jupyter

OpenDreamKit

Computational environments
for research and education

Min Ragan-Kelley
Simula Research Lab

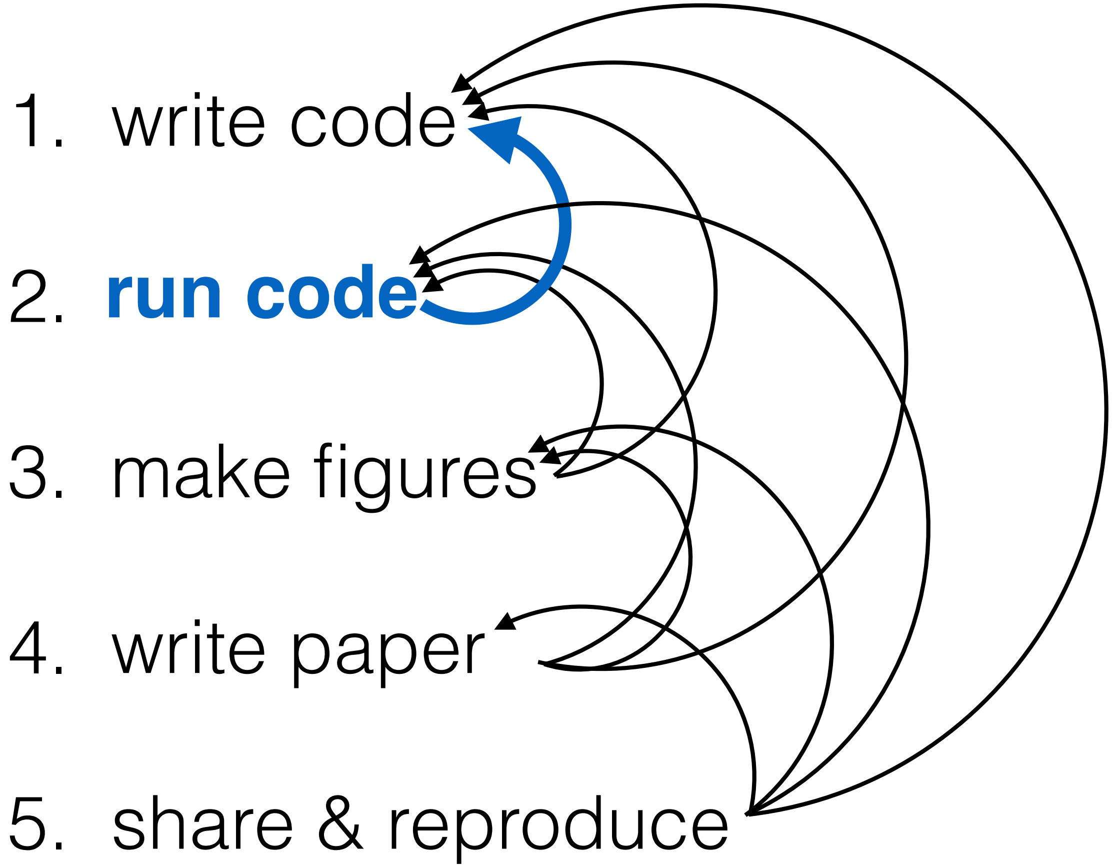
OpenDreamKit

- H2020 project
- 16 Institutions
- 4 Years (2015-2019)

Virtual Research Environments

- Generic (Jupyter, SageMath)
- Domain-specific (OOMMF micromagnetics)

How do we do computational research?



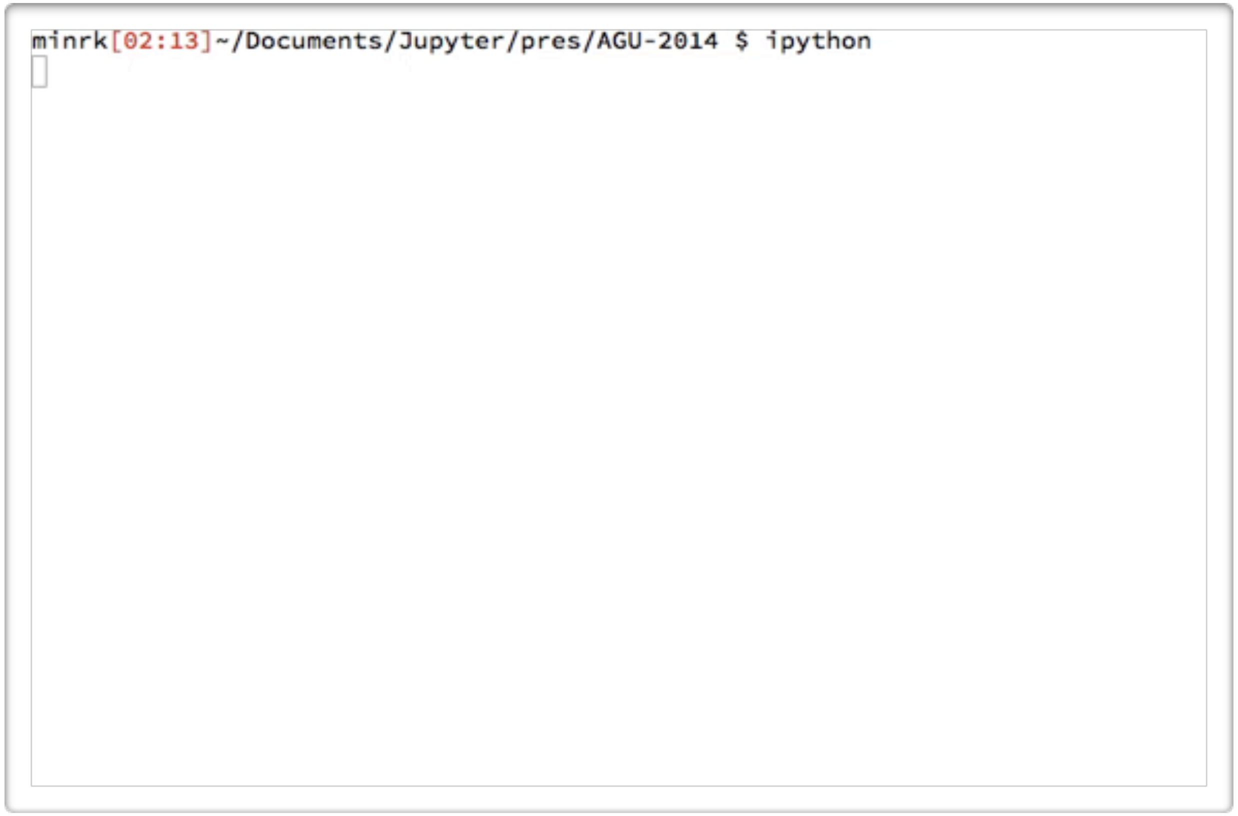
1. write code
2. **run code**
3. make figures
4. write paper
5. share & reproduce

