

Introduction to Computational Physics

PHYS 250 (Autumn 2018) – Lecture 2

David Miller

Department of Physics and the Enrico Fermi Institute
University of Chicago

October 4, 2018

Outline

1 *Introduction*

2 *Software*

3 *Physics*

Computational Physics (PHYS 250)

Course Description PHYS 250 ([link to Course Catalog](#))

This course introduces the use of computers in the physical sciences. After an introduction to programming basics, we cover numerical solutions to fundamental types of problems, including cellular automaton, artificial neural networks, computer simulations of complex systems, and finite element analysis. Additional topics may include an introduction to graphical programming, with applications to data acquisition and device control.

There are an infinite number of paths that we might follow and still not deviate from this description. I therefore would like to lay out some of the principles that will guide me, and us, in how we navigate through those many possibilities.

Outline

1 *Introduction*

2 *Software*

3 *Physics*

Version control

- The most important message of this slide is simple...**Use a software version control system for all of your code**
 - And that means now...not tomorrow or next week
 - Because if you wait until you need it, it will be too late

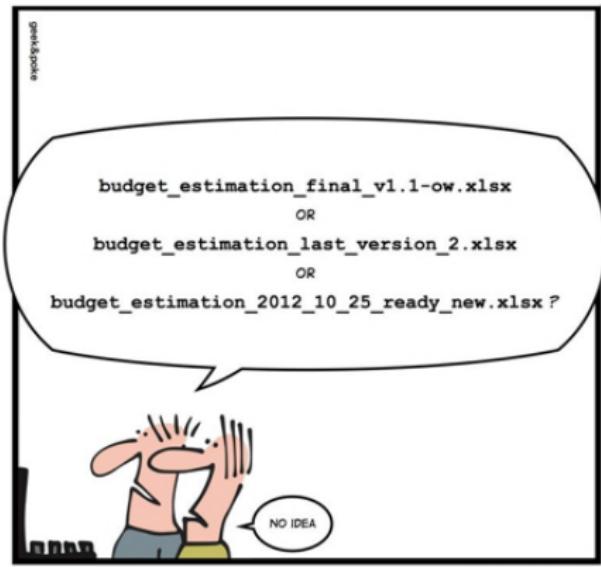
Version control

- The most important message of this slide is simple... **Use a software version control system for all of your code**

- And that means now...not tomorrow or next week
- Because if you wait until you need it, it will be too late



SIMPLY EXPLAINED



VERSION CONTROL

A brief history of version control

- **The first version control systems were designed to be used on large systems where everyone logged into the same machine**
 - They tracked code on the same filesystem where it lived (e.g., in a subdirectory)
 - SCCS and RCS are examples
- **Then client-server systems were developed, so that developers could work on their own machines**
 - Checking code into a central server to share and collaborate
 - CVS and SVN are examples
- **More recently distributed version control systems have arisen**
 - These are decentralised, so everyone has a complete copy of the repository
 - Gives a lot of freedom to developers to share and merge as they like, so liked very much by the open source community
 - git, mercurial and bit keeper are examples

git, GitHub, & GitLab

<https://git-scm.com>, <https://github.com>, <https://about.gitlab.com>

- **git is the most popular open source version control system**
 - can host huge projects (Linux Kernel, LHC experiment software, etc)
 - scales very well and it's extremely fast and powerful
 - very flexible (= complex)
- **Distributed version control systems (`git`) are great, but they're made even better by using a social coding site (`GitHub` or `GitLab`)**
- **These sites allow developers:**
 - browse code easily
 - compare different versions
 - take copies (a.k.a. fork)
 - offer patches back to upstream repositories
 - And discuss and review these patches before acceptance
 - even build websites
- **The best known social coding site is GitHub, but there are others, e.g. BitBucket and GitLab**
 - Familiarity with `git/GitHub/GitLab` will serve you well, trust me

GitHub & GitLab resources

GitHub is a free resource as long as your code remains public (you can pay for private repositories). The Physical Sciences Division (PSD) at UChicago hosts a **private GitLab** repository.

- <https://psdcomputing.uchicago.edu/page/psd-repo>



PSD Repo



PSD Repo is a software source code repository managed by the PSD Computing office

UC LDAP	Standard
UC LDAP Username	
<input type="text" value="johndoe"/>	
Password	
<input type="password" value="*****"/>	
<input type="checkbox"/> Remember me	
Sign in	

PHYS 250 GitHub

<https://github.com/UChicagoPhysics/PHYS250>

Course materials are hosted in the **GitHub** UChicagoPhysics repository

The screenshot shows the GitHub repository page for 'UChicagoPhysics / PHYS250'. At the top, there are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below that is a navigation bar with links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Insights', and 'Settings'. The main content area is titled 'University of Chicago PHYS 250 Computational Physics software repository' and includes a 'Manage topics' link and an 'Edit' button. Below this, it displays statistics: '15 commits', '1 branch', '0 releases', and '1 contributor'. A 'Branch: master' dropdown and a 'New pull request' button are also present. The commit history table lists the following changes:

File	Description	Time
fizist	Update example	Latest commit efe9be5 25 minutes ago
Examples	Update example	25 minutes ago
LearningGoals	UPdate Learning Goals	22 hours ago
Slides	Update Lecture 1 Slides	an hour ago
Syllabus	Updates to syllabus	22 hours ago
.gitignore	Update slides for day 1	3 days ago
README.md	Update README.md	3 days ago

- Slides (e.g. *these!*), syllabi, learning goals, and code examples
- Stable versions will be cross-posted to **Canvas** as well.
- Homework submission will be done via **GitHub** (*instructions to come*)

Linux “shell”

- We will be using an interface to Linux called a “shell”
- It is a command-line interpreter : you type, it executes
- Two major options are bash (as in, smash) and csh (like “sea shell”, modern version is “tcsh”, “tea sea shell”)
 - Only real difference: environment variables syntax
 - bash: `export X=value`
 - csh: `setenv X value`

Shell basics

Listing directory contents : ls, like “list”

```
> ls
```

```
Examples/ LearningGoals/ README.md Slides/  
Syllabus/ global.sty
```

Copy: cp

```
> cp stuff.txt stuff1.txt
```

Where am I?: pwd, cd

```
> pwd
```

```
/ComputationalPhysics/PHYS250/PHYS250-Autumn2018
```

```
> ls
```

```
Examples/ LearningGoals/ README.md Slides/ Syllabus/
```

```
> cd Examples/
```

```
> ls
```

```
HelloGaussian.ipynb
```

```
HelloGaussian.py
```

```
Introduction_to_Jupyter_Notebooks_and_Python.ipynb
```

Hello world!

