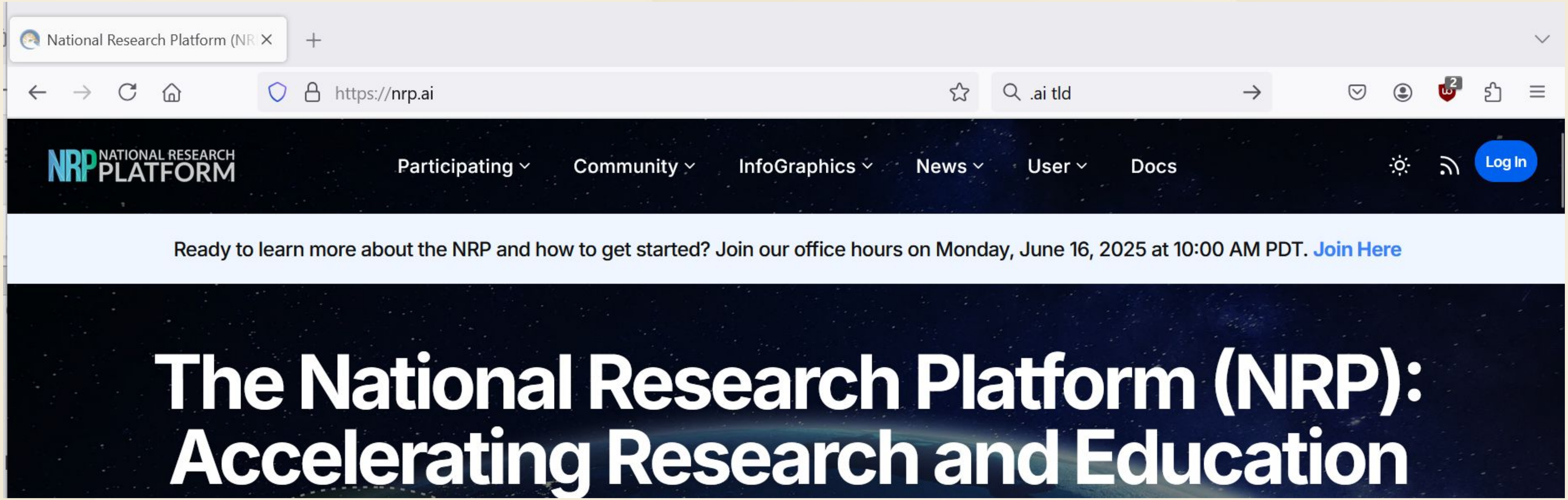
- SBS 413 (by appointment)
- Email: itsresearchsupport@humboldt.edu
- Create a HelpDesk ticket



○ Research, Creative, and Scholarly Activities - Pre-Grant IT Dreaming, Planning, and/or Budgeting Consultation

- Website: https://its.humboldt.edu/research/high-performance-computing
- Phone: x4100

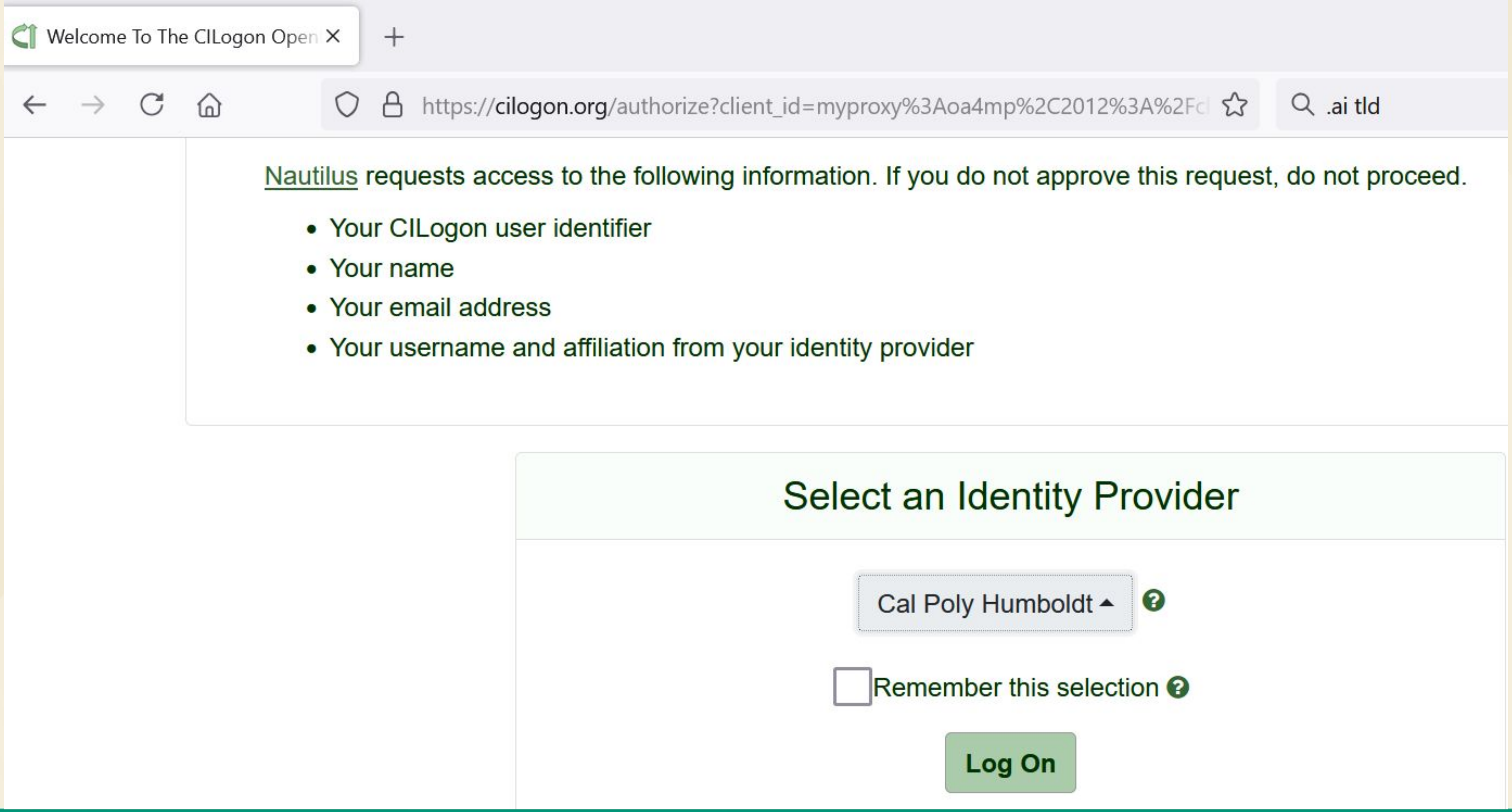
NRP Portal Pre-Login Screen



Ready to learn more about the NRP and how to get started? Join our office hours on Monday, June 16, 2025 at 10:00 AM PDT. Join Here

The National Research Platform (NRP): Accelerating Research and Education

NRP Portal CILogon Screen



Welcome To The CILogon Open
https://cilogon.org/authorize?client_id=myproxy%3Aoa4mp%2C2012%3A%2Fc
.ai tld

Nautilus requests access to the following information. If you do not approve this request, do not proceed.

- Your CILogon user identifier
- Your name
- Your email address
- Your username and affiliation from your identity provider

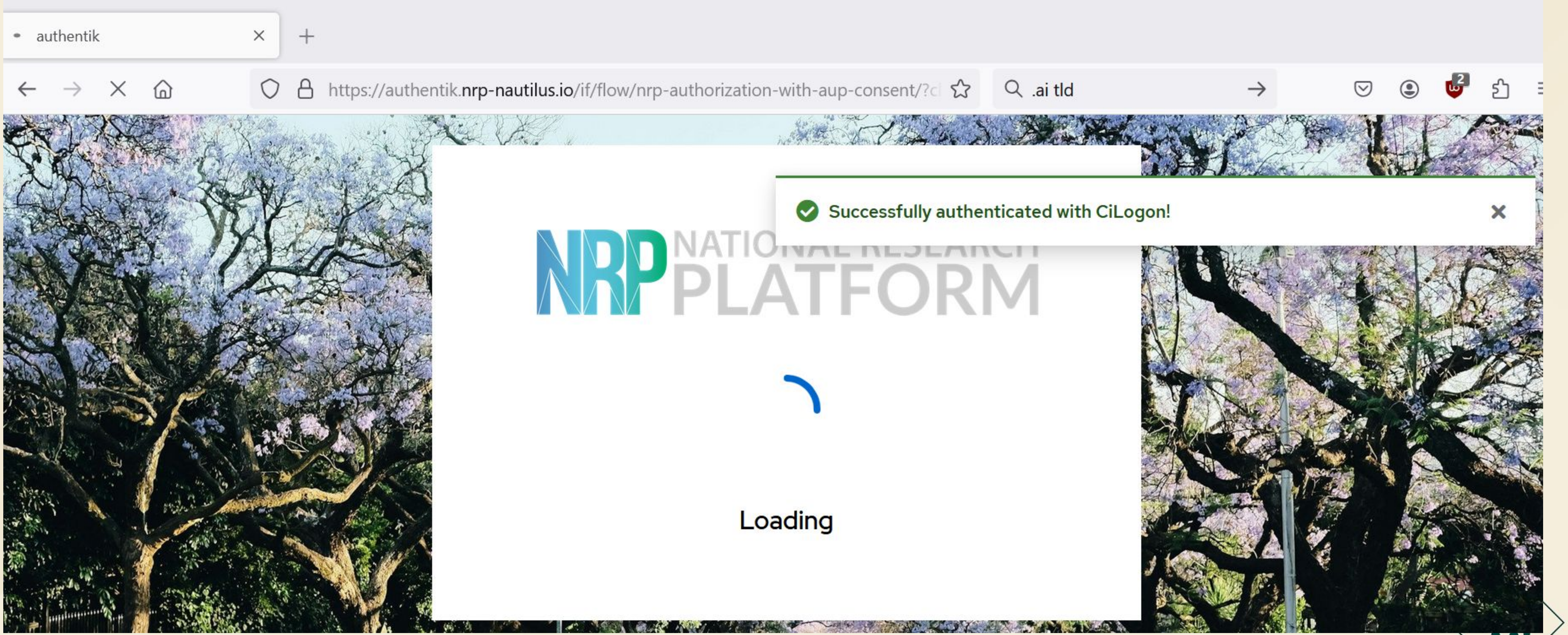
Select an Identity Provider

Cal Poly Humboldt

□ Remember this selection

Log On

NRP Portal Authentik Screen

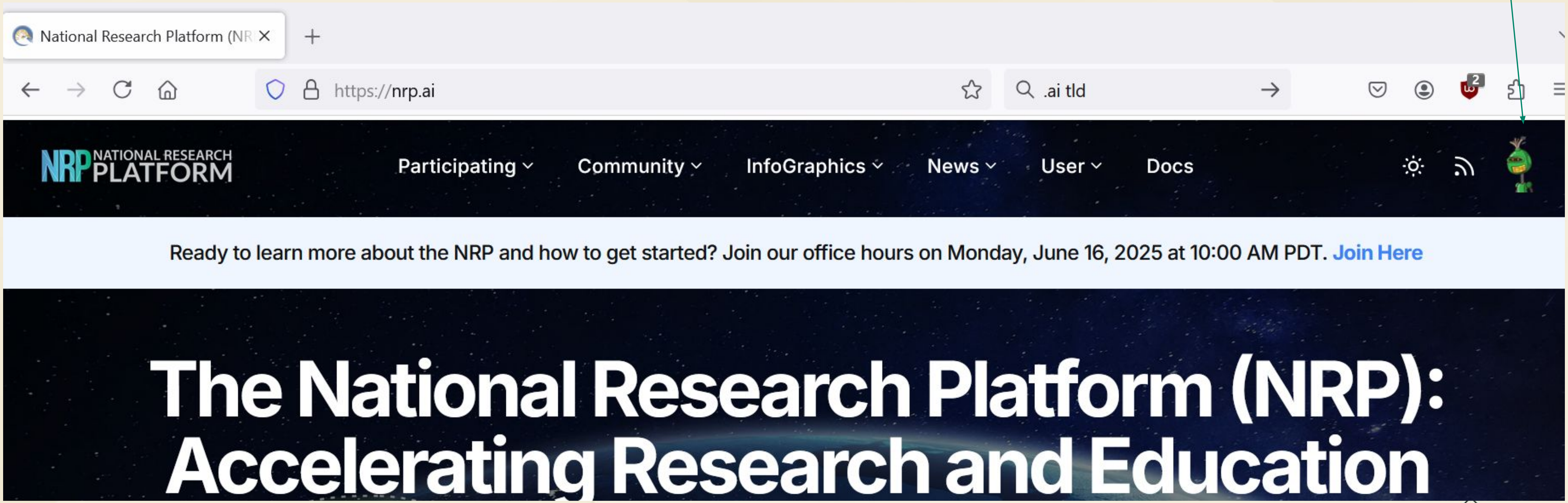


authentik
https://authentik.nrp-nautilus.io/if/flow/nrp-authorization-with-aup-consent/?c
.ai tld
NRP NATIONAL RESEARCH PLATFORM

Successfully authenticated with CiLogon!