IPython

IPython
Interactive Python

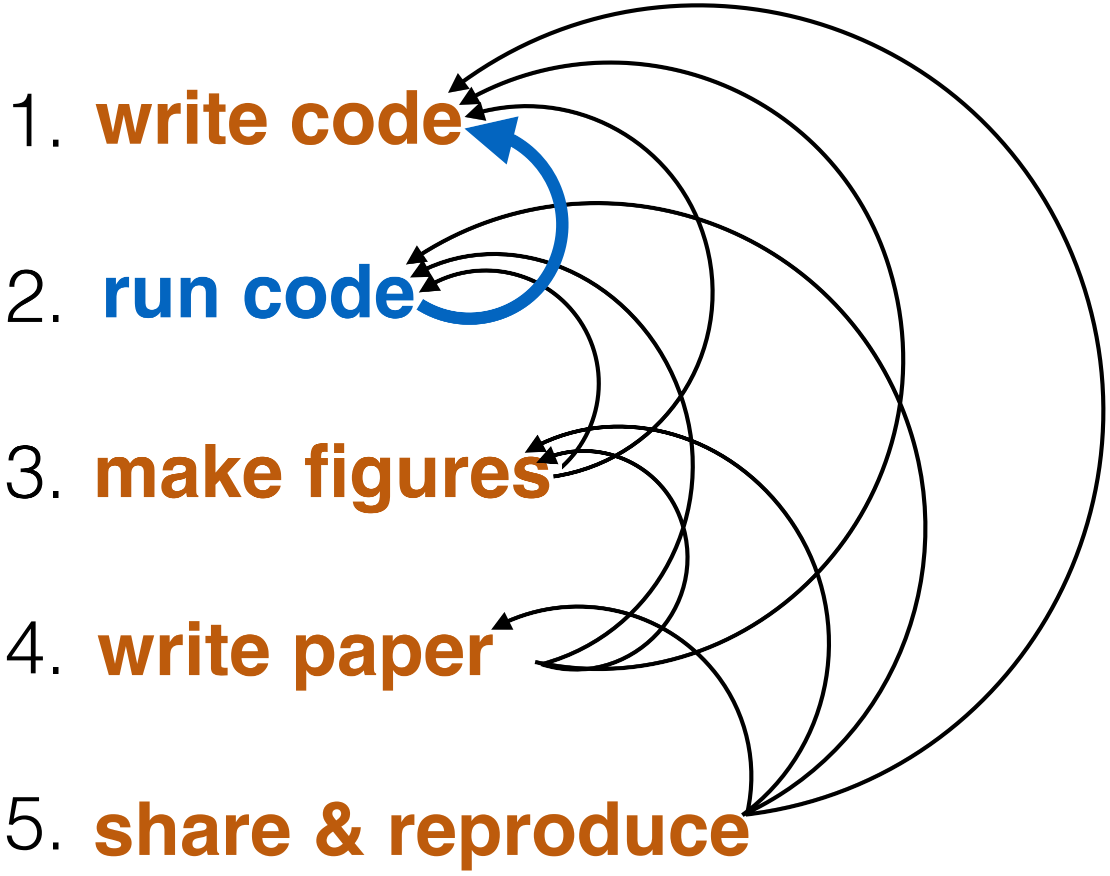
helps run code

- tab completion
- introspection
- %magics



```
minrk[02:13]~/Documents/Jupyter/pres/AGU-2014 $ ipython
```

What about Jupyter?



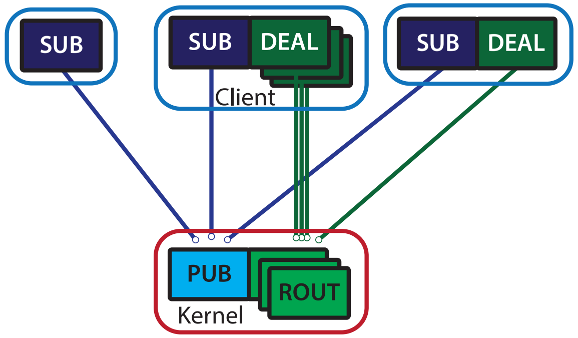
1. write code
2. run code
3. make figures
4. write paper
5. share & reproduce

IPython

jupyter

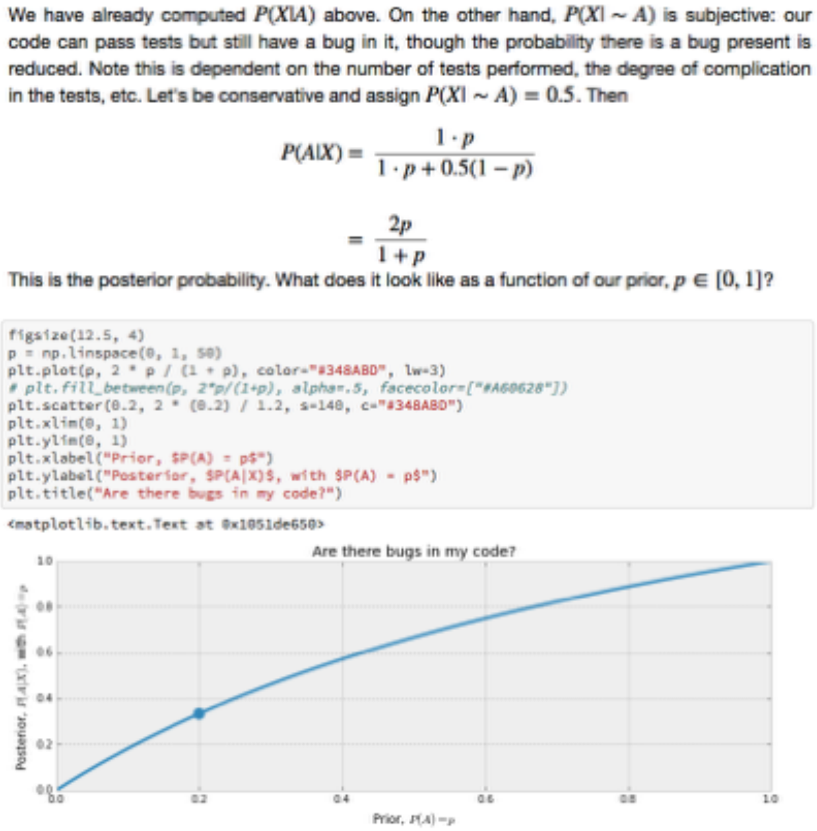
What is Jupyter?

Network Protocol



ØMQ + JSON

Document Format



We have already computed $P(X|A)$ above. On the other hand, $P(X|\sim A)$ is subjective: our code can pass tests but still have a bug in it, though the probability there is a bug present is reduced. Note this is dependent on the number of tests performed, the degree of complication in the tests, etc. Let's be conservative and assign $P(X|\sim A) = 0.5$. Then

$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1-p)}$$

$$= \frac{2p}{1+p}$$

This is the posterior probability. What does it look like as a function of our prior, $p \in [0, 1]$?

```
figsize(12.5, 4)
p = np.linspace(0, 1, 50)
plt.plot(p, 2 * p / (1 + p), color="#348ABD", lw=3)
# plt.fill_between(p, 2*p/(1+p), alpha=.5, facecolor=["#A60628"])
plt.scatter(0.2, 2 * (0.2) / 1.2, s=140, c="#348ABD")
plt.xlim(0, 1)
plt.ylim(0, 1)
plt.xlabel("Prior, $P(A) = p$")
plt.ylabel("Posterior, $P(A|X)$, with $P(A) = p$")
plt.title("Are there bugs in my code?")
```

```
<matplotlib.text.Text at 0x1051de650>
```

Are there bugs in my code?
Posterior, $P(A|X)$, with $P(A)=p$
Prior, $P(A)=p$
0.0
0.2
0.4
0.6
0.8
1.0

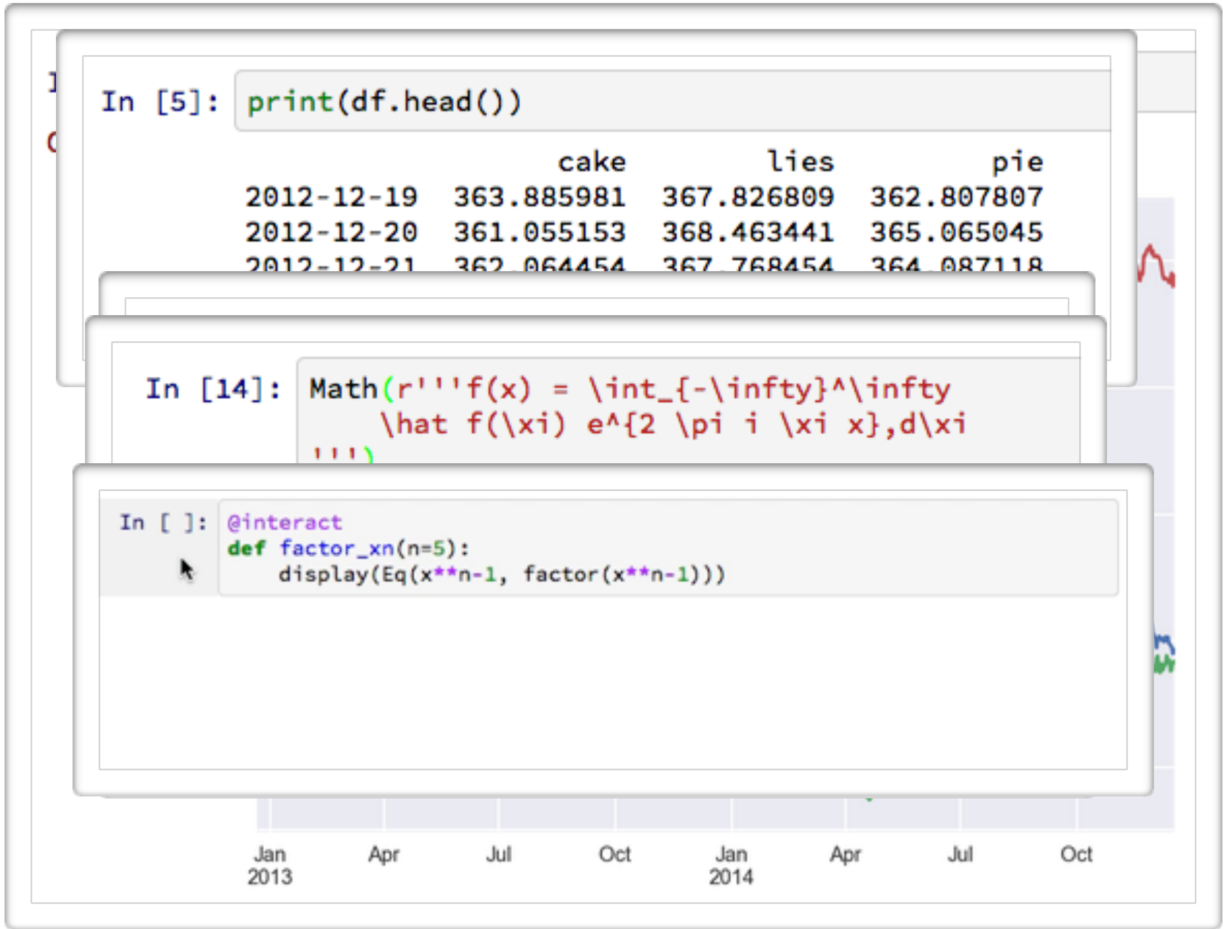
https://github.com/CamDavidsonPilon/Probabilistic-Programming-and-Bayesian-Methods-for-Hackers