Interactive in the python interpreter

```
python
>>> print "hello world"
hello
>>> CTRL-d # to exit python
```

From a script (containing the above print line):

```
python helloworld.py
```

Self-running script:

```
#!/usr/bin/env python
# This script prints hello to the screen
print "hello world"
```

```
chmod +x helloworld.py
./helloworld.py
hello world
```

Lists (I)

In my opinion, python's great advantage is **list comprehension**.

List basics

```
v = []      # empty list
v = list()  # empty list
v = [ 1, 2, 4, 5 ] ; v = [ 'a', 'b', 'c' ]
v = range(4,10,2) # results in [ 4, 6, 8 ]
v = [ 4, 2.5, 'Hi', [ 1,3,5 ] ] # can mix types
```

Append elements

```
>>> v.append( 70 )
>>> print v
```

Concatenation

```
>>> v += [ 'some', 'more', 'elements' ]
>>> v    # shows the object
```

Removal of elements

```
>>> v.remove(2.5)
>>> del v[0]
```

Lists (II)

Element acces read/write

```
>>> v[0]  
'hi'  
>>> v[0] = 'hey'  
>>> v[-1] # last element. Negative = count from the end  
>>> v[1:3] # subrange by index (start index, one-beyond)
```

Test if an element is in a list (or not)

```
>>> if 4 in v:  
...     print "Found it"  
Found it  
>>> if 200 not in v:  
...     print "Not found"  
Not found
```

for and while loops

The `for` statement iterates through a collection, iterable object or generator function.

The `while` statement merely loops until a condition is `False`.

Iterate over list

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Iterate using built-in `range` function

```
for x in range(0, 3):
    print("We're on time %d" % (x))
```

Iterate until a condition is met

```
count = 0
while count < 5:
    print(count)
    count += 1 # Same as: count = count + 1
```

Putting lists and loops together is amazing (and complex)

Filter one list into another (the “old” way)

```
newlist = []
for i in oldlist:
    if filter(i):
        newlist.append(function(i))
```

List comprehension (the “pythonic” way)

```
newlist = [function(i) for i in oldlist if filter(i)]
```

where filter and function just perform “some” operation on the list elements. Basically, the syntax is:

```
[ expression for item in list if conditional ]
```

and this replaces:

```
for item in list:
    if conditional:
        expression
```

Useful list comprehension

Filter one list into another (the “old” way)

```
>>> v = [ x**2 for x in range(10) if x % 3 == 0 ]  
>>> v  
[0, 9, 36, 81]
```

List comprehension (the “pythonic” way)

```
newlist = [function(i) for i in oldlist if filter(i)]
```

where filter and function just perform “some” operation on the list elements. Basically, the syntax is:

```
[ expression for item in list if conditional ]
```

and this replaces:

```
for item in list:  
    if conditional:  
        expression
```

Hello Gaussian!

Basic but useful code example

```
import numpy as np
import matplotlib.pyplot as plt

def p(x):
    return np.exp(-x**2)

#let's plot it
x = np.linspace(-3,3,100)
y = p(x)
plt.plot(x,y)
plt.show()
```

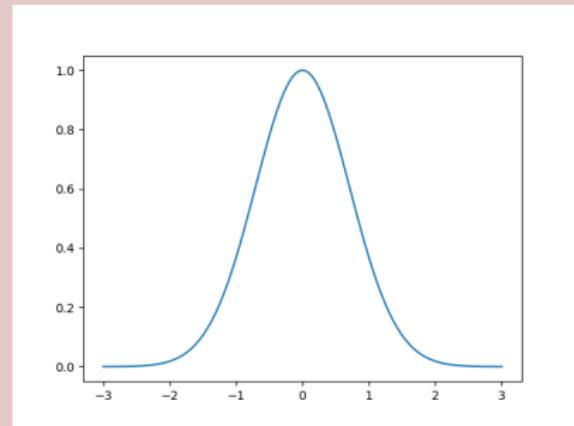
Hello Gaussian!

Basic but useful code example

```
import numpy as np  
import matplotlib.pyplot as plt
```

```
def p(x):  
    return np.exp(-x**2)
```

```
#let's plot it  
x = np.linspace(-3,3,100)  
y = p(x)  
plt.plot(x,y)  
plt.show()
```



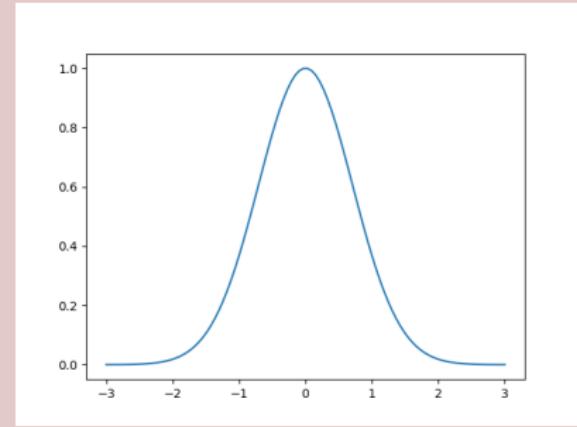
Hello Gaussian!

