Jupyter and IPython

facilitating open access
and reproducible research

Min Ragan-Kelley, Simula Research Lab

How do we do computational research?

How do we do computational research?

1. write code

How do we do computational research?

1. write code

2. run code

How do we do computational research?

1. write code

2. run code

3. make figures

How do we do computational research?

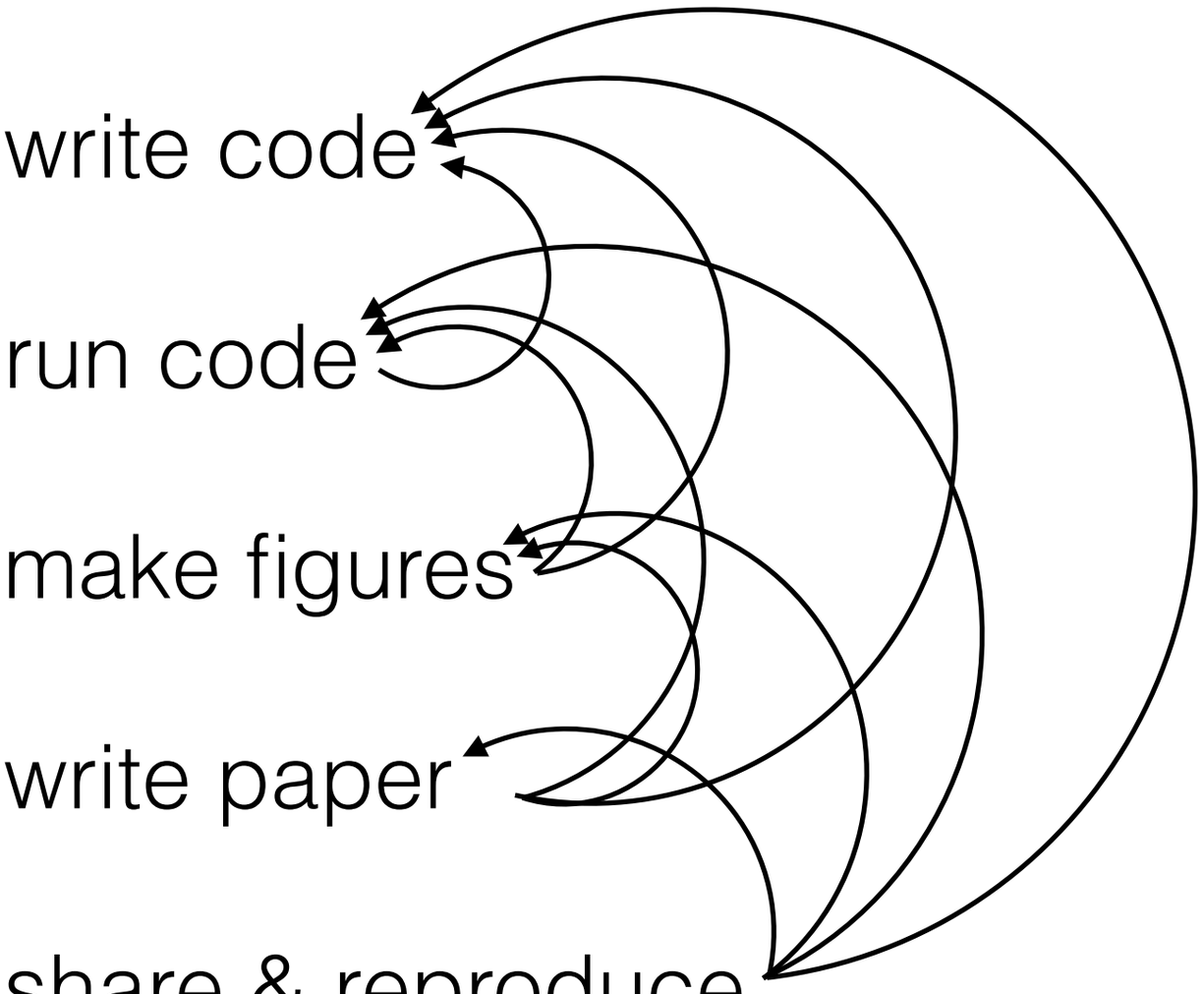
1. write code
2. run code
3. make figures
4. write paper

How do we do computational research?