Loading

NRP Portal Logged-in Screen



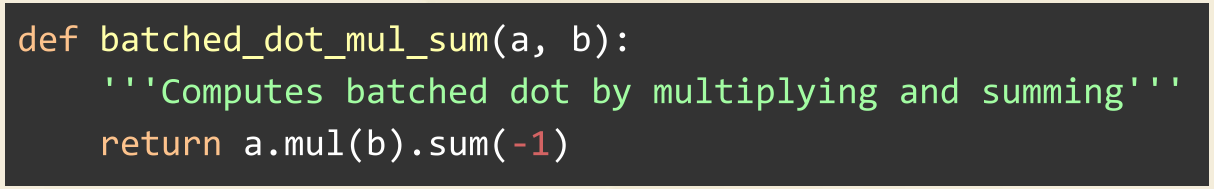
Ready to learn more about the NRP and how to get started? Join our office hours on Monday, June 16, 2025 at 10:00 AM PDT. Join Here

The National Research Platform (NRP): Accelerating Research and Education

Jupyter… Hub, Notebook, etc

- JupyterLab: Local development environment for Jupyter Notebooks, allowing for markdown and code execution in “cells”
- It is not a programming language, it supports other languages
- JupyterHub (JH): Allows for execution of Jupyter Notebooks on a remote server
- Instantly transferable from local computer to high-performance computing resources
- Your data is stored in a 5GB Persistent Volume Claim (PVC)
- Code will keep running as long as your browser tab is open to JH
- Idle JH sessions will be terminated by NRP however your PVC data is preserved

Local vs. NRP (with GPUs)



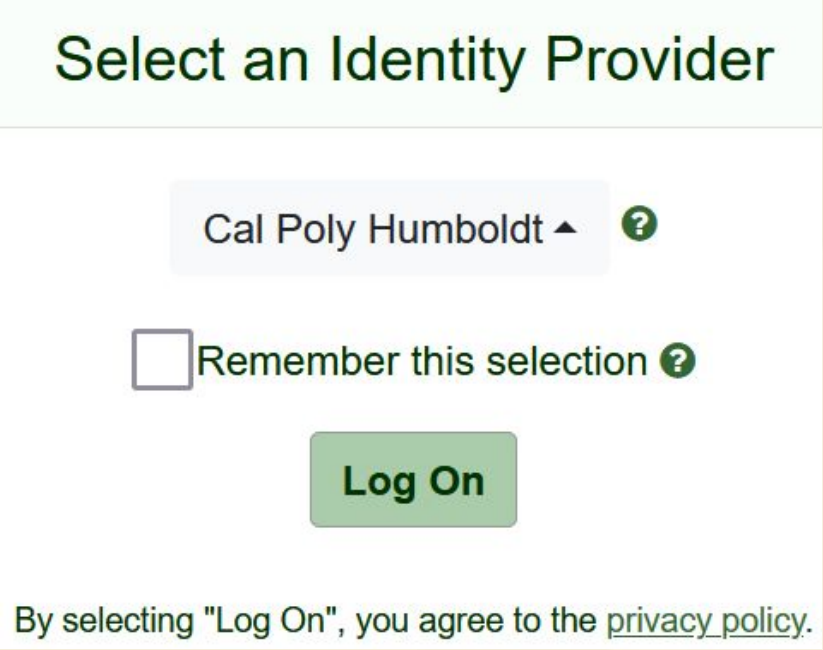
```python
def batched_dot_mul_sum(a, b):
    '''Computes batched dot by multiplying and summing'''
    return a.mul(b).sum(-1)
```

2.879 s

0.000259 s

How Do I Access JupyterHub?

- https://jupyterhub-west.nrp-nautilus.io/
- https://github.com/cal-poly-humboldt/NRP-Tutorials
- Use CyberInfrastructure Logon (CILogon)
  - Use Humboldt login



Select an Identity Provider

Cal Poly Humboldt ▴

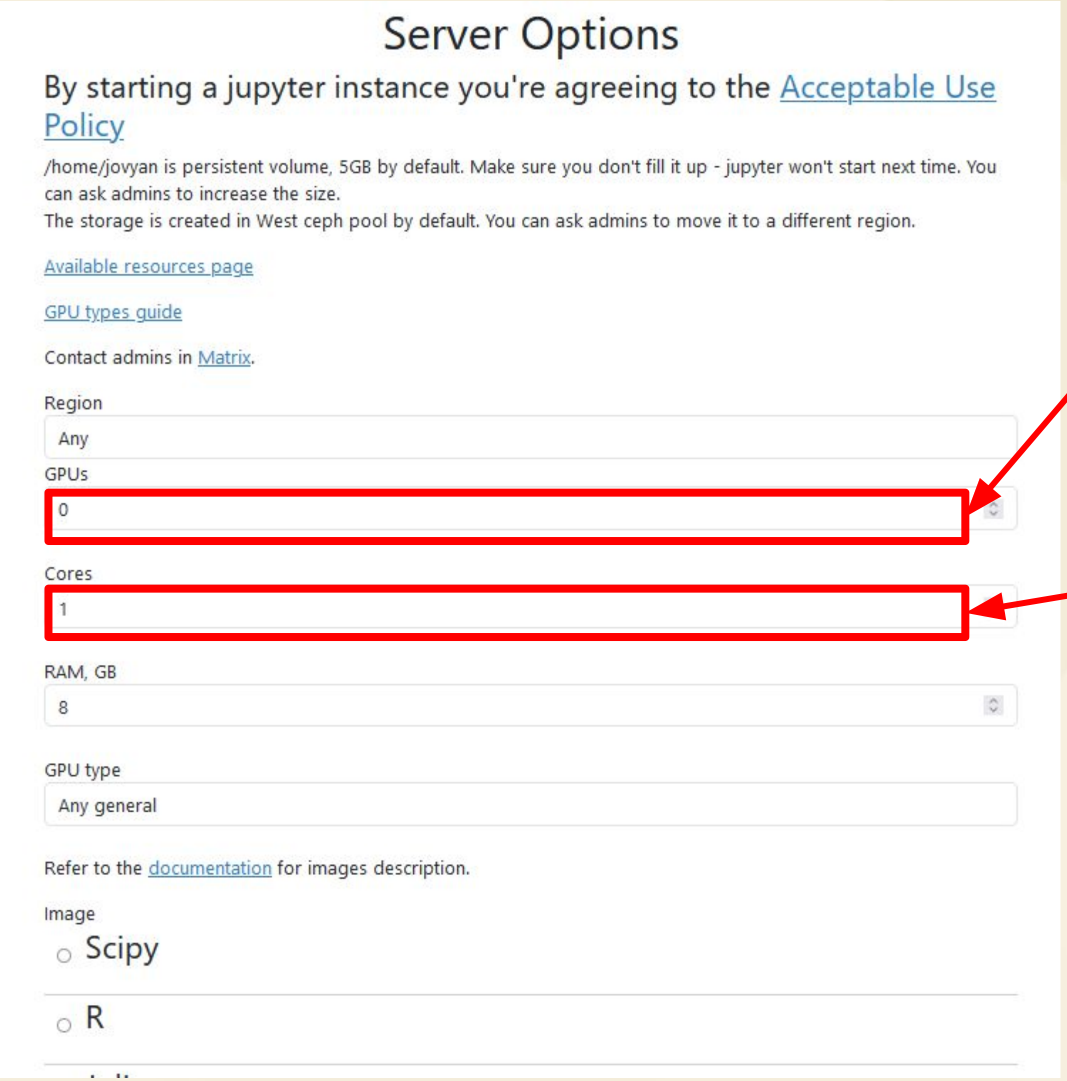
□ Remember this selection

Log On

By selecting "Log On", you agree to the privacy policy.

CILogon

JupyterHub Options



Server Options

By starting a jupyter instance you're agreeing to the Acceptable Use Policy

/home/jovyan is persistent volume, 5GB by default. Make sure you don't fill it up - jupyter won't start next time. You can ask admins to increase the size.
The storage is created in West ceph pool by default. You can ask admins to move it to a different region.

Available resources page

GPU types guide

Contact admins in Matrix.

Region: Any

GPUs: 0

Cores: 1

RAM, GB: 8

GPU type: Any general

Refer to the documentation for images description.

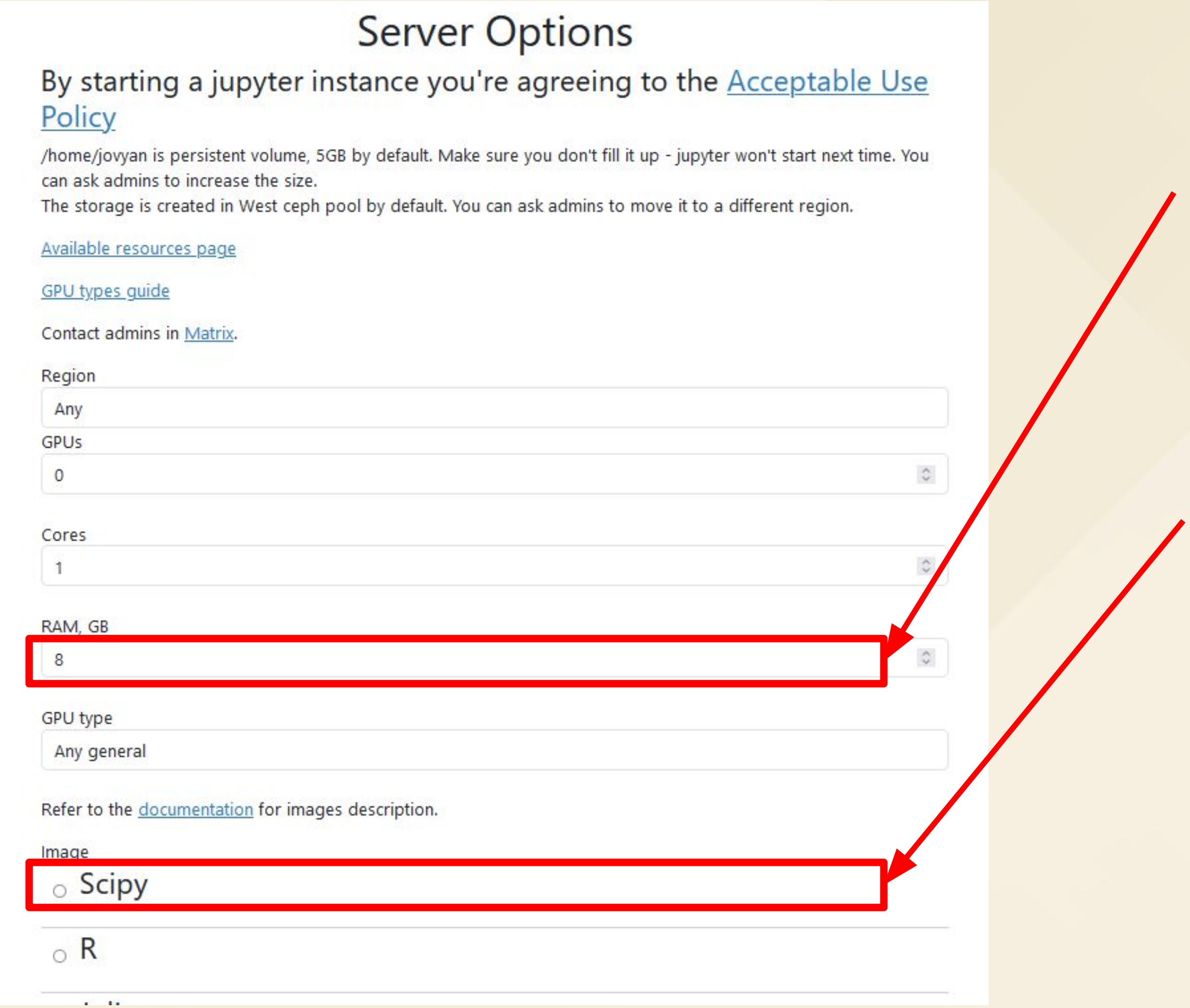
Image

○ Scipy

○ R

- Select Region if desired
- GPU if needed
  - Graphics Processing Unit
  - additional brain for graphics
  - can be adapted for other compute
- Select CPUs called cores
  - Central Processing Unit
  - It’s the brain of the computer
  - That’s where the math is done

› JupyterHub Options



Server Options

By starting a jupyter instance you're agreeing to the Acceptable Use Policy

/home/jovyan is persistent volume, 5GB by default. Make sure you don't fill it up - jupyter won't start next time. You can ask admins to increase the size.
The storage is created in West ceph pool by default. You can ask admins to move it to a different region.

Available resources page

GPU types guide

Contact admins in Matrix.

Region: Any

GPUs: 0

Cores: 1

RAM, GB: 8

GPU type: Any general

Refer to the documentation for images description.