Basic but useful code example

```
import numpy as np  
import matplotlib.pyplot as plt
```

```
def p(x):  
    return np.exp(-x**2)
```

```
#let's plot it  
x = np.linspace(-3, 3, 100)  
y = p(x)  
plt.plot(x, y)  
plt.show()
```



But what about that `linspace` thingy? Google it! ([numpy docs](#))

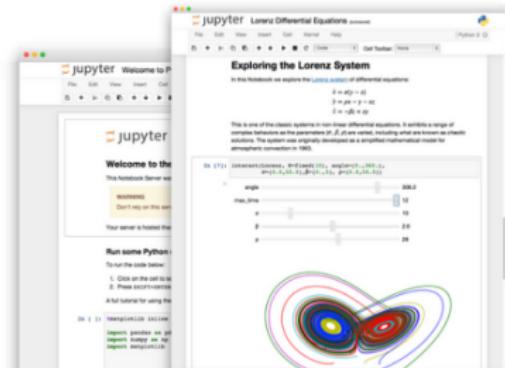
```
numpy.linspace(start, stop, num=50, endpoint=True,  
retstep=False, dtype=None)
```

“Returns num evenly spaced samples, calculated over the interval [start, stop].”

Jupyter notebooks

Interactive, web-based, integrated code and documentation environment

We will be following-up with more technical practice with python, but I want to introduce you to the resources that we'll be using this quarter for many of our examples and projects: **Jupyter notebooks**.



The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

[Try it in your browser](#)

[Install the Notebook](#)



Language of choice



Share notebooks



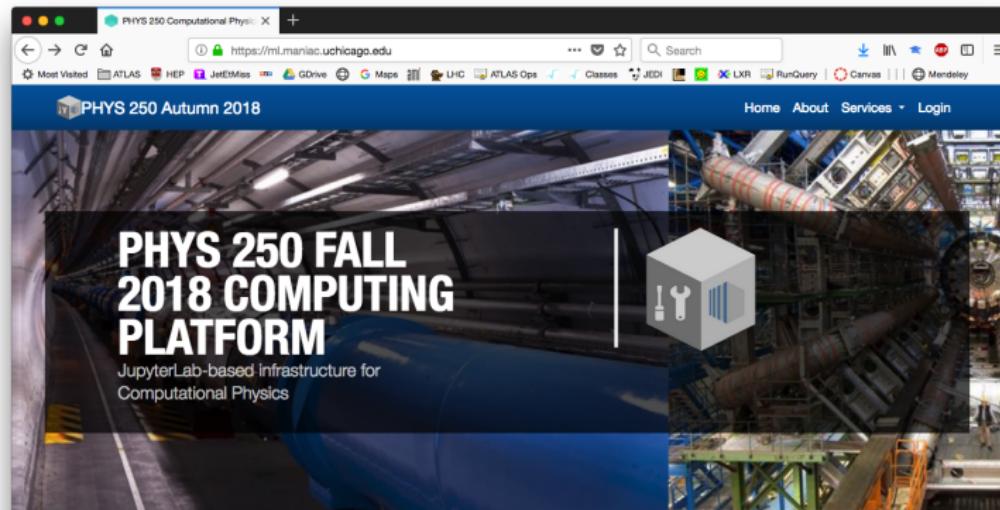
Interactive output



Big data integration

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



The screenshot shows a web browser window with the title "PHYS 250 Computational Phys: X". The URL is https://ml.maniac.uchicago.edu. The page content features a large image of the ATLAS particle detector at CERN. Overlaid on the image is the text "PHYS 250 FALL 2018 COMPUTING PLATFORM" in large white letters, with "JupyterLab-based infrastructure for Computational Physics" in smaller text below it. To the right of the text is a white cube icon with a wrench and a vertical bar. The top navigation bar includes links for Home, About, Services, and Login.

Purpose

A computational platform that supports on-demand JupyterLab instances for interactive python development as well as advanced computational resources such as those required for high-level, compute-intensive machine learning applications.

Elements

The platform provides hosted JupyterLab instances for the students in PHYS 250 (Autumn 2018) on GPU resources hosted by the computing center for the ATLAS Experiment

<input checked="" type="checkbox"/> External resources
Intro to Linux (UChicago CSIL)
Intro to Git (UChicago CSIL)
PICUP (Partnership for Integration of Computation into Undergraduate Physics)
Computational Physics text from

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window with the following details:

- Title Bar:** "Log in using Globus" and the URL "https://auth.globus.org/p/login?scope=um%3Aglobus%3A".
- Toolbar:** Standard browser icons for back, forward, search, and refresh, along with a list of most visited sites including ATLAS, HEP, JetEdTmes, GDrive, Google Maps, LHC, ATLAS Ops, Classes, JEDI, RunQuery, Canvas, and Mendeley.
- Globus Logo:** A blue "g" icon followed by the word "globus".
- Page Content:**
 - Section Header:** "Globus Account Log In".
 - Text:** "Log in to use ml front".
 - Text:** "Use your existing organizational login" and "e.g., university, national lab, facility, project".
 - Input Field:** "University of Chicago" with a dropdown arrow.
 - Text:** "Didn't find your organization? Then use [Globus ID](#) to sign in. (What's this?)".
 - Button:** "Continue" in a blue box.
 - Information Box:** "Globus uses CILogon to enable you to Log in from this organization. By clicking Continue, you agree to the CILogon privacy policy and you agree to share your username, email address, and affiliation with CILogon and Globus. You also agree for CILogon to issue a certificate that allows Globus to act on your behalf." It features the CILogon logo (a green 'C' with an upward arrow).
 - Text:** "Or".
 - Buttons:** "Sign in with Google" and "Sign in with ORCID ID".