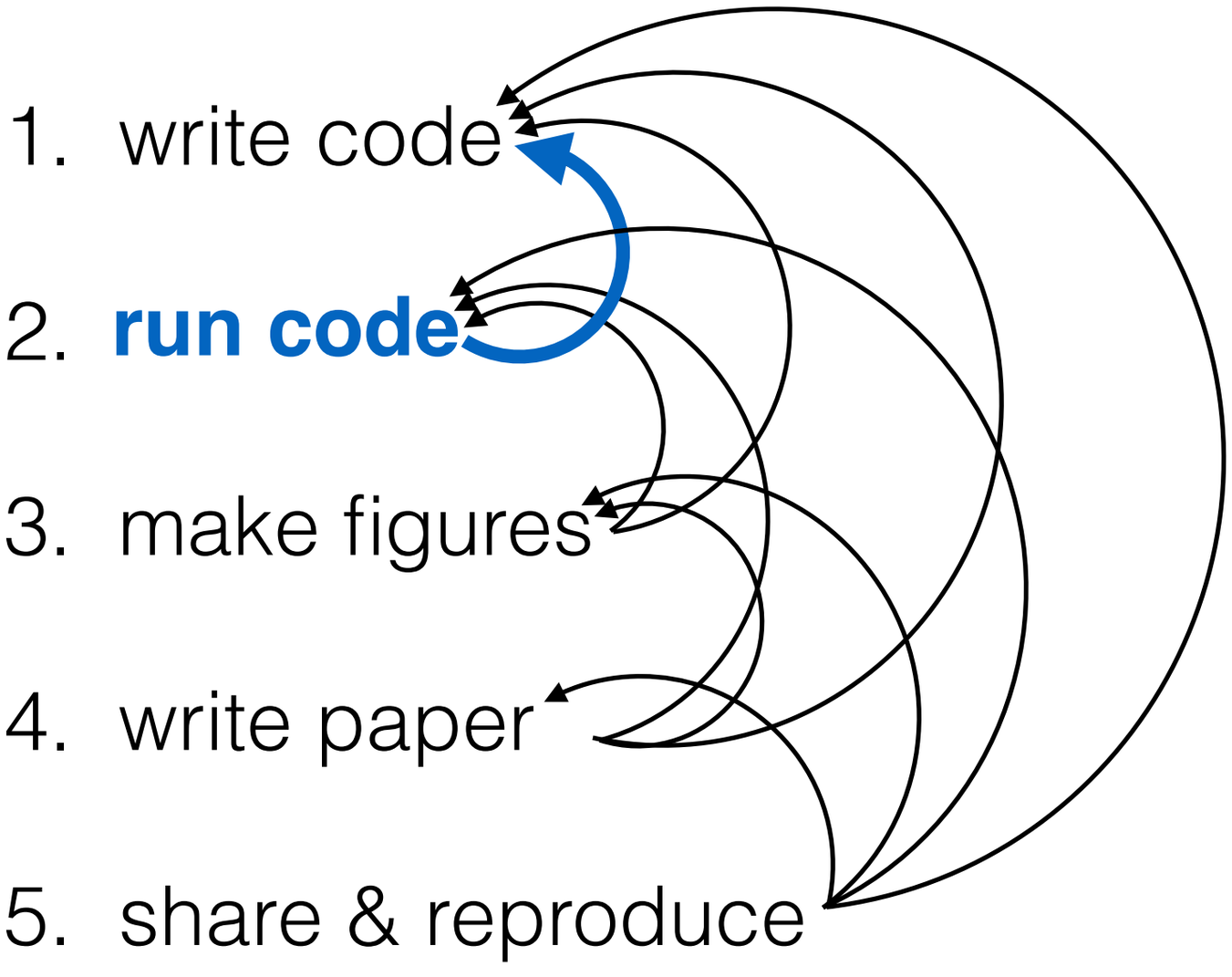
1. write code
2. run code
3. make figures
4. write paper
5. share & reproduce

How do we do computational research?

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



How do we do computational research?