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

- text
- svg, png, jpeg
- latex, pdf
- html, javascript
- interactive widgets



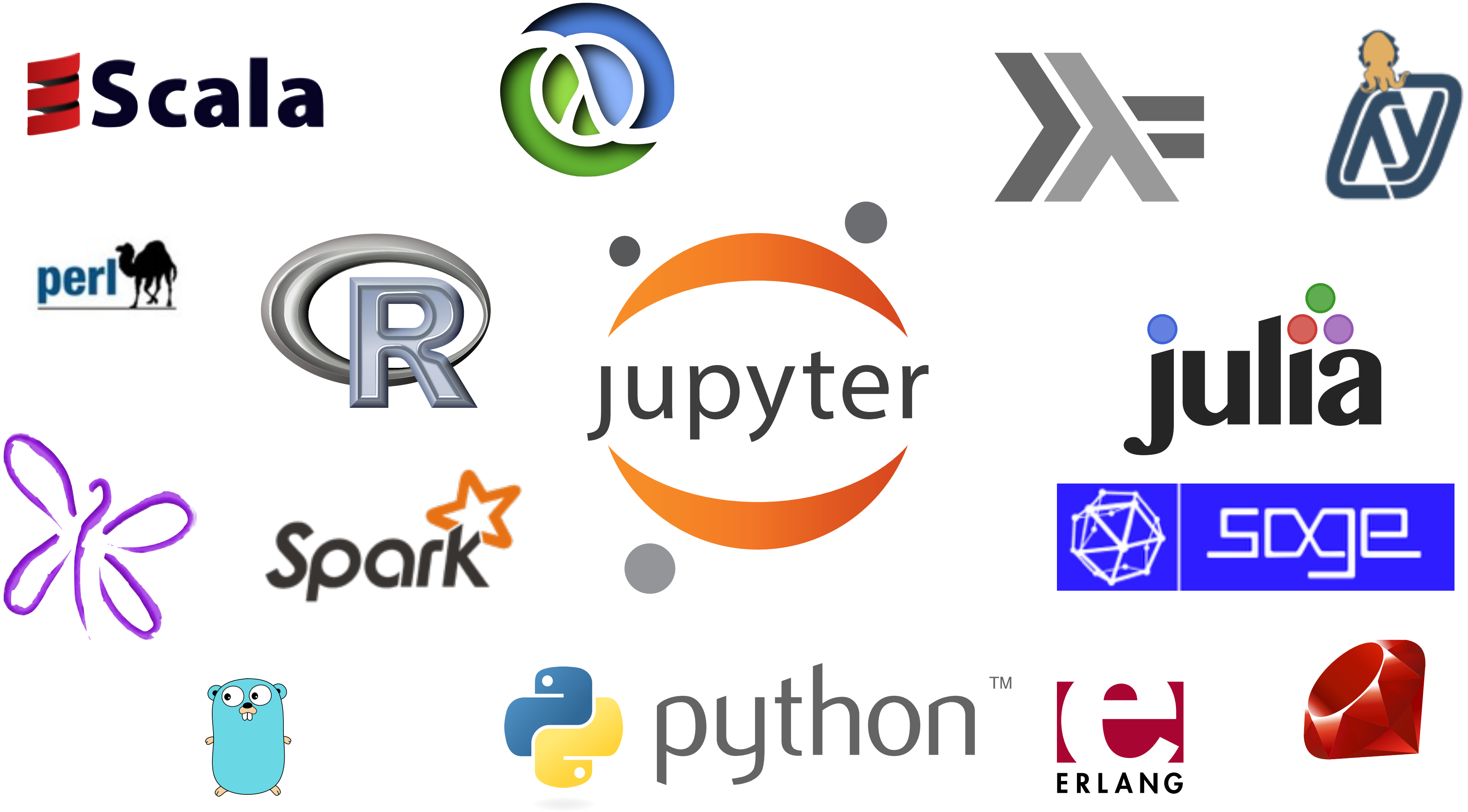
```
In [5]: print(df.head())

                  cake        lies         pie
2012-12-19  363.885981  367.826809  362.807807
2012-12-20  361.055153  368.463441  365.065045
2012-12-21  362.064454  367.768454  364.087118
```

```
In [14]: Math(r'''f(x) = \int_{-\infty}^\infty
             \hat f(\xi) e^{2 \pi i \xi x},d\xi
         ''')
```

```
In [ ]: @interact
        def factor_xn(n=5):
            display(Eq(x**n-1, factor(x**n-1)))
```

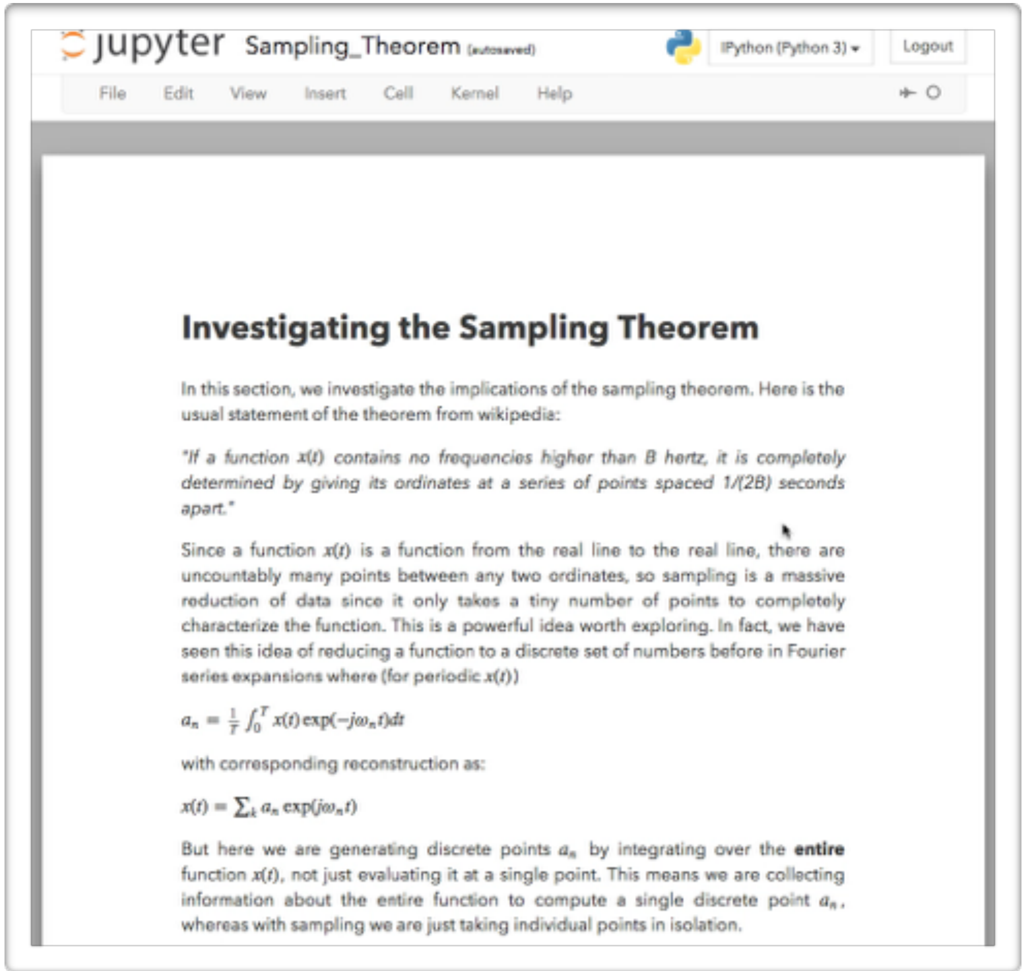
Jupyter Protocol

is language agnostic



Jupyter Notebooks

- notebook = sequence of cells
- text cell = markdown + latex)
- code cell = REP (input + output)
- metadata everywhere