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window for the PHYS 250 Computational Physics course. The URL is <https://ml.maniac.uchicago.edu/index.html>. The page features a large banner image of the ATLAS particle detector at the Large Hadron Collider. Overlaid on the banner is the text "PHYS 250 FALL 2018 COMPUTING PLATFORM" in large white letters, with "JupyterLab-based infrastructure for Computational Physics" in smaller text below it. To the right of the text is a white cube icon with a wrench and a gear inside. Below the banner, there's a section titled "Purpose" with a paragraph about the platform's purpose, followed by a section titled "Elements" with a paragraph about the platform's components. On the right side, there's a sidebar with links to "External resources" like "Intro to Linux (UChicago CSIL)" and "Intro to Git (UChicago CSIL)", and other resources like "PICUP (Partnership for Integration of Computation into Undergraduate Physics)" and "Computational Physics text from".

Purpose

A computational platform that supports on-demand JupyterLab instances for interactive python development as well as advanced computational resources such as those required for high-level, compute-intensive machine learning applications.

Elements

The platform provides hosted JupyterLab instances for the students in PHYS 250 (Autumn 2018) on GPU resources hosted by the computing center for the ATLAS Experiment

External resources

- [Intro to Linux \(UChicago CSIL\)](#)
- [Intro to Git \(UChicago CSIL\)](#)
- [PICUP \(Partnership for Integration of Computation into Undergraduate Physics\)](#)
- [Computational Physics text from](#)

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window with the title "PHYS 250 Computational Physicist". The URL in the address bar is <https://ml.maniac.uchicago.edu/profile.html>. The page content is for "PHYS 250 Autumn 2018". On the left, there is a large, semi-transparent watermark of a three-dimensional cube. On the right, there is a "Profile Information" box containing the following details:

Name:	David Miller
Email:	davemilr@uchicago.edu
Organization:	

A blue "Logout" button is located at the bottom of this box. At the bottom of the page, there is a copyright notice and logos for NSF and the U.S. Department of Energy.

© 2018 University of Chicago. This platform is supported by National Science Foundation grants: NSF OAC-1724821 "CIF21 DIBBs: El: SLATE and the Mobility of Capability", NSF CNS-1730158 "CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-CI)", NSF OAC-1541349 "OC*DNI DIBBs: The Pacific Research Platform", NSF PHY-1624739 "U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier", NSF PHY-1148698 "The Open Science Grid, The Next Five Years: Distributed High Throughput Computing for the Nation's Scientists, Researchers, Educators, and Students", the Department of Energy ASCR/NGNs DORM project "Virtual Clusters for Community Computation (VC3)", and by the Enrico Fermi Institute at the University of Chicago.

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window with the URL <https://ml.maniac.uchicago.edu/profile.html>. The page title is "PHYS 250 Computational Physics". The main content area displays a large, wireframe-style 3D cube. To the right, there is a sidebar titled "Profile Information" containing the user's name (David Miller), email (davemlr@uchicago.edu), and organization (University of Chicago). Below this is a "Logout" button. A dropdown menu titled "Profile" is open, showing options like "My Services", "Monitor", "Public Lab", and "Users". At the bottom of the page, there is a copyright notice from the University of Chicago and logos for NSF and DOE.

© 2018 University of Chicago. This platform is supported by National Science Foundation grants: NSF OAC-1724821 "CIF21 DIBBs: El: SLATE and the Mobility of Capability", NSF CNS-1730158 "El-New: Cognitive Hardware and Software Ecosystem Community Infrastructure [CHASE-El]", NSF OAC-1541349 "CC'DNI DIBBs: The Pacific Research Platform", NSF PHY-1624739 "U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier", NSF PHY-1148698 "The Open Science Grid, The Next Five Years: Distributed High Throughput Computing for the Nation's Scientists, Researchers, Educators, and Students", the Department of Energy ASCR/NGNS DDRM project "Virtual Clusters for Community Computation (VC3)", and by the Enrico Fermi Institute at the University of Chicago.

U.S. DEPARTMENT OF
ENERGY | Office of
Science

<https://ml.maniac.uchicago.edu/services.html>

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window with the title "PHYS 250 Computational Physics" and the URL "https://ml.mariana.uchicago.edu/services.html". The browser's address bar also displays "https://ml.mariana.uchicago.edu/services.html". The page content is titled "Your deployments". It contains a message: "From this page you can manage your private services. Please delete a service when you don't need it anymore. To check if a service status changed simply refresh this page." Below this is a blue button labeled "New Personal JupyterLab". The main section is titled "Running Services" and features a table header with columns: Service, Name, Started at, Ending at, GPUs, Cores, Memory, Link, Status, and Command. A message "No data available in table" is displayed below the header. Below the table is a message "Showing 0 to 0 of 0 entries". The next section is titled "All Services" and has a similar table structure. Its message "No data available in table" is followed by "Showing 0 to 0 of 0 entries". At the bottom of the page is a copyright notice: "© 2018 University of Chicago. This platform is supported by National Science Foundation grants: NSF OAC-1724821 "CIF21 DIBBs: El: SLATE and the Mobility of Capability", NSF CNS-1730158 "CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-El)", NSF OAC-1541349 "CC*DNI DIBBs: The Pacific Research Platform", NSF PHY-1624739 "U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier", NSF PHY-1148698 "The Open Science Grid, The Next Five Years: Distributed High Throughput Computing for the Nation's Scientists, Researchers, Educators, and Students", the Department of Energy ASCR/NGNs DDRM project "Virtual Clusters for Community Computation (VC4C)", and by the Environ Fermi Institute of the University of Chicago".