IPython

IPython
Interactive Python

helps run code

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

IPython
Interactive Python

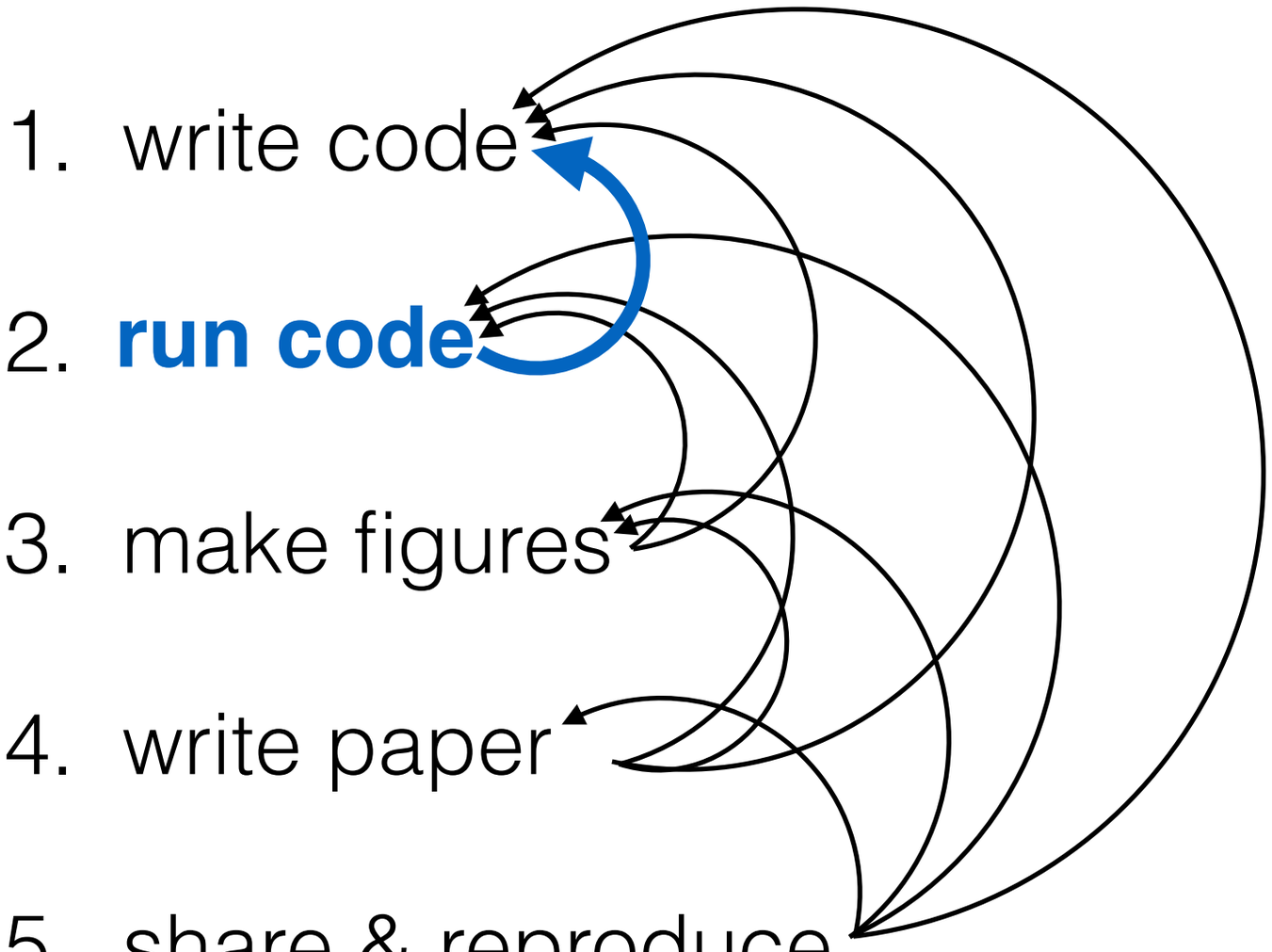
helps run code

- tab completion
- introspection
- %magics

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

What about Jupyter?