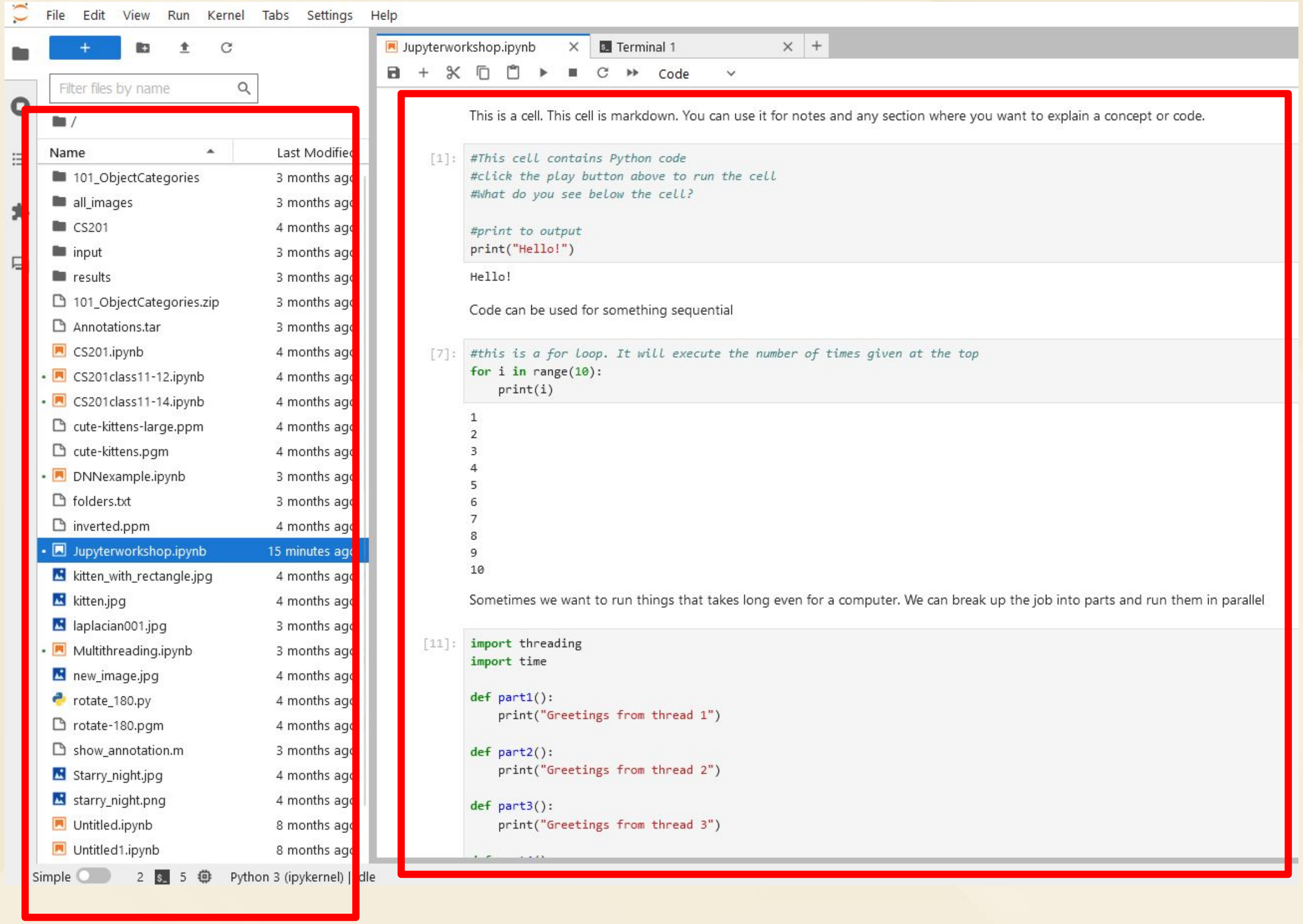
Image

○ Scipy

○ R

- Select RAM
  - Random Access Memory
  - RAM larger than file size if possible
- Select Image (environment)
  - If don’t know use default
  - Unless you are programming in R, then select R

❯ JupyterHub Interface



Name	Last Modified
101_ObjectCategories	3 months ago
all_images	3 months ago
CS201	4 months ago
input	3 months ago
results	3 months ago
101_ObjectCategories.zip	3 months ago
Annotations.tar	3 months ago
CS201.ipynb	4 months ago
CS201class11-12.ipynb	4 months ago
CS201class11-14.ipynb	4 months ago
cute-kittens-large.ppm	4 months ago
cute-kittens.pgm	4 months ago
DNNexample.ipynb	3 months ago
folders.txt	3 months ago
inverted.ppm	4 months ago
Jupyterworkshop.ipynb	15 minutes ago
kitten_with_rectangle.jpg	4 months ago
kitten.jpg	4 months ago
laplacian001.jpg	3 months ago
Multithreading.ipynb	3 months ago
new_image.jpg	4 months ago
rotate_180.py	4 months ago
rotate-180.pgm	4 months ago
show_annotation.m	3 months ago
Starry_night.jpg	4 months ago
starry_night.png	4 months ago
Untitled.ipynb	8 months ago
Untitled1.ipynb	8 months ago

This is a cell. This cell is markdown. You can use it for notes and any section where you want to explain a concept or code.

```
[1]: #This cell contains Python code
     #click the play button above to run the cell
     #What do you see below the cell?

     #print to output
     print("Hello!")
```

```
Hello!
```

Code can be used for something sequential

```
[7]: #this is a for loop. It will execute the number of times given at the top
     for i in range(10):
         print(i)
```

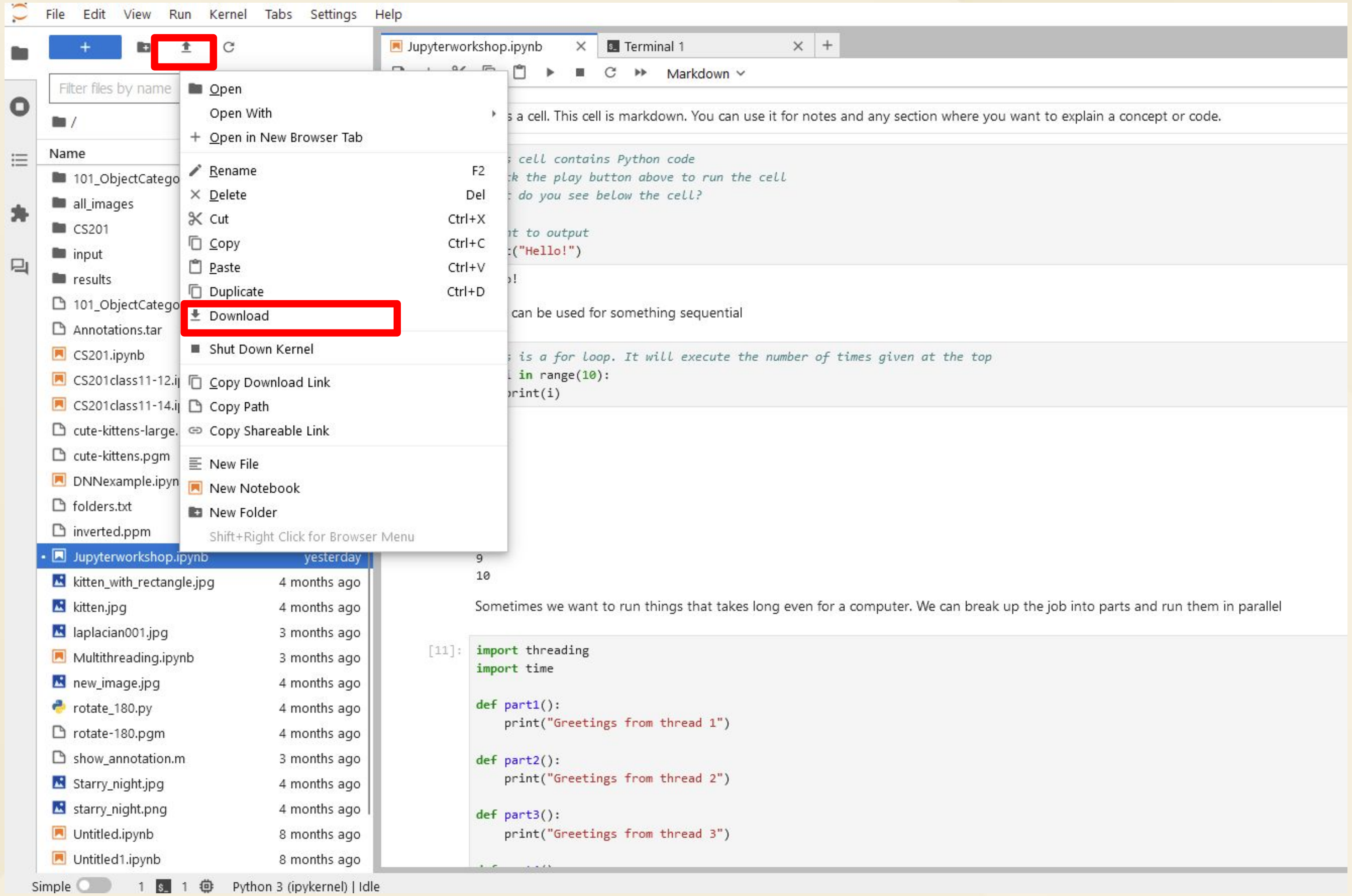
```
1
2
3
4
5
6
7
8
9
10
```

Sometimes we want to run things that takes long even for a computer. We can break up the job into parts and run them in parallel

```
[11]: import threading
      import time

      def part1():
          print("Greetings from thread 1")

      def part2():
          print("Greetings from thread 2")

      def part3():
          print("Greetings from thread 3")
```

- Graphical User Interface (GUI):
  - File system explorer on the left
  - Notebooks etc. on the right

❯ JupyterHub Interface



s a cell. This cell is markdown. You can use it for notes and any section where you want to explain a concept or code.

```
s cell contains Python code
ck the play button above to run the cell
: do you see below the cell?

nt to output
:("Hello!")
```

```
o!
```

can be used for something sequential

```
s is a for loop. It will execute the number of times given at the top
i in range(10):
print(i)
```

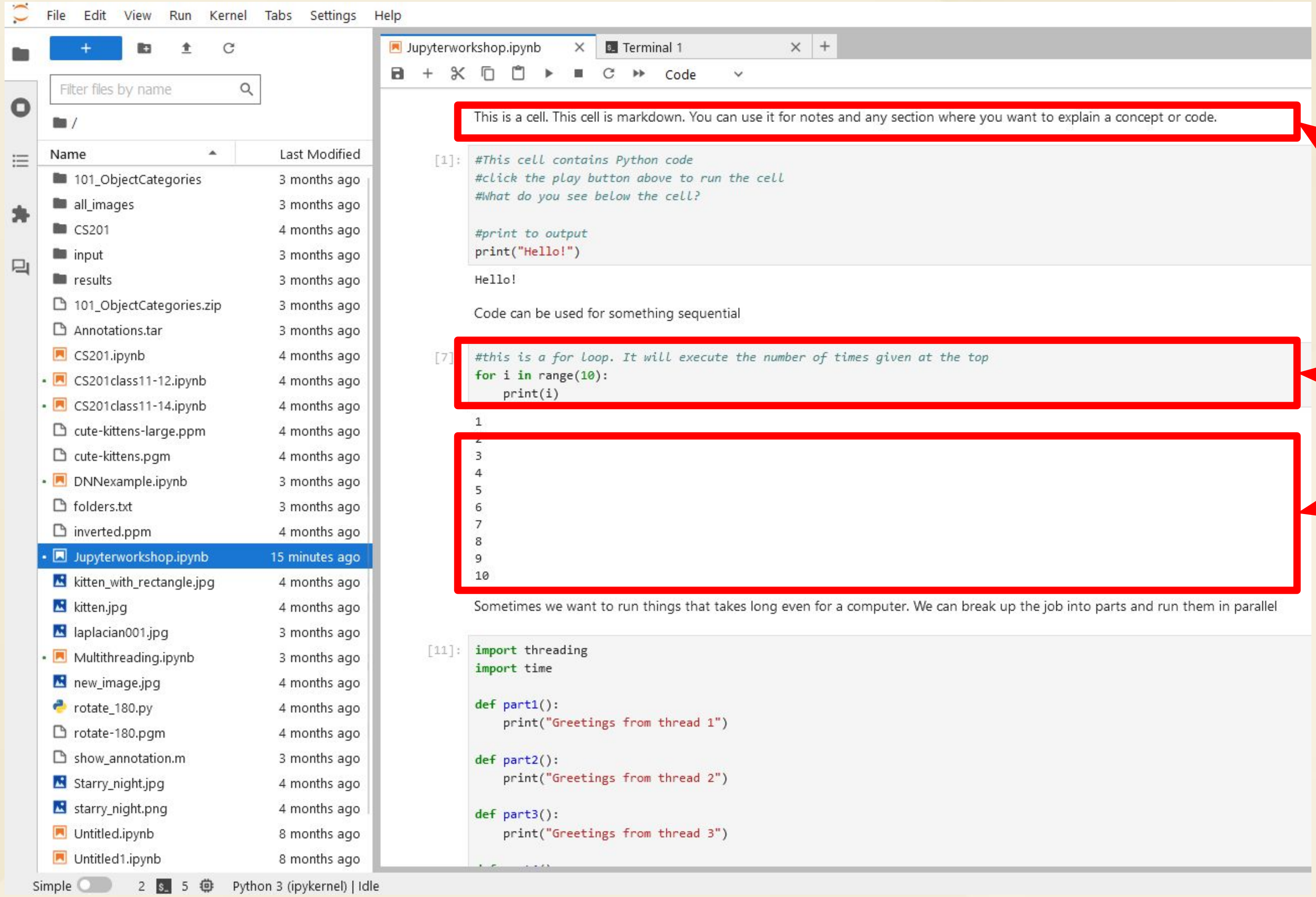
```
9
10
```

Sometimes we want to run things that takes long even for a computer. We can break up the job into parts and run them in parallel

```
[11]: import threading
      import time

      def part1():
          print("Greetings from thread 1")

      def part2():
          print("Greetings from thread 2")

      def part3():
          print("Greetings from thread 3")
```