Service	Name	Started at	Ending at	GPUs	Cores	Memory	Link	Status	Command
No data available in table									

Showing 0 to 0 of 0 entries

Service	Name	Started at	Ending at	GPUs	Cores	Memory
No data available in table						

Showing 0 to 0 of 0 entries

© 2018 University of Chicago. This platform is supported by National Science Foundation grants: NSF OAC-1724821 "CIF21 DIBBs: El: SLATE and the Mobility of Capability", NSF CNS-1730158 "CI-New: Cognitive Hardware and Software Ecosystem Community Infrastructure (CHASE-El)", NSF OAC-1541349 "CC*DNI DIBBs: The Pacific Research Platform", NSF PHY-1624739 "U.S. ATLAS Operations: Discovery and Measurement at the Energy Frontier", NSF PHY-1148698 "The Open Science Grid, The Next Five Years: Distributed High Throughput Computing for the Nation's Scientists, Researchers, Educators, and Students", the Department of Energy ASCR/NGNs DDRM project "Virtual Clusters for Community Computation (VC4C)", and by the Environ Fermi Institute of the University of Chicago.

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window with the title "PHYS 250 Computational Physics" and the URL "https://ml.merac.uchicago.edu/PrivateJupyter.html". The page is titled "PHYS 250 Autumn 2018" and includes a "Home" link and a "Profile" link in the top right. On the left, there is a section titled "Private JupyterLab" with instructions: "Use this for private code development or if you need dedicated resources." Below this is a section titled "Instructions (read first)" containing a bulleted list of guidelines. On the right, there is a form titled "Configure your JupyterLab instance" with fields for "Name" (set to "Lecture 1"), "Password" (set to "****"), "Expiration of your JupyterLab [days]" (set to "1"), "GPUs" (set to "0"), "CPUs" (set to "1"), "Memory [GB]" (set to "8"), and a "Check out a GitHub repo (use full URL including ".git")" field. A blue "START" button is at the bottom of the form.

Private JupyterLab

Use this for private code development or if you need dedicated resources.

Instructions (read first)

- Fill out the form to the right.
- Upon submission a dedicated JupyterLab instance will be spawned in the background
- You'll receive a JupyterLab link to be used once the notebook has been scheduled (this can take several minutes, or longer if the resources are busy)
- We suggest organizing your notebook in GitHub and cloning it manually once your notebook starts up. Remember to push any changes before the notebook expires.

Configure your JupyterLab instance

Please only select what you actually need.

Name *

Password *

Expiration of your JupyterLab [days] *

GPUs

CPUs

Memory [GB]

Check out a GitHub repo (use full URL including ".git")

START

Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.

The screenshot shows a web browser window titled "PHYS 250 Computational Physics" with the URL <https://mi.manaic.uchicago.edu/services.html>. The page has a dark blue header with the title "PHYS 250 Autumn 2018" and navigation links for Home, About, Services, and Profile. Below the header, there's a section titled "Your deployments" with a sub-instruction: "From this page you can manage your private services. Please delete a service when you don't need it anymore. To check if a service status changed simply refresh this page." A blue button labeled "New Personal JupyterLab" is visible. The main content area is divided into two sections: "Running Services" and "All Services".

Your deployments

From this page you can manage your private services. Please delete a service when you don't need it anymore. To check if a service status changed simply refresh this page.

New Personal JupyterLab

Running Services

Service	Name	Started at	Ending at	GPUs	Cores	Memory	Link	Status	Command
Private JupyterLab	lecture-1	Tue, 02 Oct 2018 14:31:14 GMT	Wed, 03 Oct 2018 14:31:14 GMT	0	1	8Gi	http://mi.usatlas.org:31155	Running	Delete