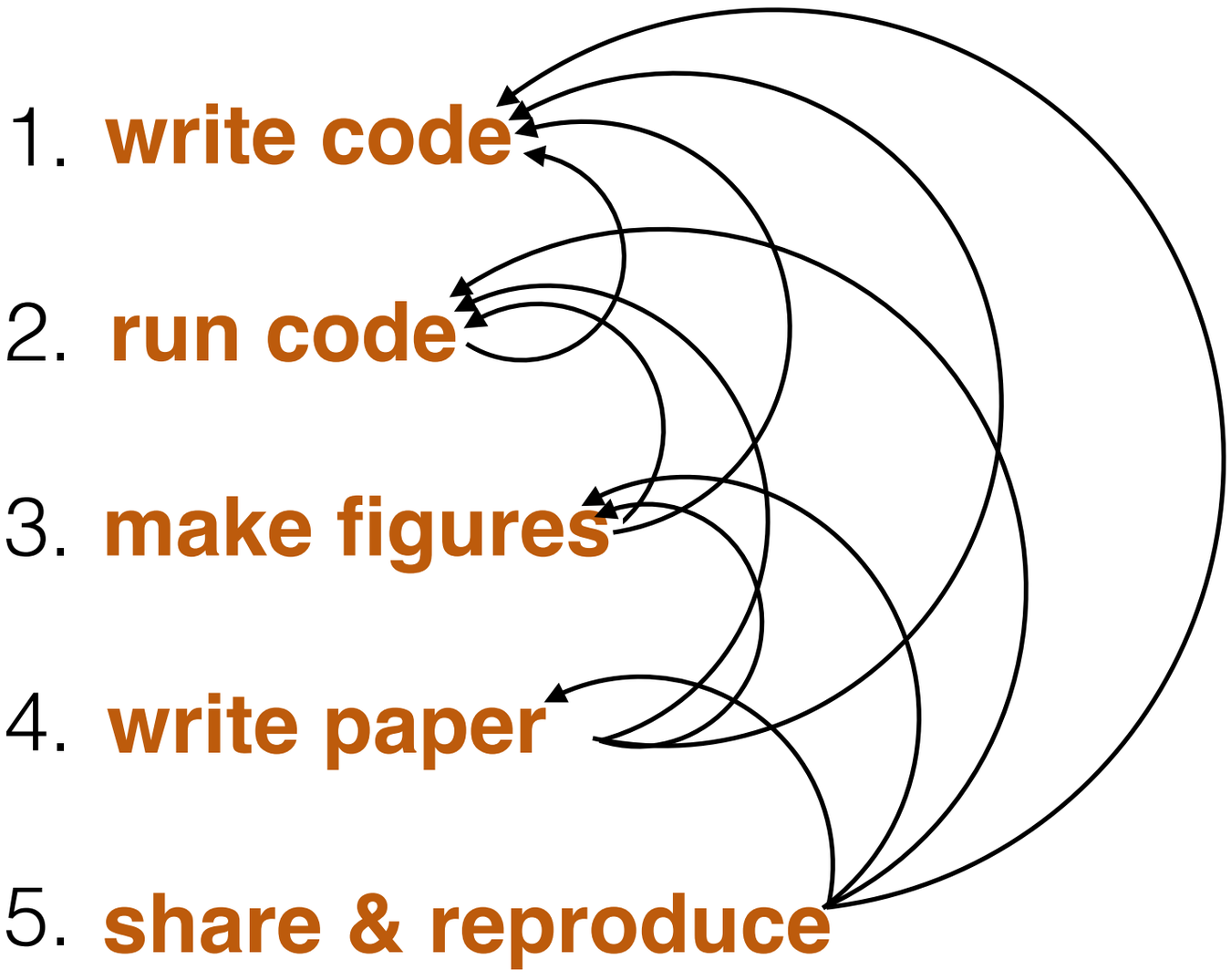
1. write code
2. **run code**
3. make figures
4. write paper
5. share & reproduce



IPython

What about Jupyter?

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

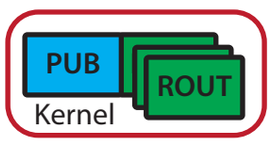


What is Jupyter?

Rich REPL Protocol

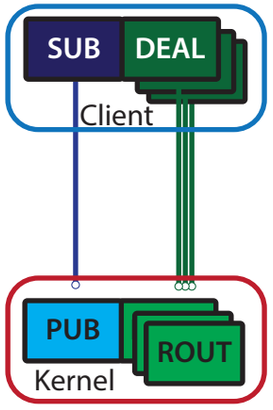
What is Jupyter?