Investigating the Sampling Theorem

In this section, we investigate the implications of the sampling theorem. Here is the usual statement of the theorem from wikipedia:

"If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart."

Since a function $x(t)$ is a function from the real line to the real line, there are uncountably many points between any two ordinates, so sampling is a massive reduction of data since it only takes a tiny number of points to completely characterize the function. This is a powerful idea worth exploring. In fact, we have seen this idea of reducing a function to a discrete set of numbers before in Fourier series expansions where (for periodic $x(t)$)

$a_n = \frac{1}{T} \int_0^T x(t) \exp(-j\omega_n t) dt$

with corresponding reconstruction as:

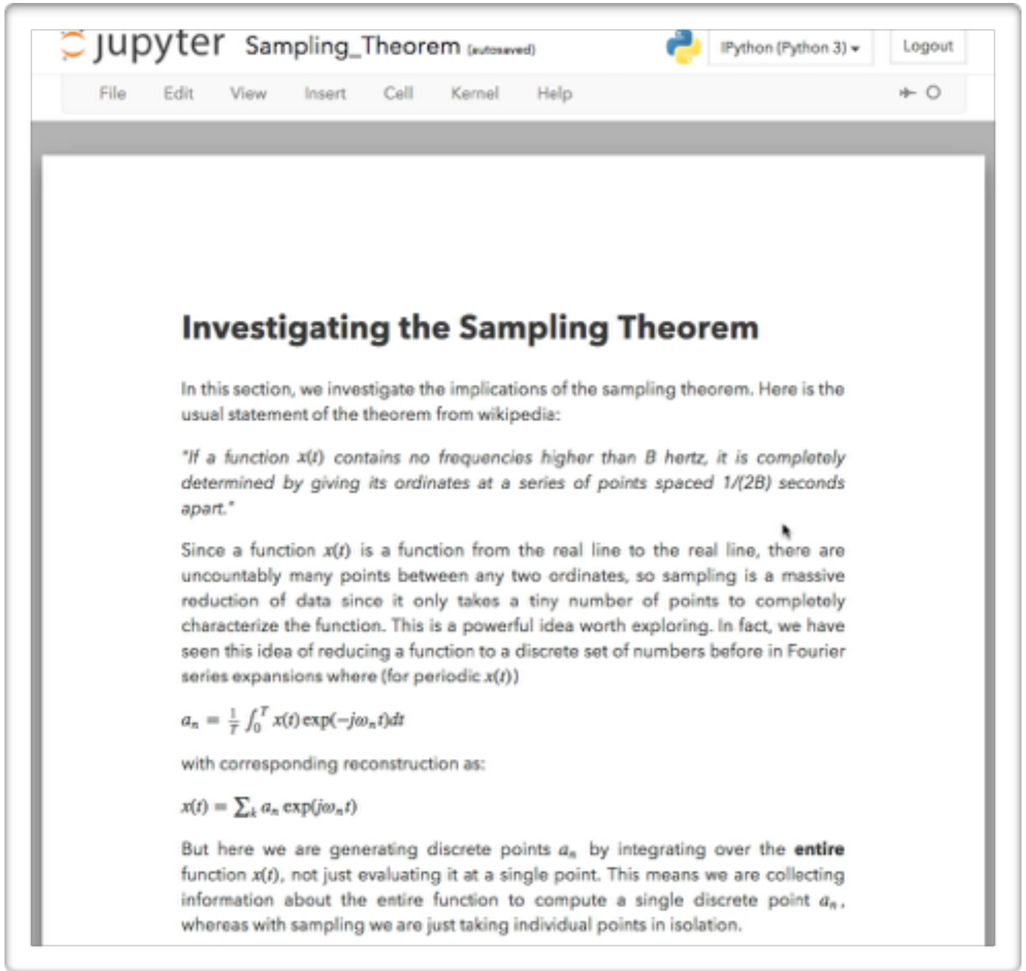
$x(t) = \sum_k a_n \exp(j\omega_n t)$

But here we are generating discrete points a_n by integrating over the **entire** function $x(t)$, not just evaluating it at a single point. This means we are collecting information about the entire function to compute a single discrete point a_n, whereas with sampling we are just taking individual points in isolation.

https://github.com/unpingco/Python-for-Signal-Processing

Jupyter Notebooks

- Plain Text (JSON)
- Publicly documented schema
- Machine readable, easy to understand
- Transformable (nbconvert)



Investigating the Sampling Theorem

In this section, we investigate the implications of the sampling theorem. Here is the usual statement of the theorem from wikipedia:

"If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart."

Since a function $x(t)$ is a function from the real line to the real line, there are uncountably many points between any two ordinates, so sampling is a massive reduction of data since it only takes a tiny number of points to completely characterize the function. This is a powerful idea worth exploring. In fact, we have seen this idea of reducing a function to a discrete set of numbers before in Fourier series expansions where (for periodic $x(t)$)

$a_n = \frac{1}{T} \int_0^T x(t) \exp(-j\omega_n t) dt$

with corresponding reconstruction as:

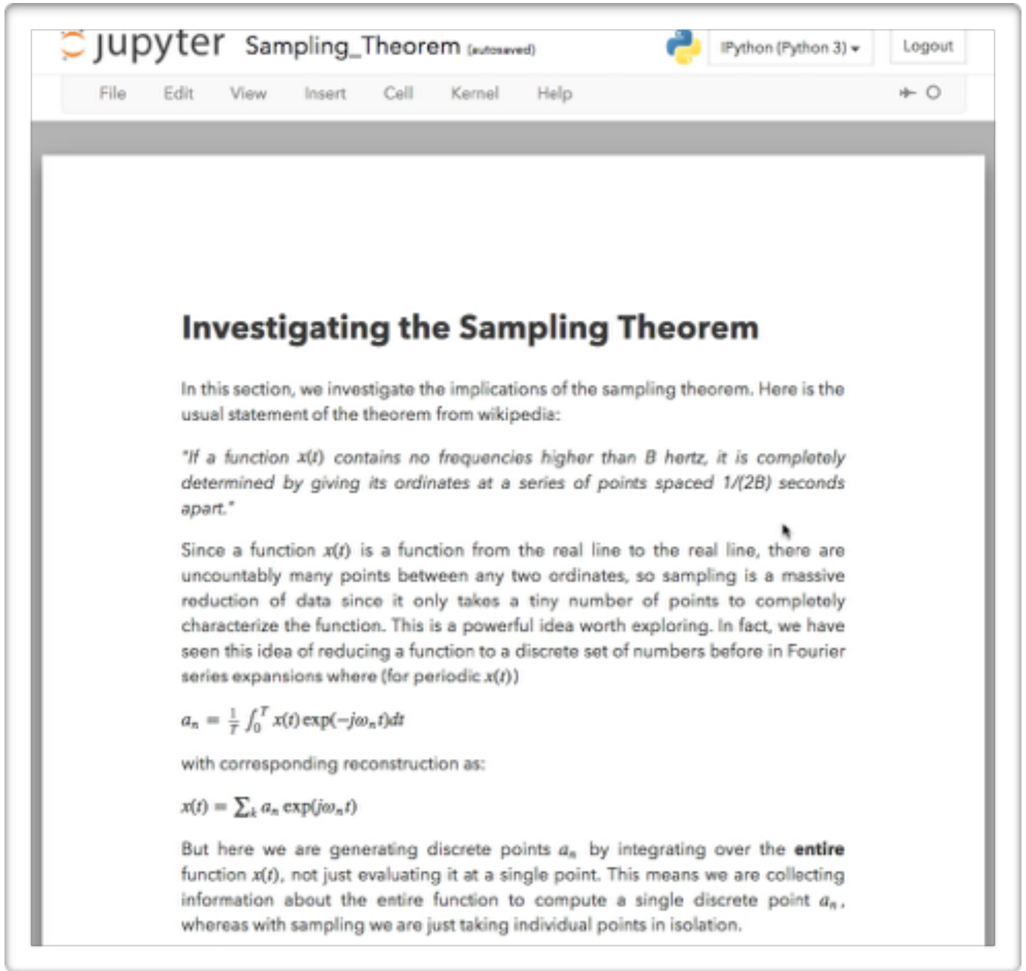
$x(t) = \sum_k a_n \exp(j\omega_n t)$

But here we are generating discrete points a_n by integrating over the **entire** function $x(t)$, not just evaluating it at a single point. This means we are collecting information about the entire function to compute a single discrete point a_n, whereas with sampling we are just taking individual points in isolation.

https://github.com/unpingco/Python-for-Signal-Processing

Jupyter Notebooks

- interactive environment
- input format
- output format



jupyter Sampling_Theorem (autosaved)
IPython (Python 3)
Logout
File Edit View Insert Cell Kernel Help

Investigating the Sampling Theorem

In this section, we investigate the implications of the sampling theorem. Here is the usual statement of the theorem from wikipedia:

"If a function $x(t)$ contains no frequencies higher than B hertz, it is completely determined by giving its ordinates at a series of points spaced 1/(2B) seconds apart."

Since a function $x(t)$ is a function from the real line to the real line, there are uncountably many points between any two ordinates, so sampling is a massive reduction of data since it only takes a tiny number of points to completely characterize the function. This is a powerful idea worth exploring. In fact, we have seen this idea of reducing a function to a discrete set of numbers before in Fourier series expansions where (for periodic $x(t)$)

$$a_n = \frac{1}{T}\int_0^T x(t)\exp(-j\omega_n t)dt$$

with corresponding reconstruction as:

$$x(t) = \sum_k a_n \exp(j\omega_n t)$$

But here we are generating discrete points a_n by integrating over the **entire** function $x(t)$, not just evaluating it at a single point. This means we are collecting information about the entire function to compute a single discrete point a_n, whereas with sampling we are just taking individual points in isolation.

https://github.com/unpingco/Python-for-Signal-Processing

Lifecycle of a Computational Idea

1. Explore an idea interactively in a Notebook
2. *Build/add to a library based on what you learn*
3. Record and collaborate on analyses in Notebooks
4. Document, demonstrate, and share in Notebooks
5. Computational companions, reproducible papers

Applications of Jupyter Notebooks

- **nbconvert** - convert notebooks to other formats (rst, html, latex/pdf, markdown, script, reveal.js slides)
- **nbviewer** - nbconvert to html on the web
- **nbgrader** - automated grading of notebooks
- **tmpnb** - containerized (docker) transient deployments of notebooks
- **thebe** - transient kernels on the web, without notebooks
- **dexy** - reproducible document-based workflows
- **jupyterhub** - multi-user notebook server for classes, groups
- **binder** - online notebooks populated from GitHub repos

jupyter

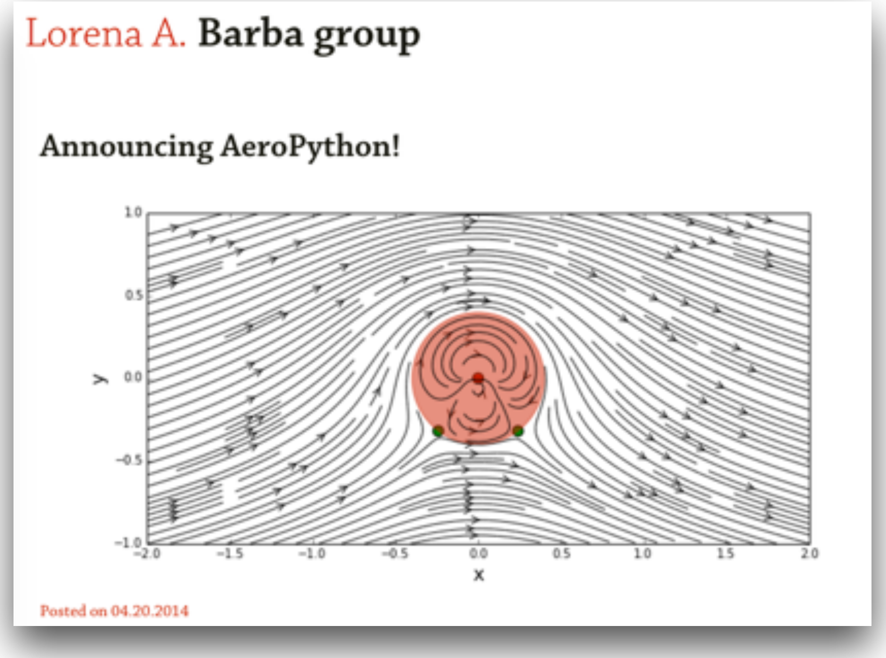
Jupyter in the Classroom

Lorena Barba

AeroPython

Aerodynamics, George Washington U

Flipped Classroom



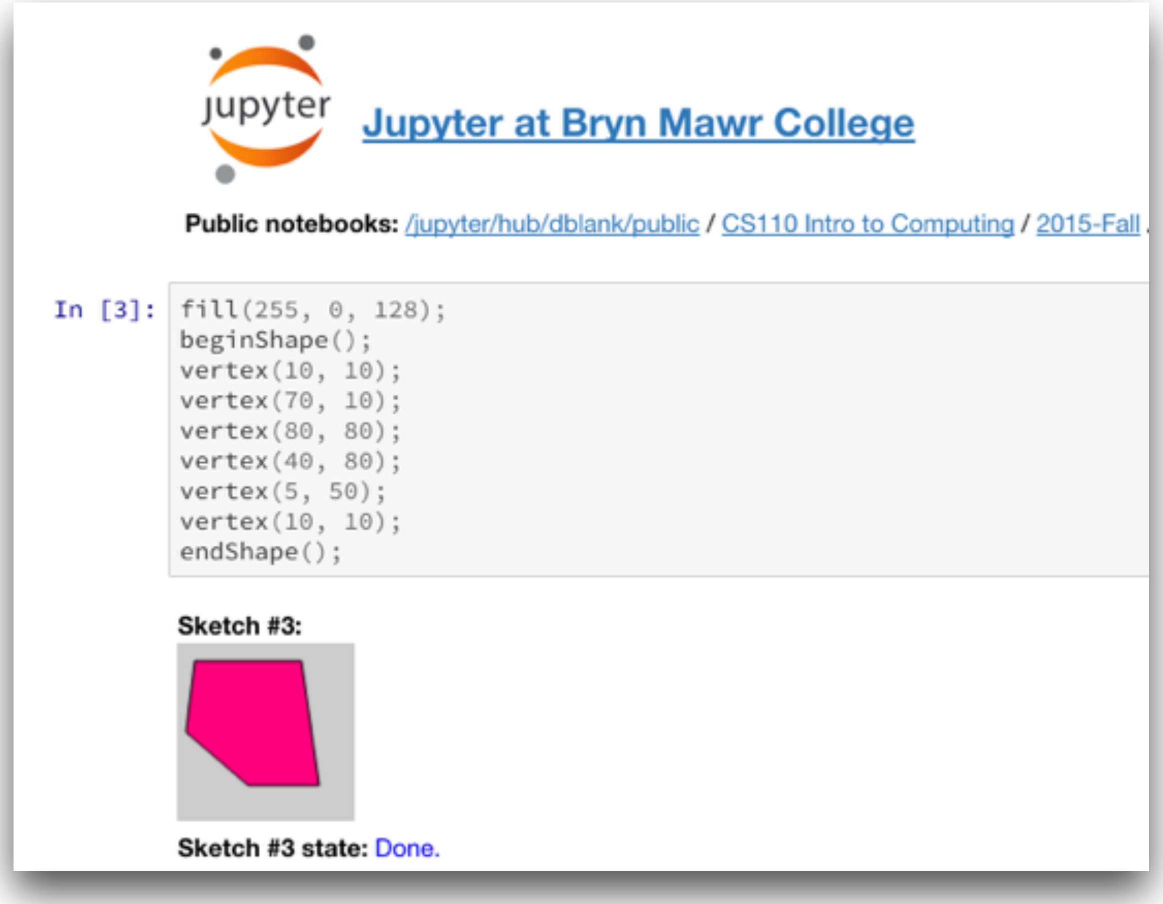
Lorena A. Barba group

Announcing AeroPython!