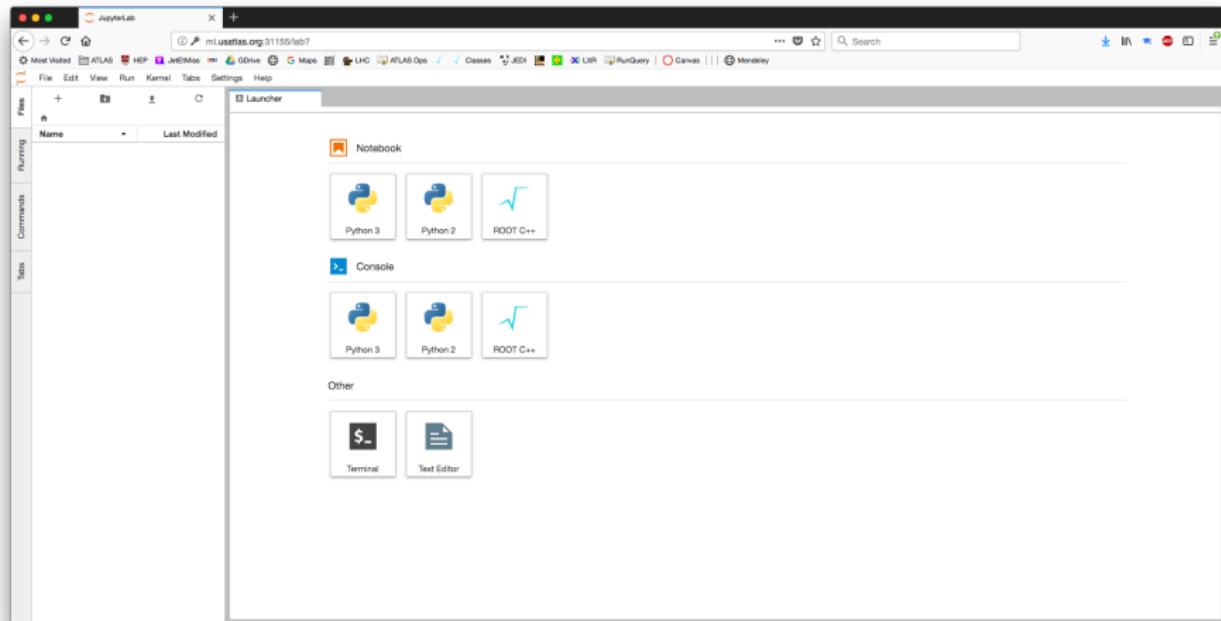
Showing 1 to 1 of 1 entries

All Services

Service	Name	Started at	Ending at	GPUs	Cores	Memory
Private JupyterLab	lecture-1	Tue, 02 Oct 2018 14:31:14 GMT	Wed, 03 Oct 2018 14:31:14 GMT	0	1	8
Private JupyterLab	instructor-lab	Mon, 01 Oct 2018 18:40:41 GMT	Tue, 02 Oct 2018 18:40:41 GMT	0	1	8
Private JupyterLab	test-3	Fri, 28 Sep 2018 20:07:37 GMT	Sat, 29 Sep 2018 20:07:37 GMT	0	1	8
Private JupyterLab	my-lab 2	Fri, 28 Sep 2018 19:59:02 GMT	Sat, 29 Sep 2018 19:59:02 GMT	0	1	8
Private JupyterLab	my-lab 2	Fri, 28 Sep 2018 19:58:19 GMT	Sat, 29 Sep 2018 19:58:19 GMT	0	1	8

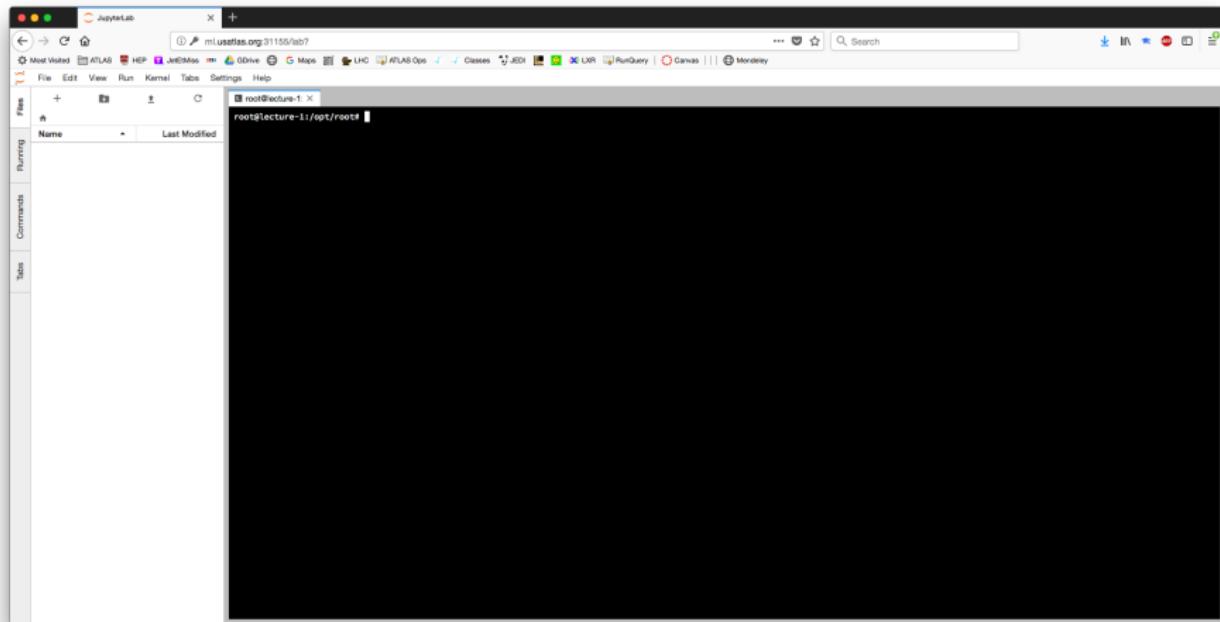
Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



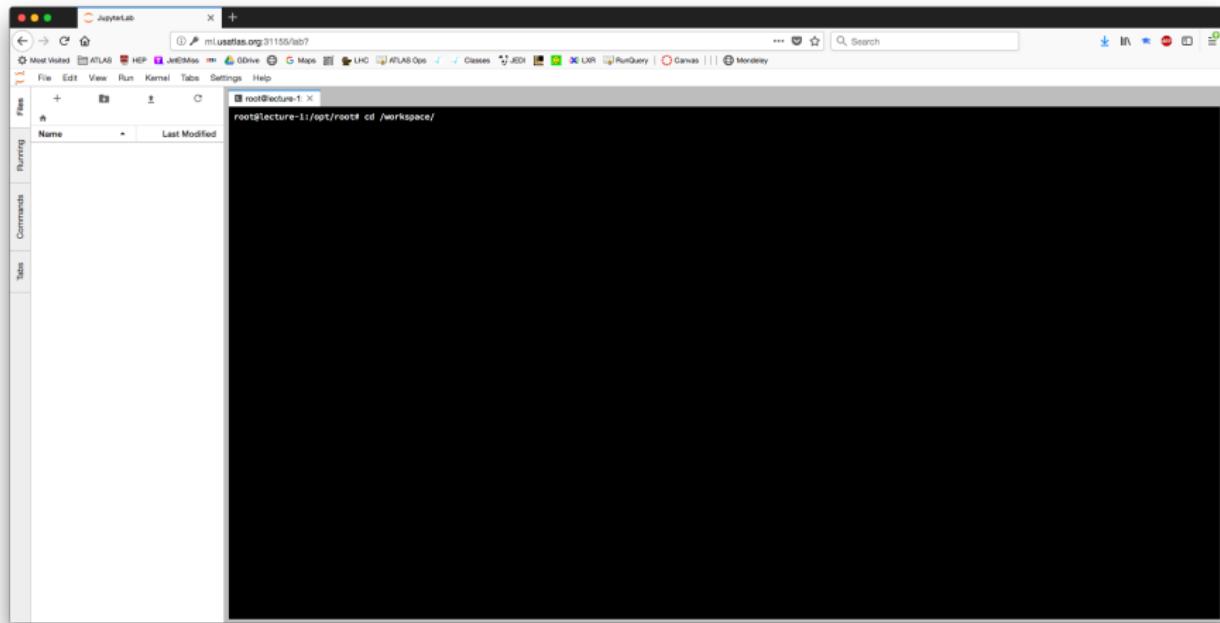
Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