Rich REPL Protocol



What is Jupyter?

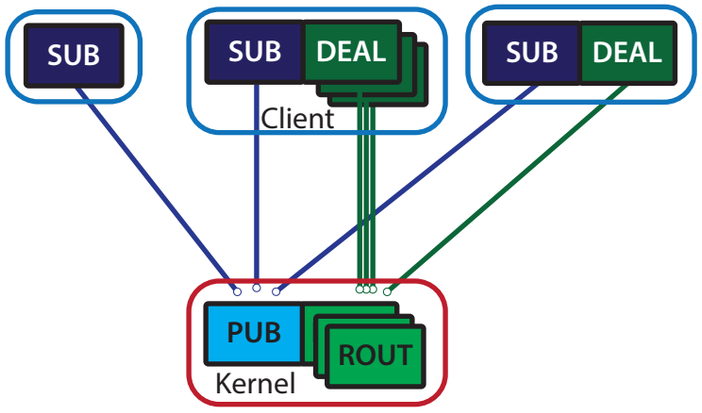
Rich REPL Protocol



ØMQ + JSON

What is Jupyter?

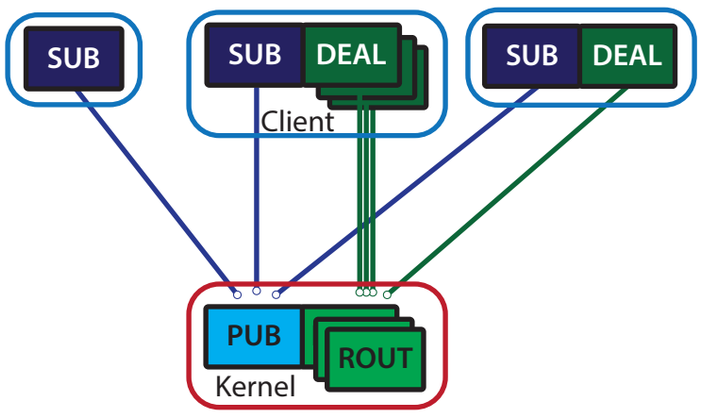
Rich REPL Protocol



ØMQ + JSON

What is Jupyter?

Rich REPL Protocol

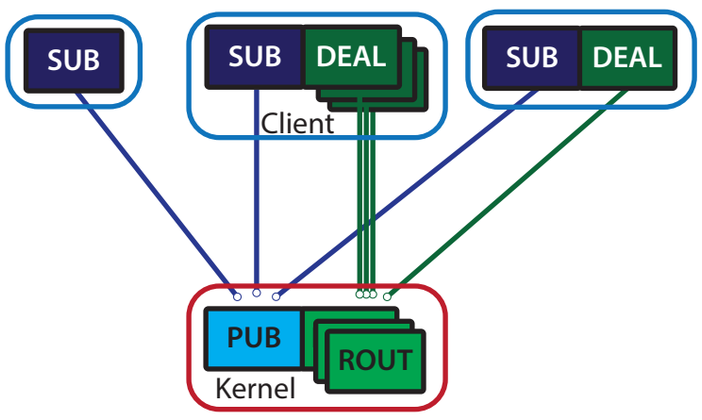


ØMQ + JSON

Document Format

What is Jupyter?

Rich REPL 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

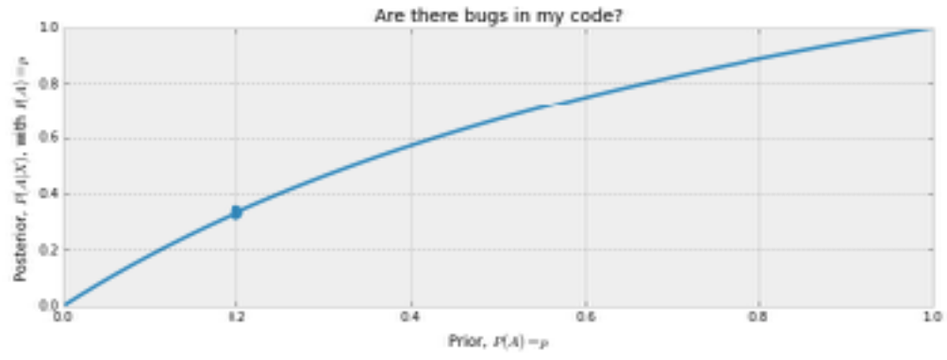
$$P(A|X) = \frac{1 \cdot p}{1 \cdot p + 0.5(1-p)}$$