Posted on 04.20.2014

http://lorenabarba.com/blog/announcing-aeropython

Jupyter in the Classroom



Jupyter at Bryn Mawr College

Public notebooks: /jupyter/hub/dblank/public / CS110 Intro to Computing / 2015-Fall

```
In [3]: fill(255, 0, 128);
        beginShape();
        vertex(10, 10);
        vertex(70, 10);
        vertex(80, 80);
        vertex(40, 80);
        vertex(5, 50);
        vertex(10, 10);
        endShape();
```

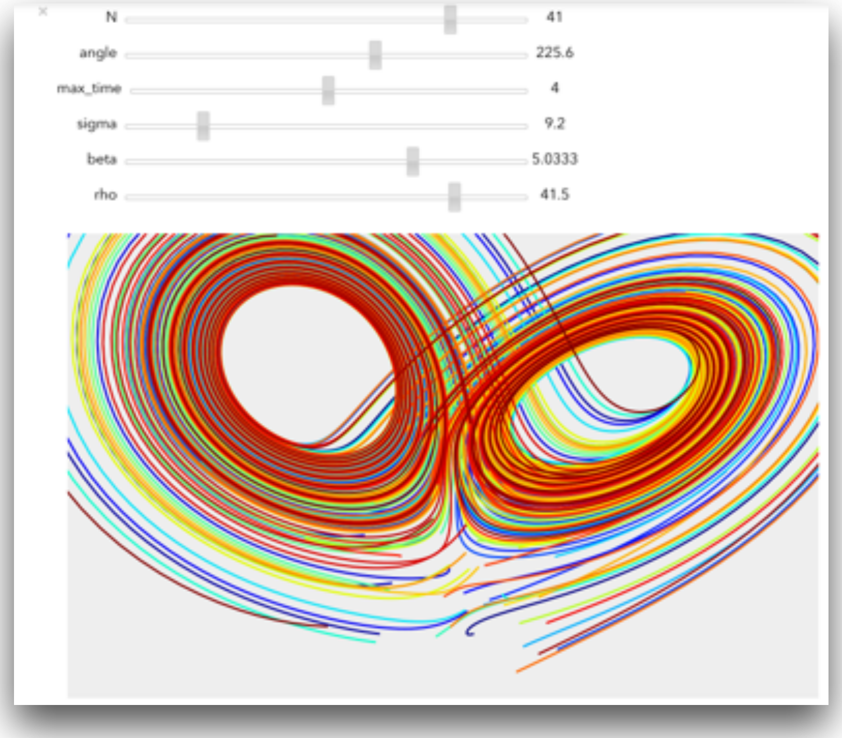
Sketch #3:

Sketch #3 state: Done.