- File System:
  - upload files
  - download files
    - R-click the file
    - select Download

❯ JupyterHub Interface



File Edit View Run Kernel Tabs Settings Help
+
Filter files by name
/
Jupyterworkshop.ipynb
Terminal 1
Code
Simple
2
5
Python 3 (ipykernel) | Idle

Name	Last Modified
101_ObjectCategories	3 months ago
all_images	3 months ago
CS201	4 months ago
input	3 months ago
results	3 months ago
101_ObjectCategories.zip	3 months ago
Annotations.tar	3 months ago
CS201.ipynb	4 months ago
CS201class11-12.ipynb	4 months ago
CS201class11-14.ipynb	4 months ago
cute-kittens-large.ppm	4 months ago
cute-kittens.pgm	4 months ago
DNNexample.ipynb	3 months ago
folders.txt	3 months ago
inverted.ppm	4 months ago
Jupyterworkshop.ipynb	15 minutes ago
kitten_with_rectangle.jpg	4 months ago
kitten.jpg	4 months ago
laplacian001.jpg	3 months ago
Multithreading.ipynb	3 months ago
new_image.jpg	4 months ago
rotate_180.py	4 months ago
rotate-180.pgm	4 months ago
show_annotation.m	3 months ago
Starry_night.jpg	4 months ago
starry_night.png	4 months ago
Untitled.ipynb	8 months ago
Untitled1.ipynb	8 months ago

This is a cell. This cell is markdown. You can use it for notes and any section where you want to explain a concept or code.

```
[1]: #This cell contains Python code
     #click the play button above to run the cell
     #What do you see below the cell?

     #print to output
     print("Hello!")
```

```
Hello!
```

Code can be used for something sequential

```
[7]: #this is a for loop. It will execute the number of times given at the top
     for i in range(10):
         print(i)
```

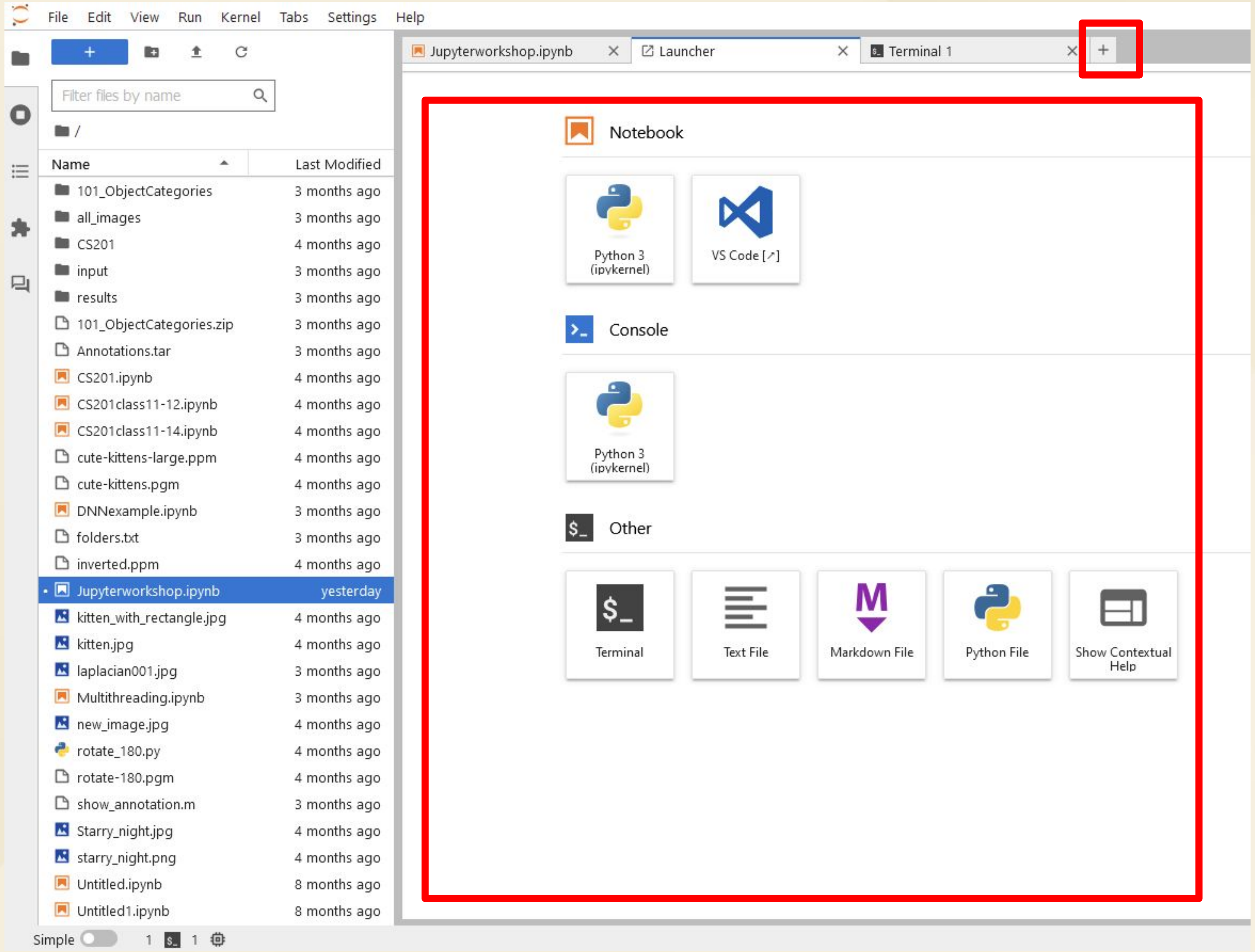
```
1
2
3
4
5
6
7
8
9
10
```

Sometimes we want to run things that takes long even for a computer. We can break up the job into parts and run them in parallel

```
[11]: import threading
      import time

      def part1():
          print("Greetings from thread 1")

      def part2():
          print("Greetings from thread 2")

      def part3():
          print("Greetings from thread 3")
```

- Notebook:
  - ○ Cells
    - ■ Markdown for notes
    - ■ Code for Python or R
    - ■ Output areas after the code runs
  - ○ Pick which cells to run

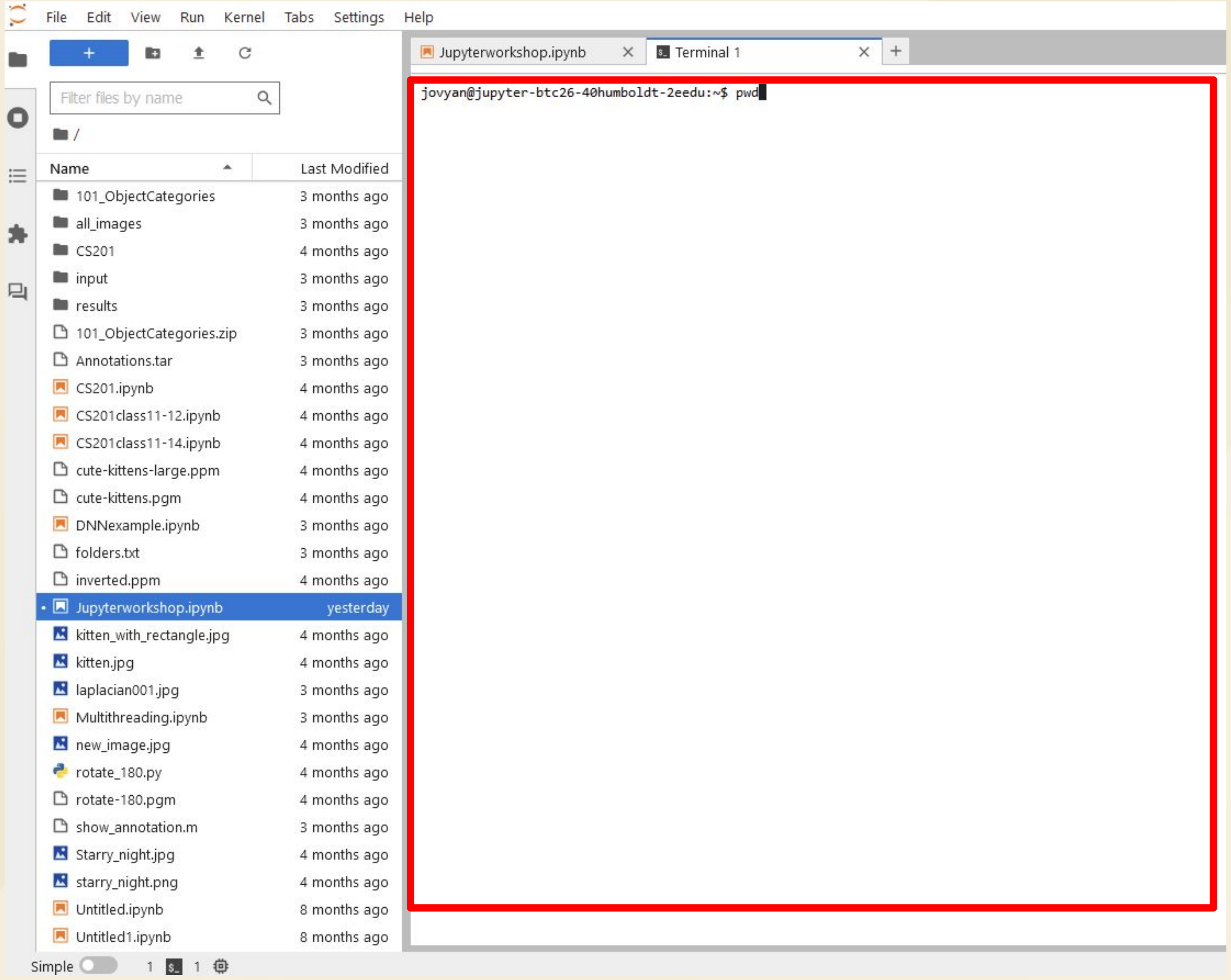
❯ JupyterHub Interface



Name	Last Modified
101_ObjectCategories	3 months ago
all_images	3 months ago
CS201	4 months ago
input	3 months ago
results	3 months ago
101_ObjectCategories.zip	3 months ago
Annotations.tar	3 months ago
CS201.ipynb	4 months ago
CS201class11-12.ipynb	4 months ago
CS201class11-14.ipynb	4 months ago
cute-kittens-large.ppm	4 months ago
cute-kittens.pgm	4 months ago
DNNexample.ipynb	3 months ago
folders.txt	3 months ago
inverted.ppm	4 months ago
Jupyterworkshop.ipynb	yesterday
kitten_with_rectangle.jpg	4 months ago
kitten.jpg	4 months ago
laplacian001.jpg	3 months ago
Multithreading.ipynb	3 months ago
new_image.jpg	4 months ago
rotate_180.py	4 months ago
rotate-180.pgm	4 months ago
show_annotation.m	3 months ago
Starry_night.jpg	4 months ago
starry_night.png	4 months ago
Untitled.ipynb	8 months ago
Untitled1.ipynb	8 months ago

- Click the + to use the Launcher
  - Open a New Tab
    - New Notebook File
    - Terminal Windows