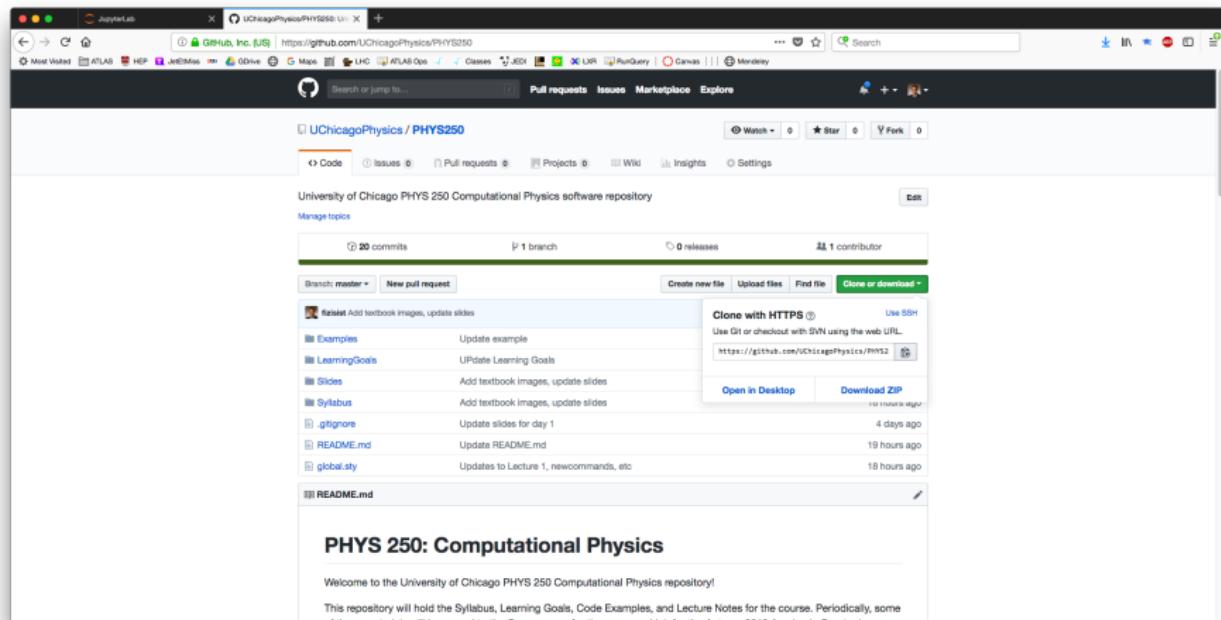
Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



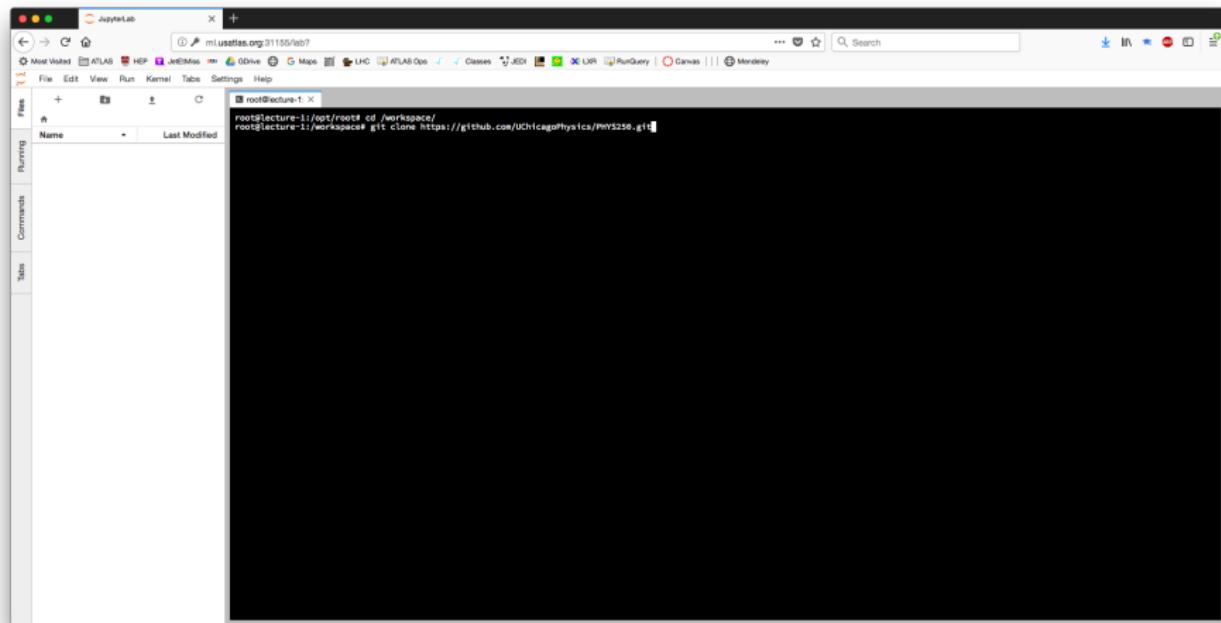
Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