Doug Blank
CS, Brynmawr
Calysto (Multiple languages, not Python)
JupyterHub

Jupyter in the Classroom

Brian Granger
Computational Physics, Cal Poly
JupyterHub

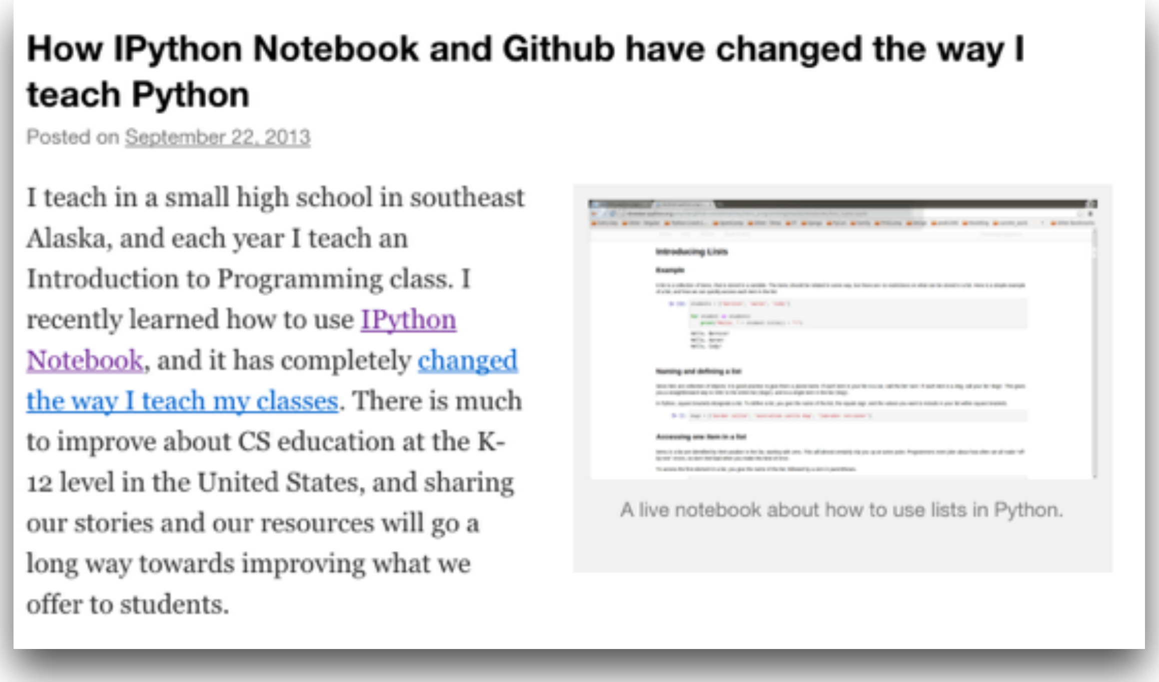


N 41
angle 225.6
max_time 4
sigma 9.2
beta 5.0333
rho 41.5

https://github.com/ellisonbg/phys202-2015

Jupyter in the Classroom

Eric Matthes
High School Programming
Alaska, USA



How IPython Notebook and Github have changed the way I teach Python

Posted on September 22, 2013

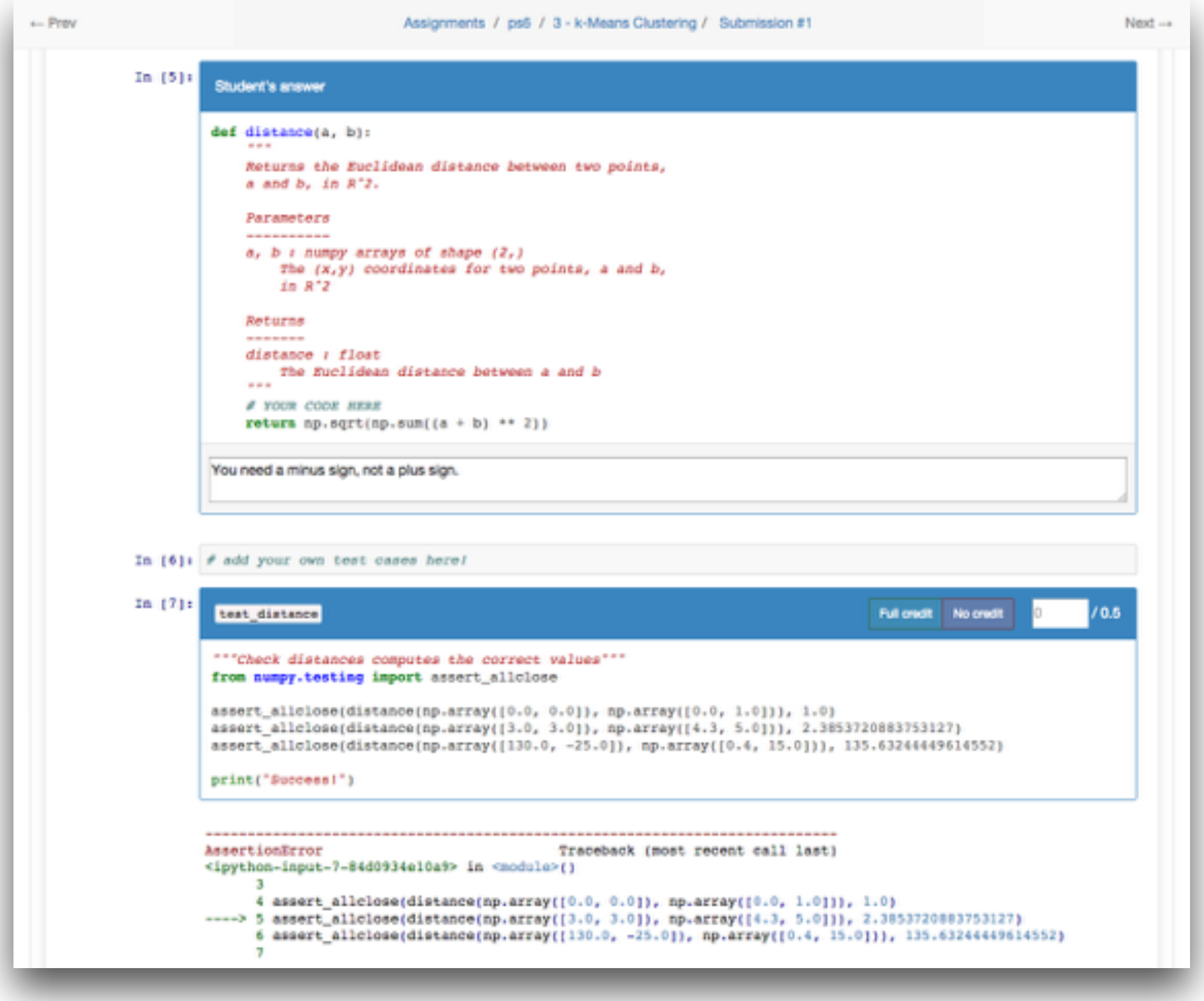
I teach in a small high school in southeast Alaska, and each year I teach an Introduction to Programming class. I recently learned how to use IPython Notebook, and it has completely changed the way I teach my classes. There is much to improve about CS education at the K-12 level in the United States, and sharing our stories and our resources will go a long way towards improving what we offer to students.

A live notebook about how to use lists in Python.

https://peak5390.wordpress.com/2013/09/22/how-ipython-notebook-and-github-have-changed-the-way-i-teach-python/

Jupyter in the Classroom

Jessica Hamrick
Computational Models of Cognition
Psychology, UC Berkeley
220 students
JupyterHub
NBGrader



```
In [5]:
def distance(a, b):
    """
    Returns the Euclidean distance between two points,
    a and b, in R^2.

    Parameters
    ----------
    a, b : numpy arrays of shape (2,)
        The (x,y) coordinates for two points, a and b,
        in R^2

    Returns
    -------
    distance : float
        The Euclidean distance between a and b
    """
    # YOUR CODE HERE
    return np.sqrt(np.sum((a + b) ** 2))
```

You need a minus sign, not a plus sign.

```
In [6]: # add your own test cases here!
```

```
In [7]: test_distance
"""Check distances computes the correct values"""
from numpy.testing import assert_allclose

assert_allclose(distance(np.array([0.0, 0.0]), np.array([0.0, 1.0])), 1.0)
assert_allclose(distance(np.array([3.0, 3.0]), np.array([4.3, 5.0])), 2.3853720883753127)
assert_allclose(distance(np.array([130.0, -25.0]), np.array([0.4, 15.0])), 135.63244449614552)

print("Success!")
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-7-84d0934e10a9> in <module>()
      3
      4 assert_allclose(distance(np.array([0.0, 0.0]), np.array([0.0, 1.0])), 1.0)
----> 5 assert_allclose(distance(np.array([3.0, 3.0]), np.array([4.3, 5.0])), 2.3853720883753127)
      6 assert_allclose(distance(np.array([130.0, -25.0]), np.array([0.4, 15.0])), 135.63244449614552)
      7
```

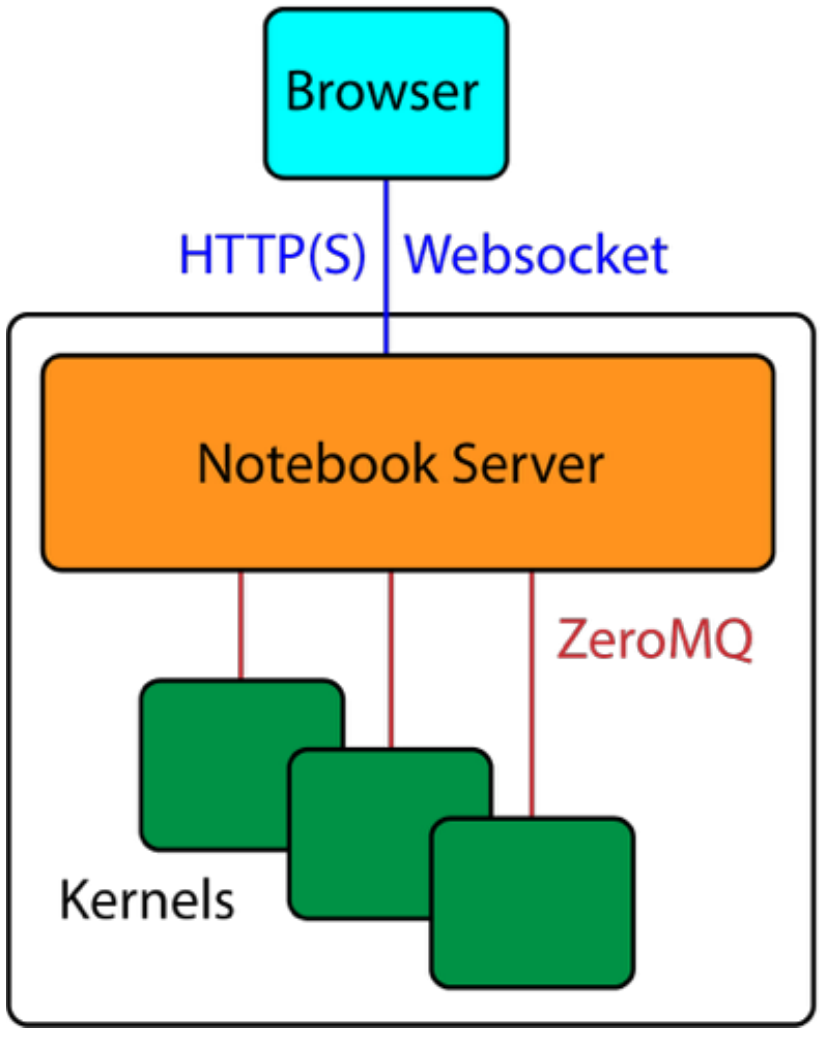
NBGrader

Notebooks as assignments

- distributed to students via JupyterHub
- turned in online
- Test-based auto-grading
- Manual grading of prose assignments

jupyterhub

multi-user notebook server



Browser

HTTP(S)