› JupyterHub Interface



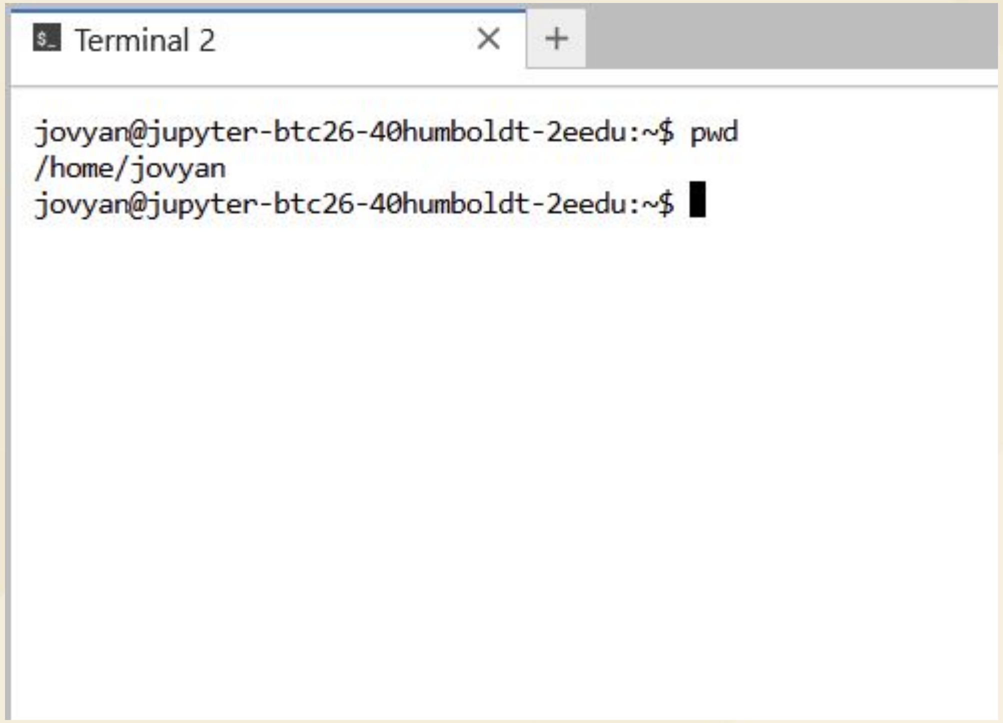
Name	Last Modified
101_ObjectCategories	3 months ago
all_images	3 months ago
CS201	4 months ago
input	3 months ago
results	3 months ago
101_ObjectCategories.zip	3 months ago
Annotations.tar	3 months ago
CS201.ipynb	4 months ago
CS201class11-12.ipynb	4 months ago
CS201class11-14.ipynb	4 months ago
cute-kittens-large.ppm	4 months ago
cute-kittens.pgm	4 months ago
DNNexample.ipynb	3 months ago
folders.txt	3 months ago
inverted.ppm	4 months ago
Jupyterworkshop.ipynb	yesterday
kitten_with_rectangle.jpg	4 months ago
kitten.jpg	4 months ago
laplacian001.jpg	3 months ago
Multithreading.ipynb	3 months ago
new_image.jpg	4 months ago
rotate_180.py	4 months ago
rotate-180.pgm	4 months ago
show_annotation.m	3 months ago
Starry_night.jpg	4 months ago
starry_night.png	4 months ago
Untitled.ipynb	8 months ago
Untitled1.ipynb	8 months ago

```
jovyan@jupyter-btc26-40humboldt-2eedu:~$ pwd
```

- Terminal:
  - Linux Command Line Interface
    - Usually optional
    - can use to install additional packages
  - Allows for installations
  - Command to try:
    - pwd

Interactive demo of JH

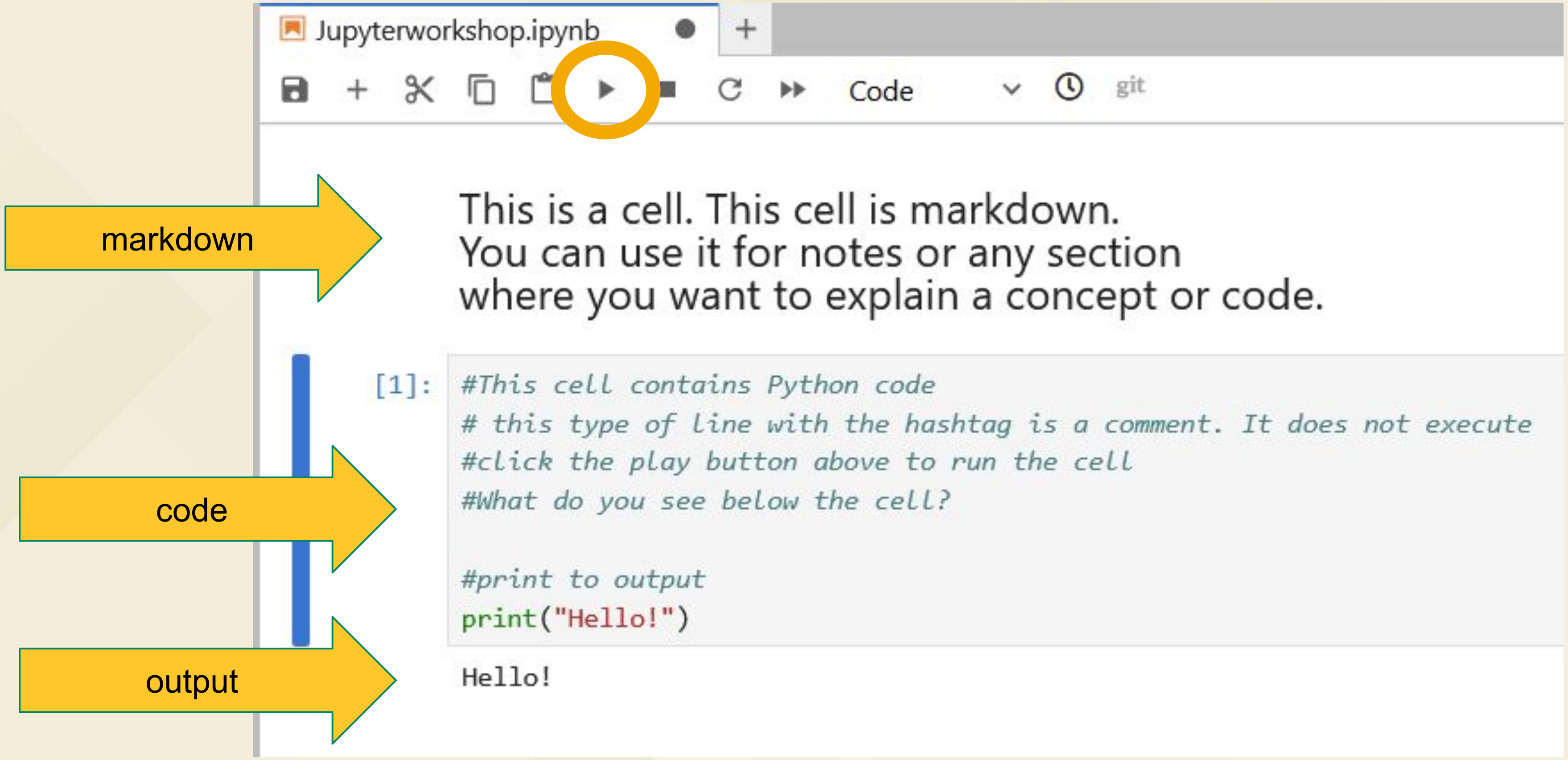
JupyterHub Environment: Linux Terminal



```
jovyan@jupyter-btc26-40humboldt-2eedu:~$ pwd
/home/jovyan
jovyan@jupyter-btc26-40humboldt-2eedu:~$ 
```

- Full Linux terminal
  - run commands
    - ex: pwd

JupyterHub Environment: Notebook

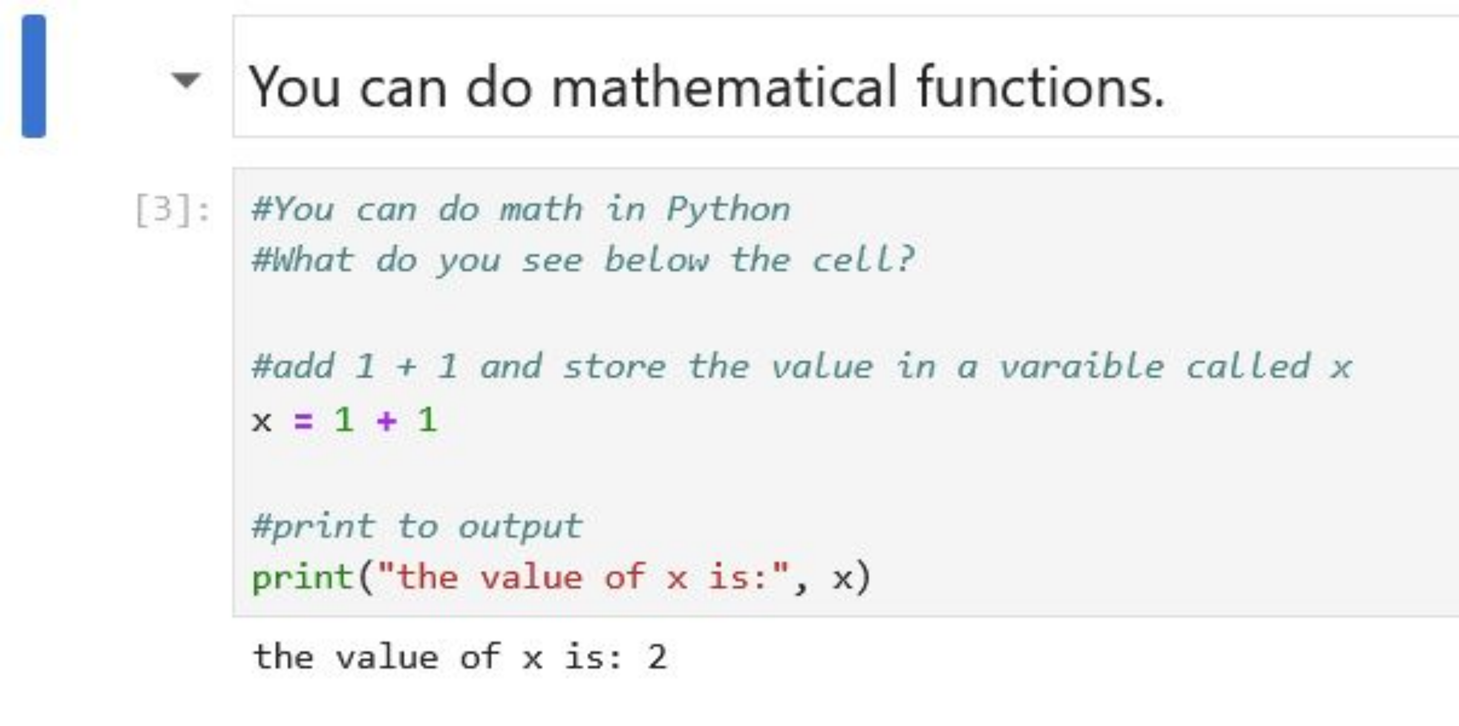


This is a cell. This cell is markdown.
You can use it for notes or any section
where you want to explain a concept or code.

```
[1]: #This cell contains Python code
     # this type of line with the hashtag is a comment. It does not execute
     #click the play button above to run the cell
     #What do you see below the cell?

     #print to output
     print("Hello!")
```

```
Hello!
```

Python Basics: Math



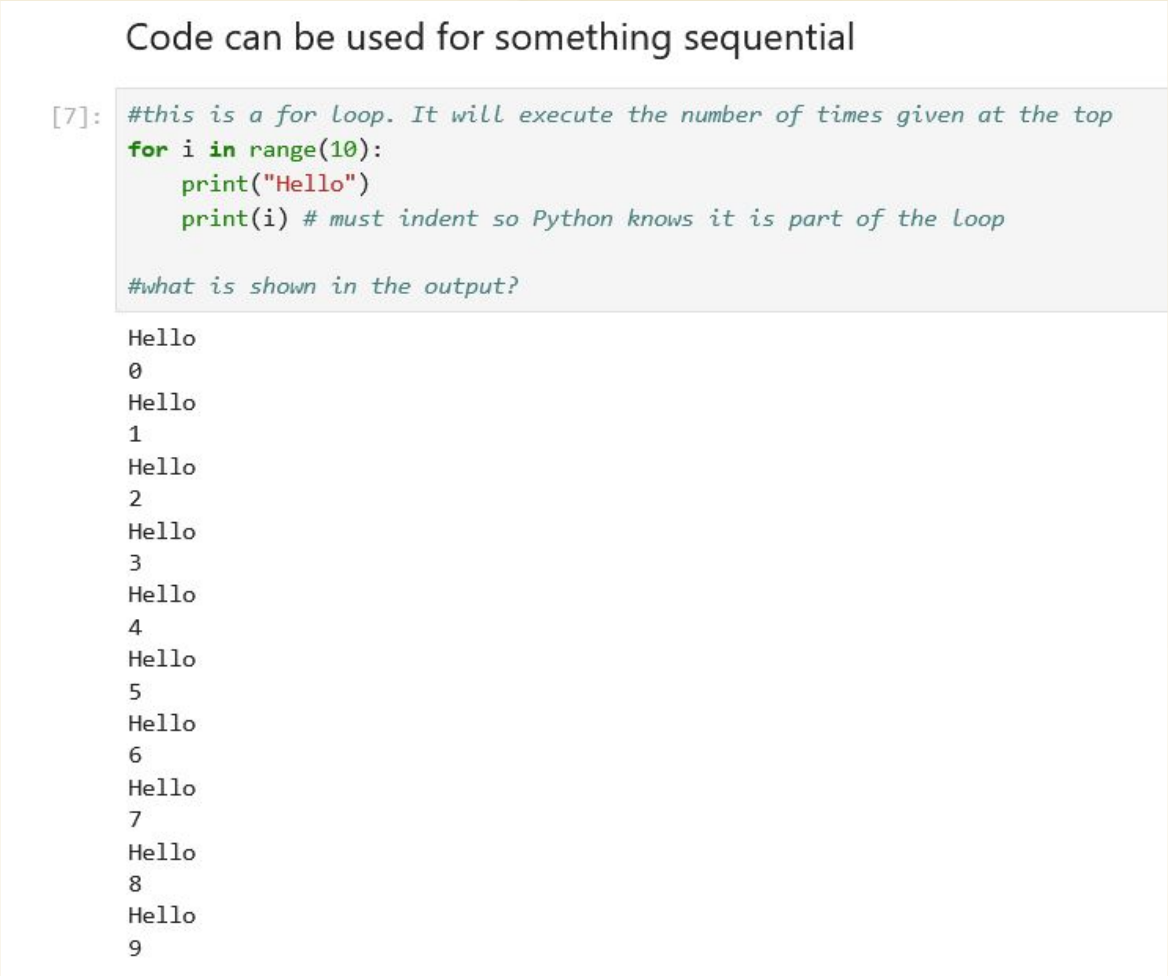
You can do mathematical functions.