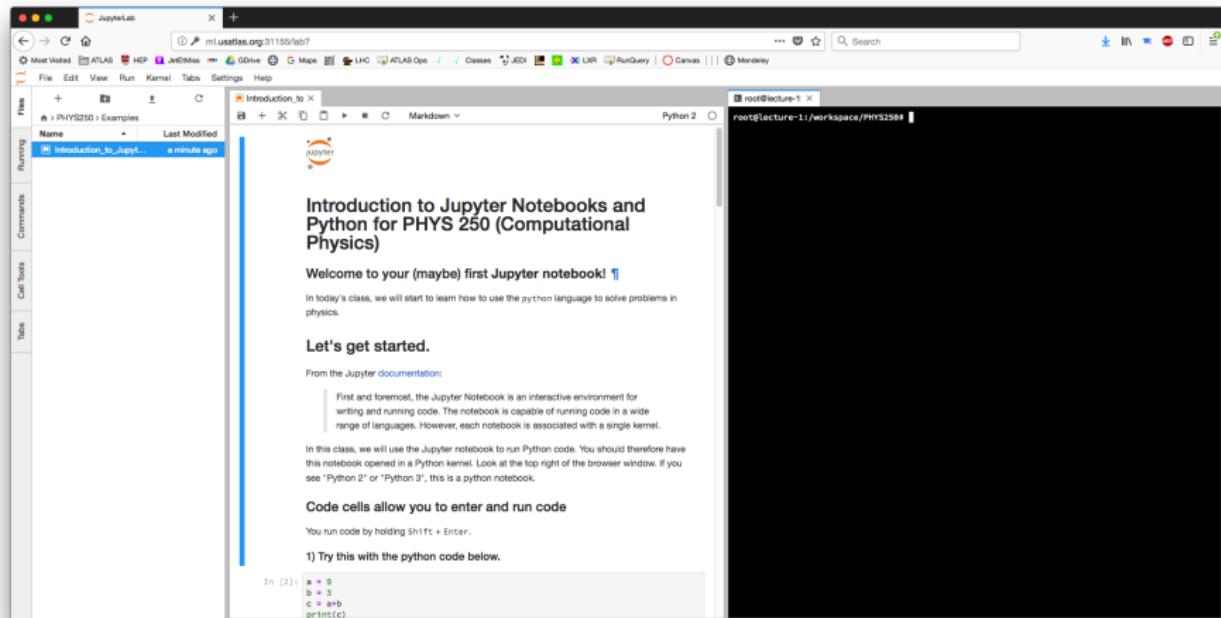
Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



Jupyter notebooks on the PHYS 250 computing platform

Built for machine learning applications running mostly in Jupyter.



Outline

1 *Introduction*

2 *Software*

3 *Physics*

Random Numbers

Random numbers may seem innocuous, but they underlie nearly everything you do electronically, rely on for security, or employ in the context of simulations and evaluation of models.

- **cryptography**
- **computer simulation**
 - the most well version of which is named after the gambling town, **Monte Carlo**
- **sampling**

As such, their use, their control, and the evaluation of just how random they are, is of paramount importance for computational physics.

And you thought lava lamps were a thing of the past



Cryptography relies on the ability to generate random numbers that are both unpredictable and secret. But **random** is a very malleable term.

True vs. Pseudo: it's more than semantics

The computers of today are, by design and by implementation, **deterministic**. That means that an **algorithm cannot**, on its own and without input from an external source, provide a stream of 100% uncorrelated random sequences. Hence the division:

- **TRNGs: True** random number generators
 - generally use some physical process that is unpredictable, but often slow and requires specialized hardware
 - possibly combined with some compensation mechanism that might remove any bias in the process
 - examples include: lava lamps, quantum processes, radioactive decays, thermal noise
- **PRNGs: Pseudo-random** number generators
 - based on algorithms and, therefore, not truly *random*
 - do not require special hardware and therefore are very portable and fast
 - can be reproduced given initial conditions

We will, perhaps obviously, focus on PRNGs.

(but if you want to build a wall of lava lamps in KPTC, I will cheer you on)

Pseudo-random number generators (PRNGs)

The idea behind an algorithmic PRNG is to generate a sequence of numbers, $x_1, x_2, x_3 \dots$ using a recurrence of the form

$$x_i = f(x_{i-1}, x_{i-2}, \dots, x_{i-n}), \quad (1)$$

where n initial numbers (“seeds”) are needed to begin the recurrence. The magic lies in the function, f , used, and the resulting *uniformity* and *correlation length* across some sequence of numbers.

Linear congruential generators (LCGs) are one of the oldest PRNGs and have the form

$$x_{i+1} = (ax_i + c) \mod m \quad (2)$$

IBM mainframes in the 60s had $a = 65539$, $c = 0$, and $m = 2^{31}$. This leads to

$$x_{i+2} = 6x_{i+1} - 9x_i \quad (3)$$

which, maybe you can tell, isn’t great.

Python random number generator: Mersenne Twister

Python comes with a built-in (`stdlib`) random number generator that can be accessed with:

```
import random as rng  
rng.random()
```

In addition, the `numpy` package provides an even more developed set of tools with `numpy.random`.

```
from numpy import random as rng  
rng.random()
```

Both of these use the Mersenne Twister algorithm, which was developed in 1997 by Matsumoto and Nishimura and is a version of a generalized feedback shift register PRNG. The name comes from the fact that the period is given by a Mersenne prime

$$M_n = 2^n - 1, n \in \mathbb{N} \quad (4)$$

(In preparing this, I found out that the largest known prime number $2^{77,232,917} - 1$ is a Mersenne prime, found on December 26, 2017.)

Testing the uniformity of this PRNG

A histogram is a graphical representation of a discrete probability distribution.
To make a simple histogram, all you need to do is:

```
from numpy import random as rng
import matplotlib.pyplot as plt
data = rng.random(100)
plt.hist(data)
```