Websocket

Notebook Server

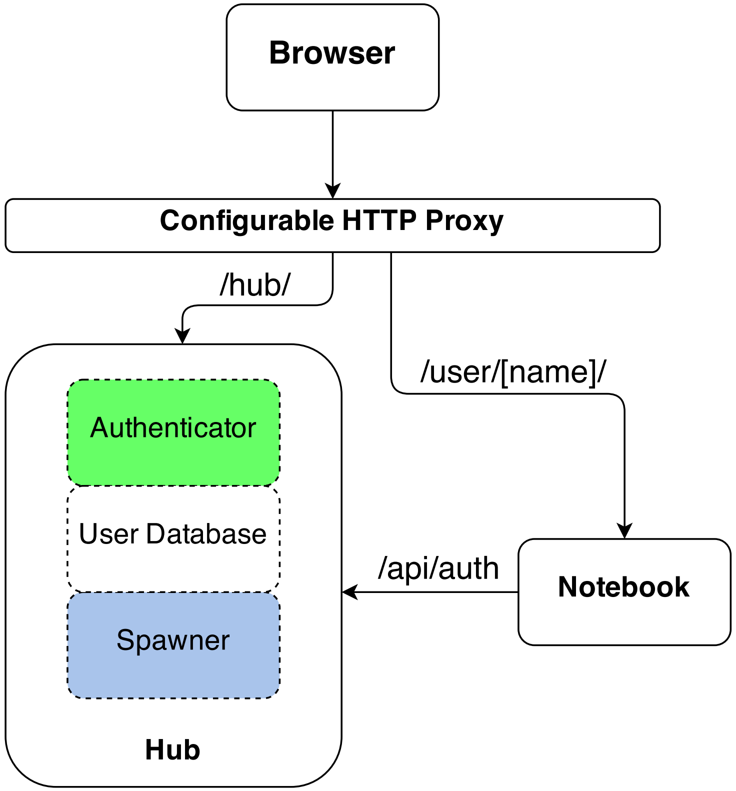
ZeroMQ

Kernels

Notebook Application

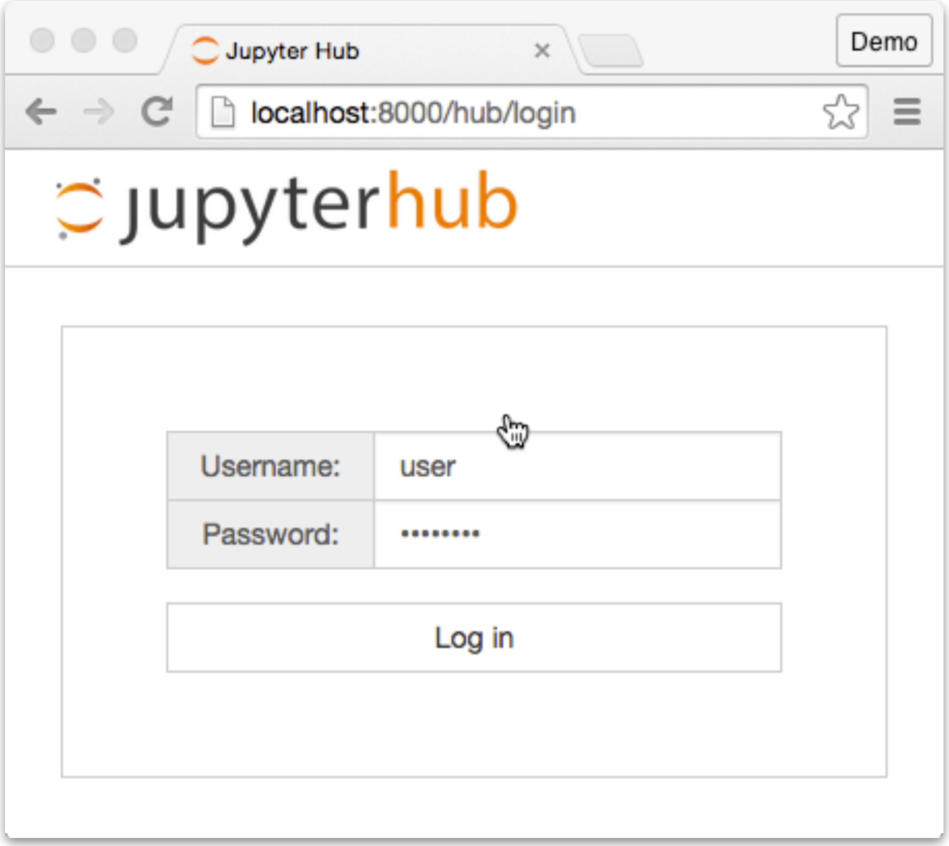
jupyterhub

- Manages authentication
- Spawns single-user servers on-demand
- Resume from db without restarting other components



jupyterhub

- Initial request is handled by Hub
- User authenticates via form / OAuth
- Spawner starts single-user server
- Hub notifies Proxy
- Redirects user to `/user/[name]`
- Single-user Server verifies auth with Hub



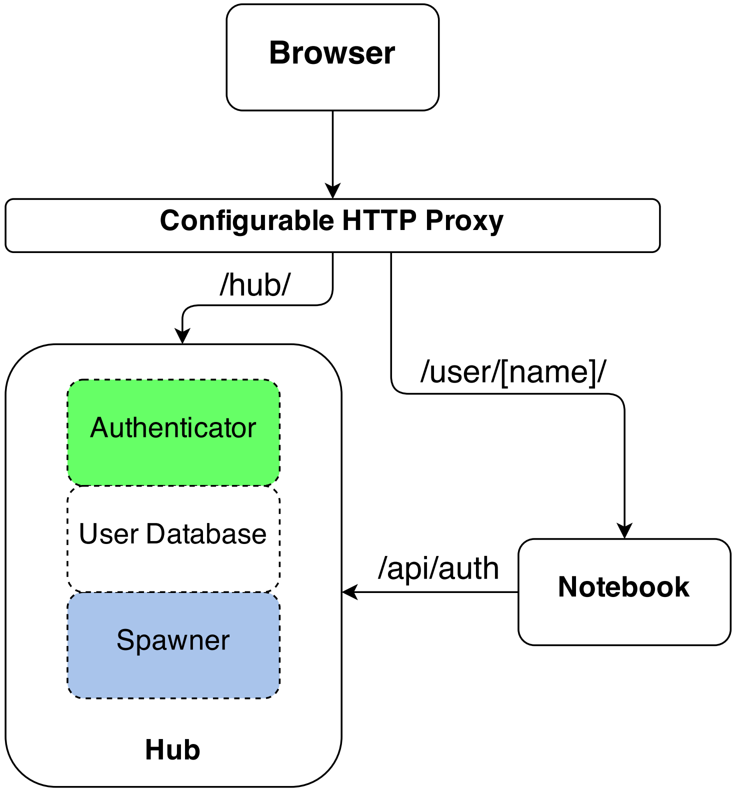
localhost:8000/hub/login

Username: user

Password: •••••••••

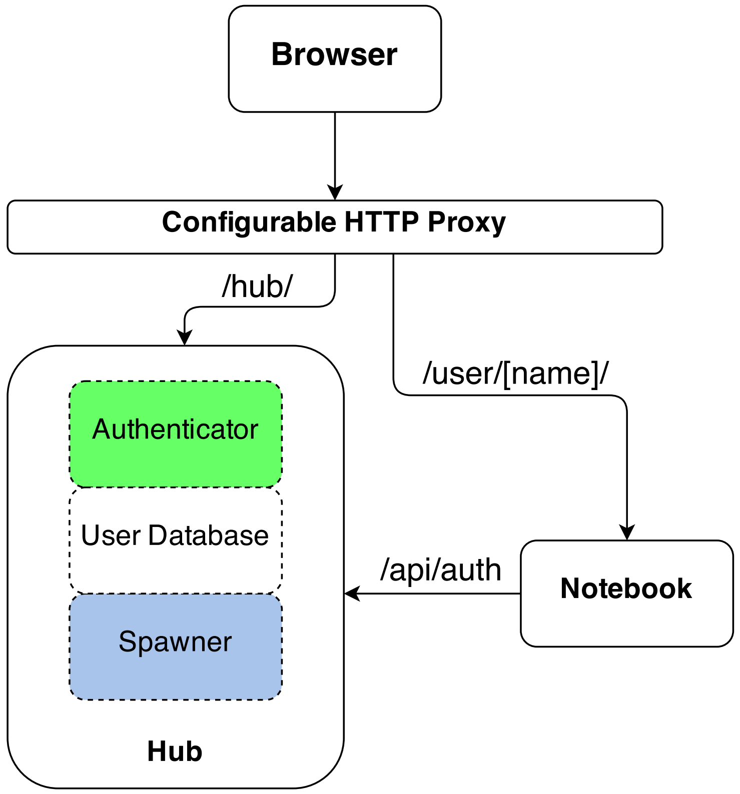
jupyterhub

- Pluggable Authentication
  - PAM, OAuth, SSO, etc.
- Pluggable Spawning
  - Popen, Docker, PBS, EC2, etc.



jupyterhub

- User-controls
  - specify resources √
  - multiple servers per user
- Sharing
  - publishing notebooks for other users
  - long-term: live collaboration



Browser

Configurable HTTP Proxy

/hub/

/user/[name]/

Authenticator

User Database

Spawner

Hub

/api/auth

Notebook

Disce Doce
RERVM COGNOSCERE CAVSAS

The
University
Of
Sheffield.

CERN

BRYN
MAWR
COLLEGE

jupyterhub

THE UNIVERSITY OF CALIFORNIA · BERKELEY · 1868
LET THERE BE LIGHT

WIKIMEDIA
FOUNDATION

NERSC

XSEDE
Extreme Science and Engineering
Discovery Environment

BROOKHAVEN
NATIONAL LABORATORY

jupyter