[3]:

```
#You can do math in Python
#What do you see below the cell?

#add 1 + 1 and store the value in a varaible called x
x = 1 + 1

#print to output
print("the value of x is:", x)
```

```
the value of x is: 2
```

› Python Basics: Looping

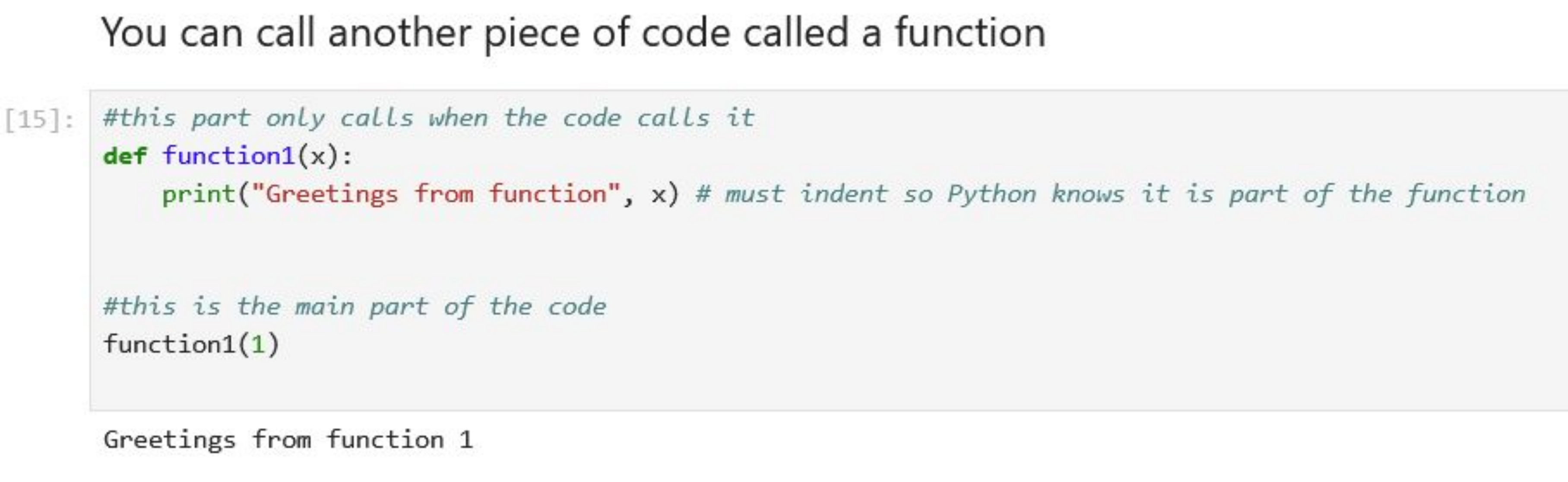


Code can be used for something sequential

```
[7]: #this is a for loop. It will execute the number of times given at the top
     for i in range(10):
         print("Hello")
         print(i) # must indent so Python knows it is part of the loop

     #what is shown in the output?
```

```
Hello
0
Hello
1
Hello
2
Hello
3
Hello
4
Hello
5
Hello
6
Hello
7
Hello
8
Hello
9
```

Python Basics: Functions

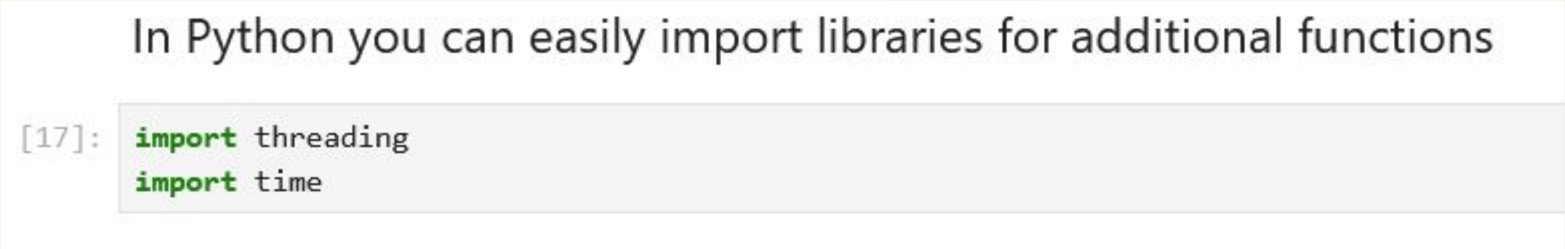


You can call another piece of code called a function

[15]:
```python
#this part only calls when the code calls it
def function1(x):
    print("Greetings from function", x) # must indent so Python knows it is part of the function


#this is the main part of the code
function1(1)
```

```
Greetings from function 1
```

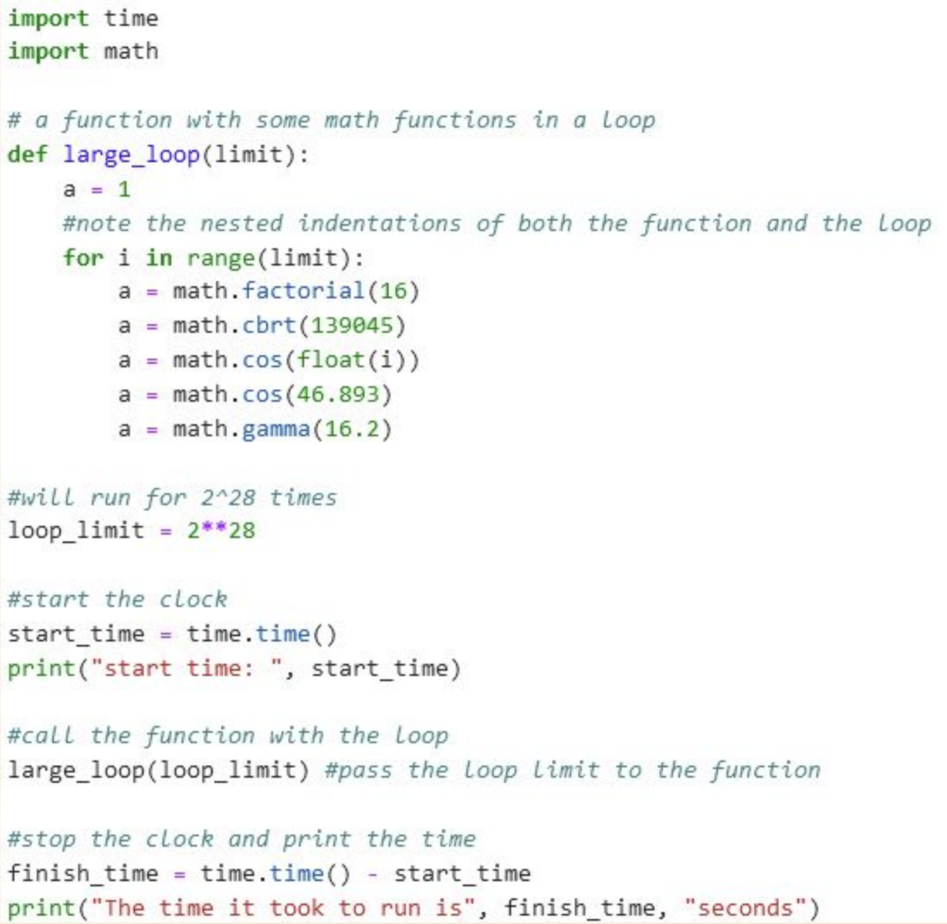
Python Basics: Libraries



In Python you can easily import libraries for additional functions

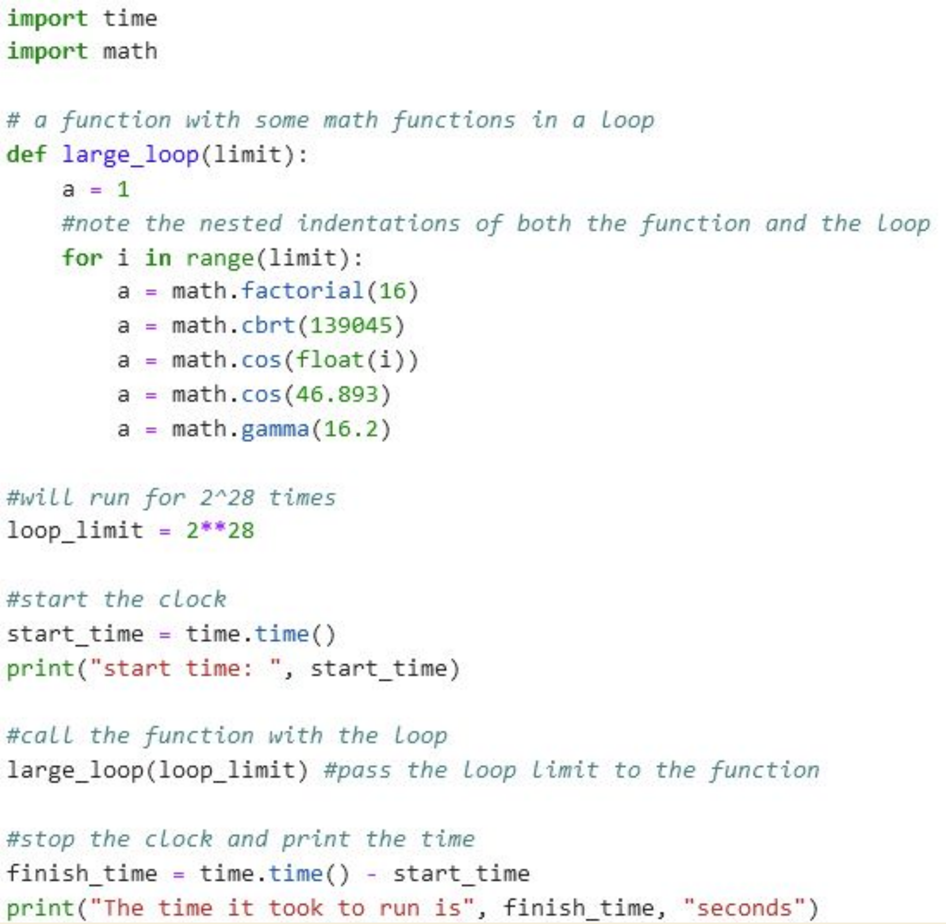
```python
import threading
import time
```

Stress the CPU: Some Math in Serial



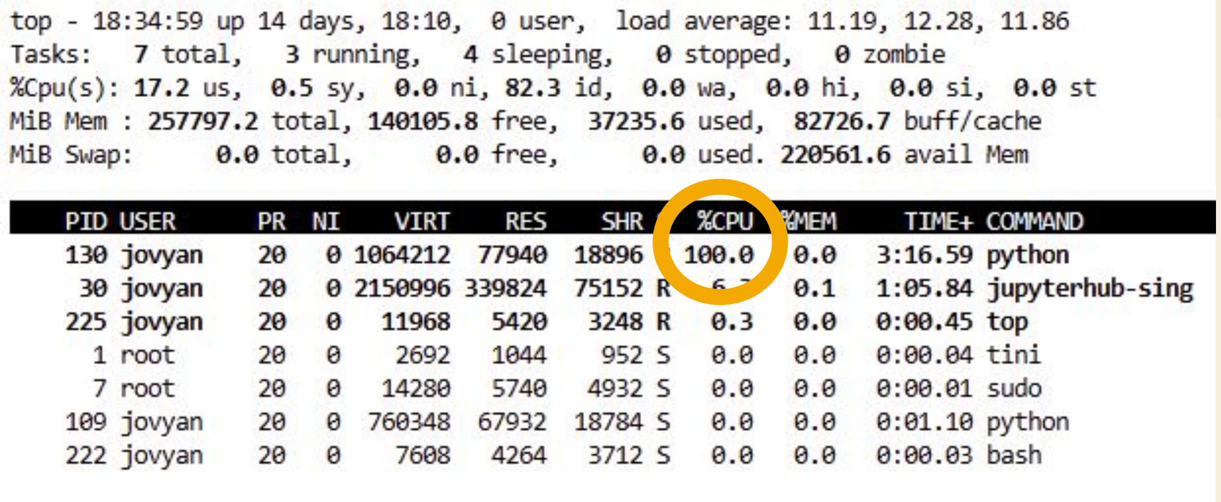
```python
import time
import math

# a function with some math functions in a loop
def large_loop(limit):
    a = 1
    #note the nested indentations of both the function and the loop
    for i in range(limit):
        a = math.factorial(16)
        a = math.cbrt(139045)
        a = math.cos(float(i))
        a = math.cos(46.893)
        a = math.gamma(16.2)

#will run for 2^28 times
loop_limit = 2**28

#start the clock
start_time = time.time()
print("start time: ", start_time)

#call the function with the loop
large_loop(loop_limit) #pass the loop limit to the function

#stop the clock and print the time
finish_time = time.time() - start_time
print("The time it took to run is", finish_time, "seconds")
```

Stress the CPU: Some Math in Serial



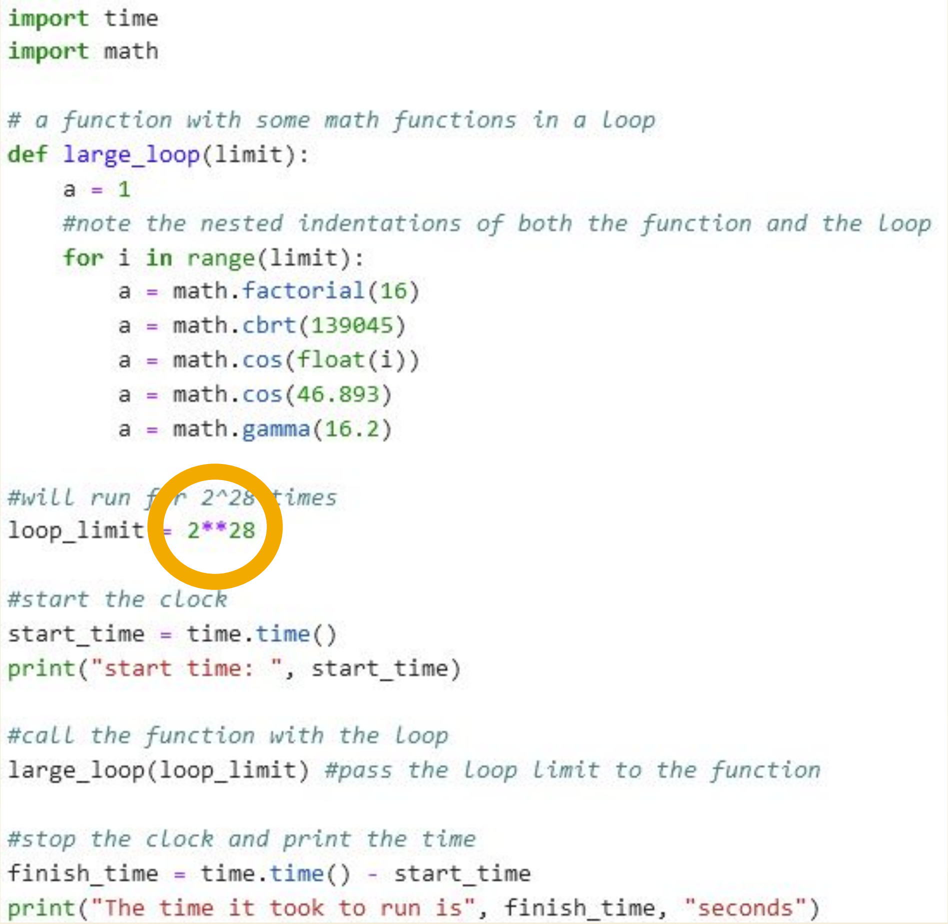
```python
import time
import math

# a function with some math functions in a loop
def large_loop(limit):
    a = 1
    #note the nested indentations of both the function and the loop
    for i in range(limit):
        a = math.factorial(16)
        a = math.cbrt(139045)
        a = math.cos(float(i))
        a = math.cos(46.893)
        a = math.gamma(16.2)

#will run for 2^28 times
loop_limit = 2**28

#start the clock
start_time = time.time()
print("start time: ", start_time)

#call the function with the loop
large_loop(loop_limit) #pass the loop limit to the function

#stop the clock and print the time
finish_time = time.time() - start_time
print("The time it took to run is", finish_time, "seconds")
```

- Running 1 CPU only



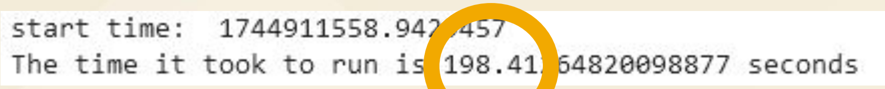
```
top - 18:34:59 up 14 days, 18:10,  0 user,  load average: 11.19, 12.28, 11.86
Tasks:   7 total,   3 running,   4 sleeping,   0 stopped,   0 zombie
%Cpu(s): 17.2 us,  0.5 sy,  0.0 ni, 82.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257797.2 total, 140105.8 free,  37235.6 used,  82726.7 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 220561.6 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  130 jovyan    20   0 1064212  77940  18896 R 100.0   0.0   3:16.59 python
   30 jovyan    20   0 2150996 339824  75152 R   6.7   0.1   1:05.84 jupyterhub-sing
  225 jovyan    20   0   11968   5420   3248 R   0.3   0.0   0:00.45 top
    1 root      20   0    2692   1044    952 S   0.0   0.0   0:00.04 tini
    7 root      20   0   14280   5740   4932 S   0.0   0.0   0:00.01 sudo
  109 jovyan    20   0  760348  67932  18784 S   0.0   0.0   0:01.10 python
  222 jovyan    20   0    7608   4264   3712 S   0.0   0.0   0:00.03 bash
```

› Stress the CPU: Some Math in Serial



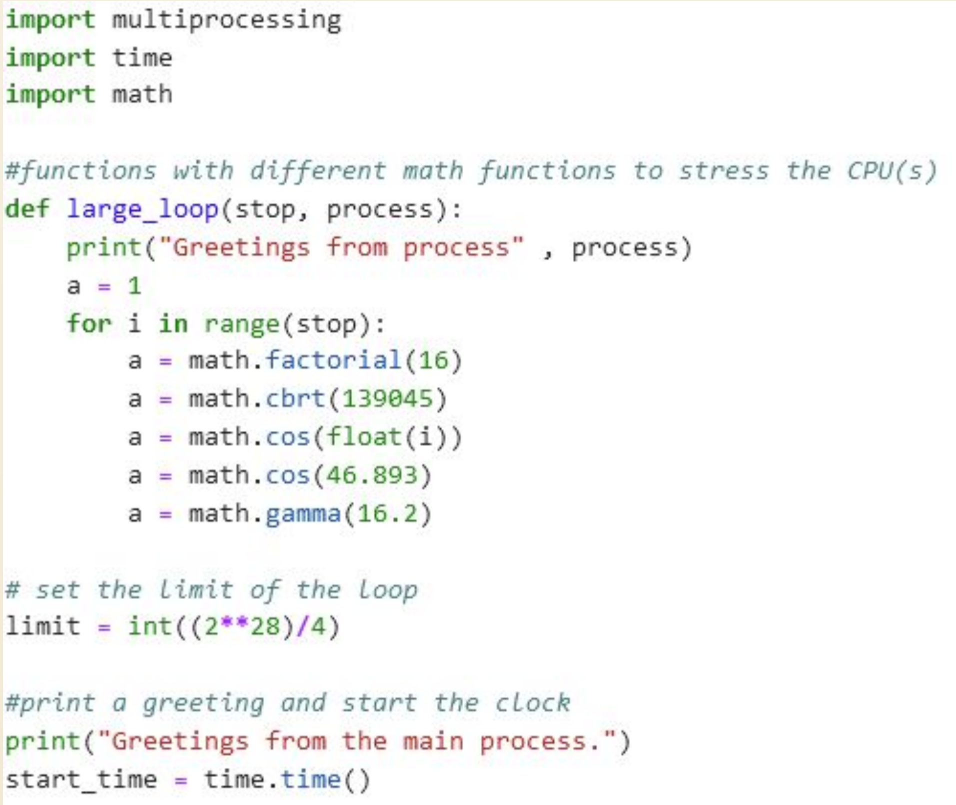
```python
import time
import math

# a function with some math functions in a loop
def large_loop(limit):
    a = 1
    #note the nested indentations of both the function and the loop
    for i in range(limit):
        a = math.factorial(16)
        a = math.cbrt(139045)
        a = math.cos(float(i))
        a = math.cos(46.893)
        a = math.gamma(16.2)

#will run for 2^28 times
loop_limit = 2**28

#start the clock
start_time = time.time()
print("start time: ", start_time)

#call the function with the loop
large_loop(loop_limit) #pass the loop limit to the function

#stop the clock and print the time
finish_time = time.time() - start_time
print("The time it took to run is", finish_time, "seconds")
```

- 198 seconds for 1 CPU

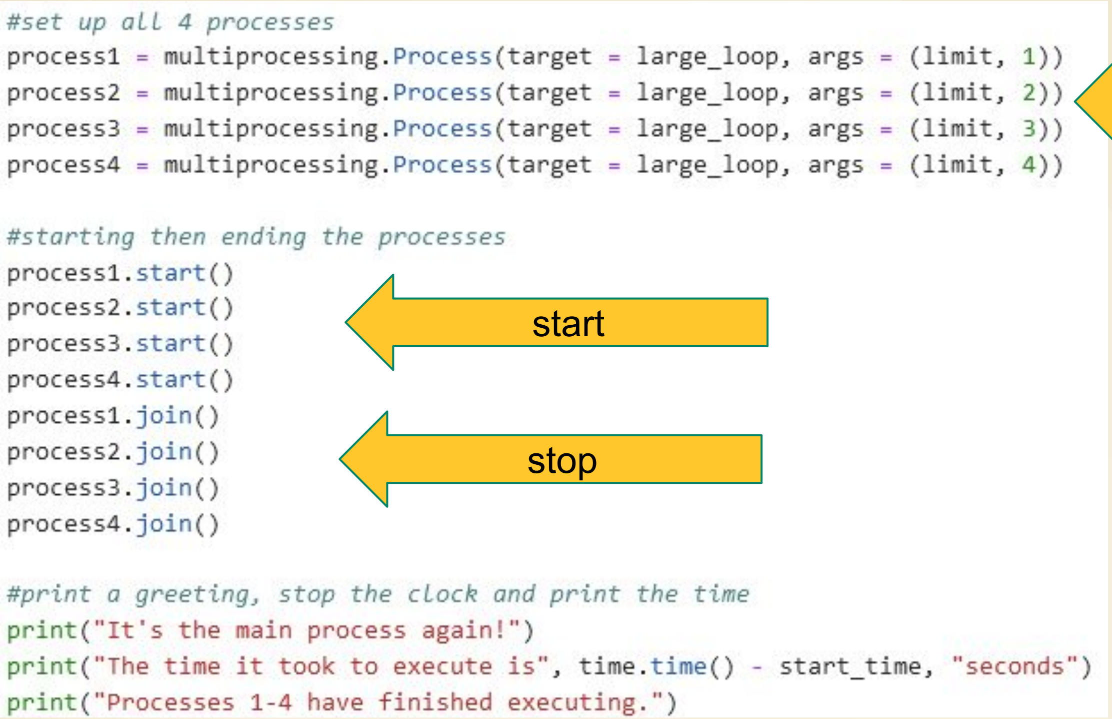


```
start time:  1744911558.942[illegible]457
The time it took to run is 198.41[illegible]64820098877 seconds
```

Multiprocessing



```
import multiprocessing
import time
import math

#functions with different math functions to stress the CPU(s)
def large_loop(stop, process):
    print("Greetings from process" , process)
    a = 1
    for i in range(stop):
        a = math.factorial(16)
        a = math.cbrt(139045)
        a = math.cos(float(i))
        a = math.cos(46.893)
        a = math.gamma(16.2)

# set the limit of the loop
limit = int((2**28)/4)

#print a greeting and start the clock
print("Greetings from the main process.")
start_time = time.time()
```



```
#set up all 4 processes
process1 = multiprocessing.Process(target = large_loop, args = (limit, 1))
process2 = multiprocessing.Process(target = large_loop, args = (limit, 2))
process3 = multiprocessing.Process(target = large_loop, args = (limit, 3))
process4 = multiprocessing.Process(target = large_loop, args = (limit, 4))

#starting then ending the processes
process1.start()
process2.start()
process3.start()
process4.start()
process1.join()
process2.join()
process3.join()
process4.join()

#print a greeting, stop the clock and print the time
print("It's the main process again!")
print("The time it took to execute is", time.time() - start_time, "seconds")
print("Processes 1-4 have finished executing.")
```

4 processes

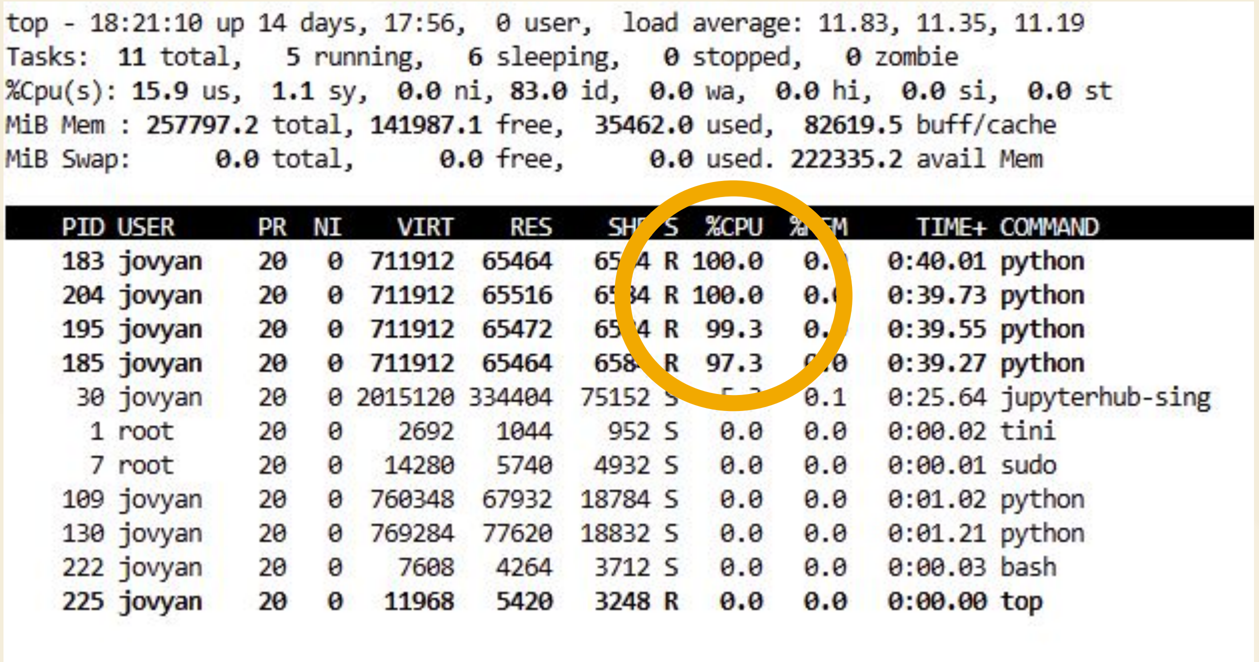
start

stop

- break up the code into 4 parts and run in parallel

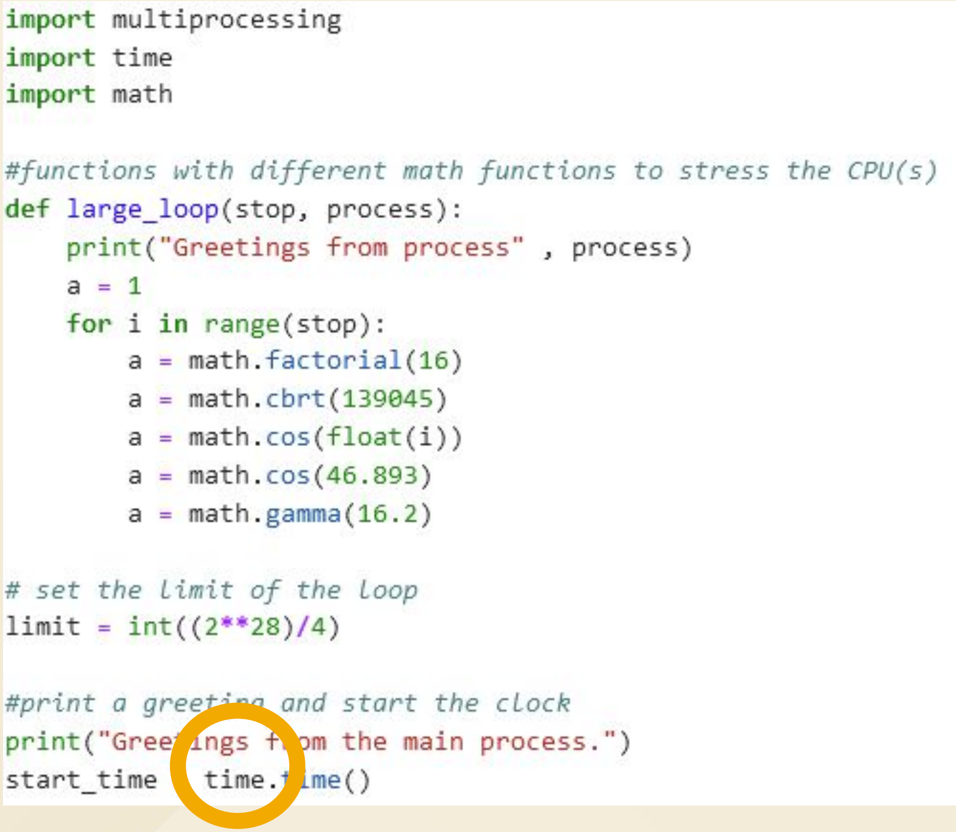
› Multiprocessing

- Running 4 CPUs simultaneously



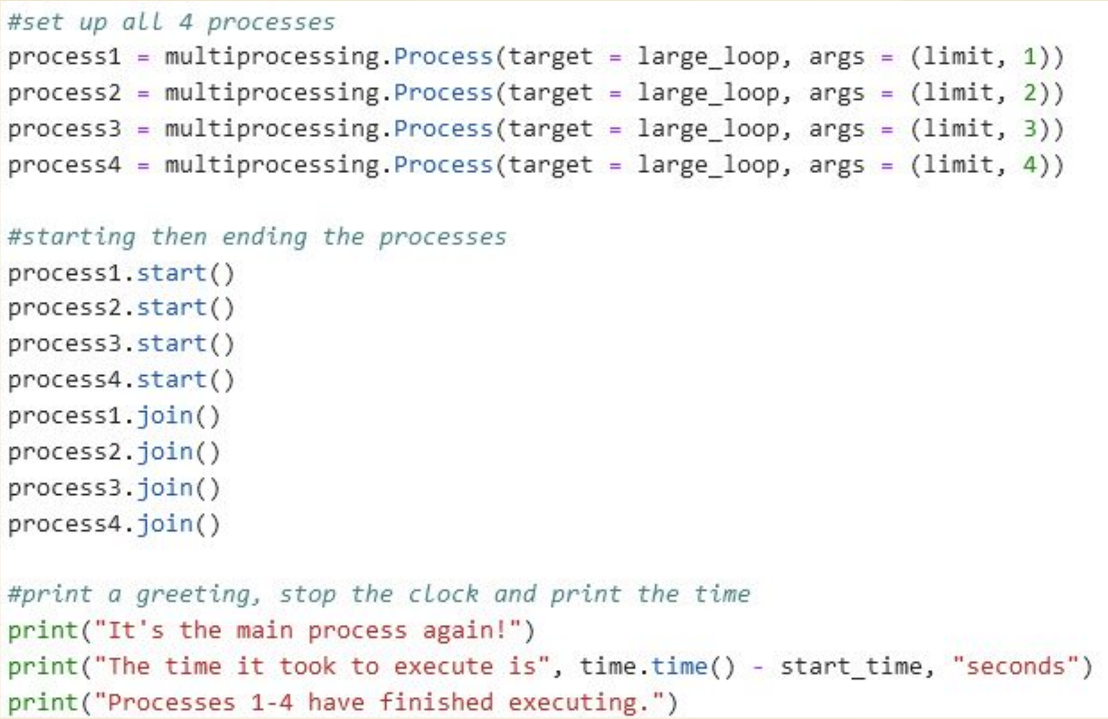
```
top - 18:21:10 up 14 days, 17:56,  0 user,  load average: 11.83, 11.35, 11.19
Tasks:  11 total,   5 running,   6 sleeping,   0 stopped,   0 zombie
%Cpu(s): 15.9 us,  1.1 sy,  0.0 ni, 83.0 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem : 257797.2 total, 141987.1 free,  35462.0 used,  82619.5 buff/cache
MiB Swap:      0.0 total,      0.0 free,      0.0 used. 222335.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    183 jovyan    20   0  711912  65464   6584 R 100.0   0.0   0:40.01 python
    204 jovyan    20   0  711912  65516   6584 R 100.0   0.0   0:39.73 python
    195 jovyan    20   0  711912  65472   6584 R  99.3   0.0   0:39.55 python
    185 jovyan    20   0  711912  65464   6584 R  97.3   0.0   0:39.27 python
     30 jovyan    20   0 2015120 334404  75152 S   5.3   0.1   0:25.64 jupyterhub-sing
      1 root      20   0    2692   1044    952 S   0.0   0.0   0:00.02 tini
      7 root      20   0   14280   5740   4932 S   0.0   0.0   0:00.01 sudo
    109 jovyan    20   0  760348  67932  18784 S   0.0   0.0   0:01.02 python
    130 jovyan    20   0  769284  77620  18832 S   0.0   0.0   0:01.21 python
    222 jovyan    20   0    7608   4264   3712 S   0.0   0.0   0:00.03 bash
    225 jovyan    20   0   11968   5420   3248 R   0.0   0.0   0:00.00 top
```

› Multiprocessing

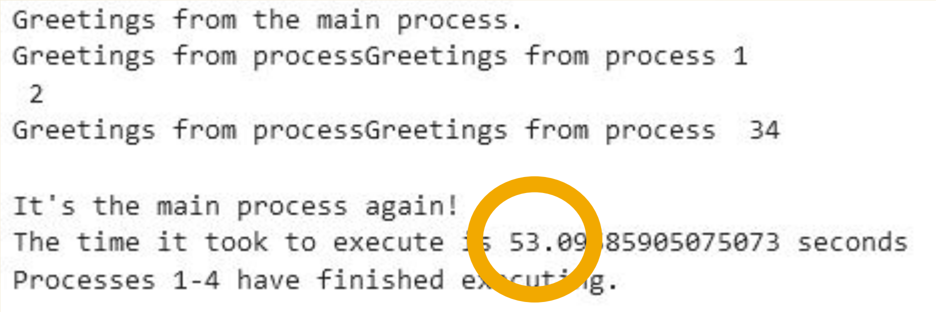


```
import multiprocessing
import time
import math

#functions with different math functions to stress the CPU(s)
def large_loop(stop, process):
    print("Greetings from process" , process)
    a = 1
    for i in range(stop):
        a = math.factorial(16)
        a = math.cbrt(139045)
        a = math.cos(float(i))
        a = math.cos(46.893)
        a = math.gamma(16.2)

# set the limit of the loop
limit = int((2**28)/4)

#print a greeting and start the clock
print("Greetings from the main process.")
start_time = time.time()
```

- About ¼ the time



```
#set up all 4 processes
process1 = multiprocessing.Process(target = large_loop, args = (limit, 1))
process2 = multiprocessing.Process(target = large_loop, args = (limit, 2))
process3 = multiprocessing.Process(target = large_loop, args = (limit, 3))
process4 = multiprocessing.Process(target = large_loop, args = (limit, 4))

#starting then ending the processes
process1.start()
process2.start()
process3.start()
process4.start()
process1.join()
process2.join()
process3.join()
process4.join()

#print a greeting, stop the clock and print the time
print("It's the main process again!")
print("The time it took to execute is", time.time() - start_time, "seconds")
print("Processes 1-4 have finished executing.")
```



```
Greetings from the main process.
Greetings from processGreetings from process 1
 2
Greetings from processGreetings from process  34

It's the main process again!
The time it took to execute is 53.09[illegible]85905075073 seconds
Processes 1-4 have finished executing.
```

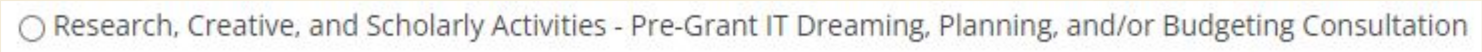
NRP Beyond JupyterHub

Margarete Walden Fisheries Simulation

- Laptop - runtime is hours
- Jupyter - runtime a few minutes
- Kubernetes - increase parameters, automation, multiple parallel runs
- Python automated Kubernetes - further increased automation, expanded scope of research

How to Contact Us

- SBS 413 (by appointment)
- Email: itsresearchsupport@humboldt.edu
- Create a HelpDesk ticket



○ Research, Creative, and Scholarly Activities - Pre-Grant IT Dreaming, Planning, and/or Budgeting Consultation

- Website: https://its.humboldt.edu/research/high-performance-computing
- Phone: x4100