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

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>
```



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

Jupyter Protocol

REP*L over JSON + ØMQ

Jupyter Protocol

REP*L over JSON + ØMQ

- Read

```
msg_type = 'execute_request'
content = {
    'code' : """
        import pandas as pd
        df = pd.read_csv('mydata.csv')
        """,
    ...
}
```

Jupyter Protocol

REP*L over JSON + ØMQ

- Read
- Eval

```
msg_type = 'execute_reply'
content = {
    'execution_count': 3,
    'status': 'ok',
    ...
}
```

Jupyter Protocol
REP*L over JSON + ØMQ

- Read
- Eval
- Print*

```
msg_type = 'display_data'
content = {
    'data': {
        "text/plain": "<MyDataFrame at 0x...>",
        "text/html": "<table>...</table>",
        ...
    },
    'metadata': {},
    ...
}
```

Jupyter Protocol

REP*L over JSON + ØMQ

- Read
- Eval
- Print*
- Loop

```
msg_type = 'display_data'
content = {
    'data': {
        "text/plain": "<MyDataFrame at 0x...>",
        "text/html": "<table>...</table>",
        ...
    },
    'metadata': {},
    ...
}
```

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

Jupyter Protocol
supercharge the P in REP*L

any mime-type output

- text

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

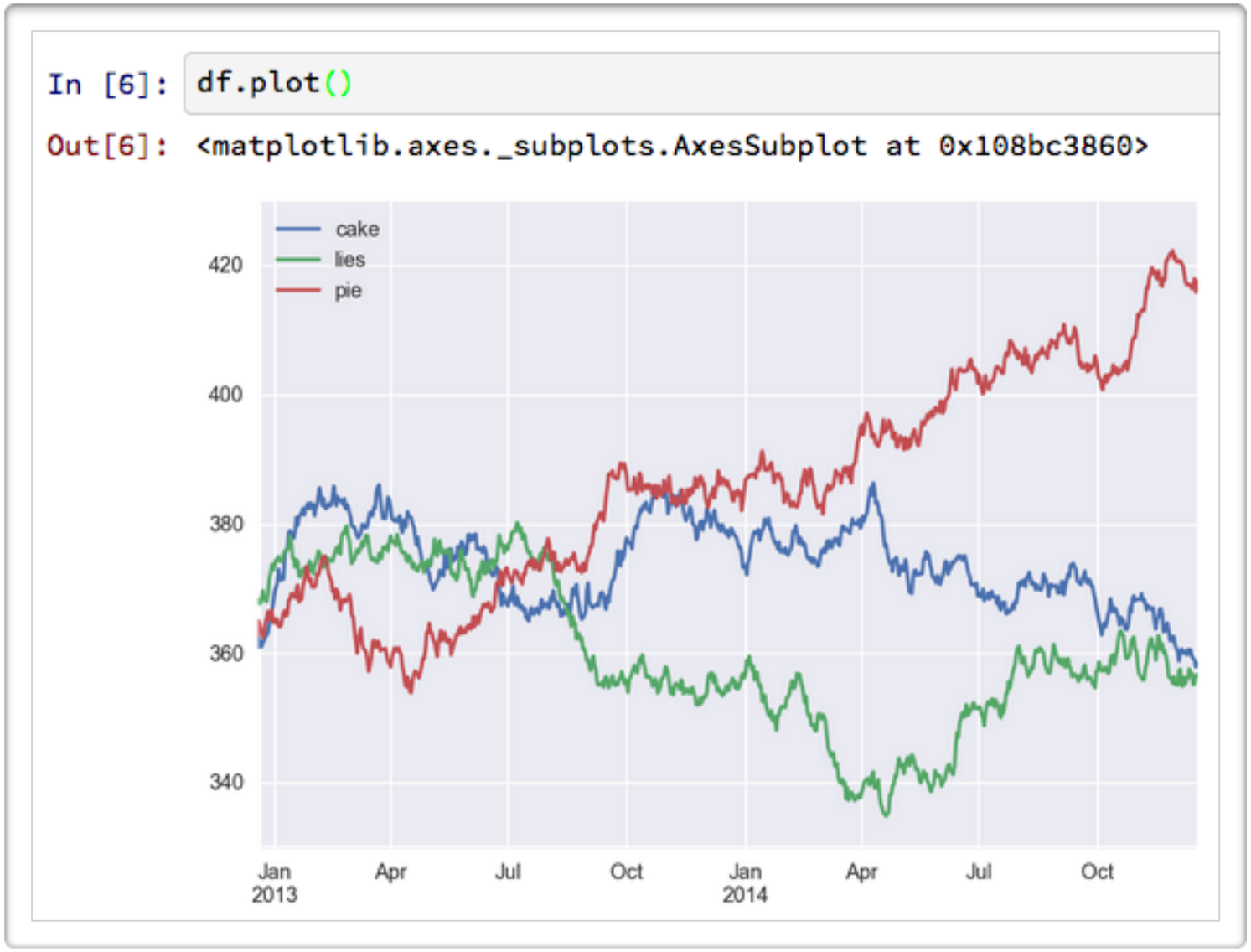
```
                 cake        lies         pie
2012-12-19  363.885981  367.826809  362.807807
2012-12-20  361.055153  368.463441  365.065045
2012-12-21  362.064454  367.768454  364.087118
2012-12-22  361.110406  368.457023  363.762849
2012-12-23  361.890903  369.800517  362.596256
```

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

- text
- svg, png, jpeg



Jupyter Protocol

supercharge the P in REP*L

any mime-type output

- text
- svg, png, jpeg
- latex, pdf

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

Out[14]:

$$f(x) = \int_{-\infty}^{\infty} \hat{f}(\xi) e^{2\pi i \xi x}, d\xi$$

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

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

In [16]: `df.tail()`

Out[16]:

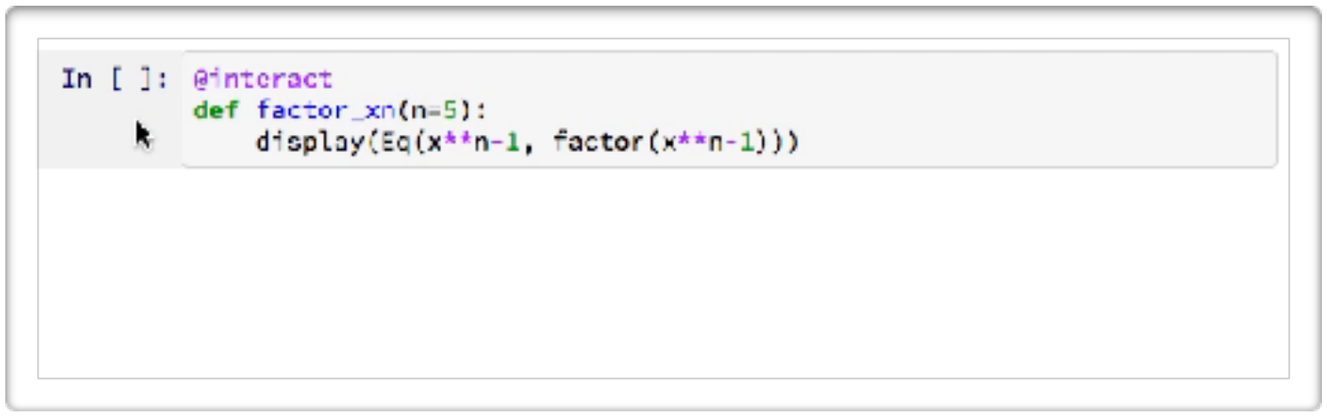
	cake	lies	pie
2014-12-14	400.537295	387.213920	371.035670
2014-12-15	402.107164	386.883925	370.902248
2014-12-16	402.548479	386.407872	369.037149
2014-12-17	403.010896	387.532590	369.017640
2014-12-18	403.191969	388.824959	369.630229

Jupyter Protocol

supercharge the P in REP*L

any mime-type output

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



Jupyter Protocol

is language agnostic

Jupyter Protocol

is language agnostic

Jupyter Protocol

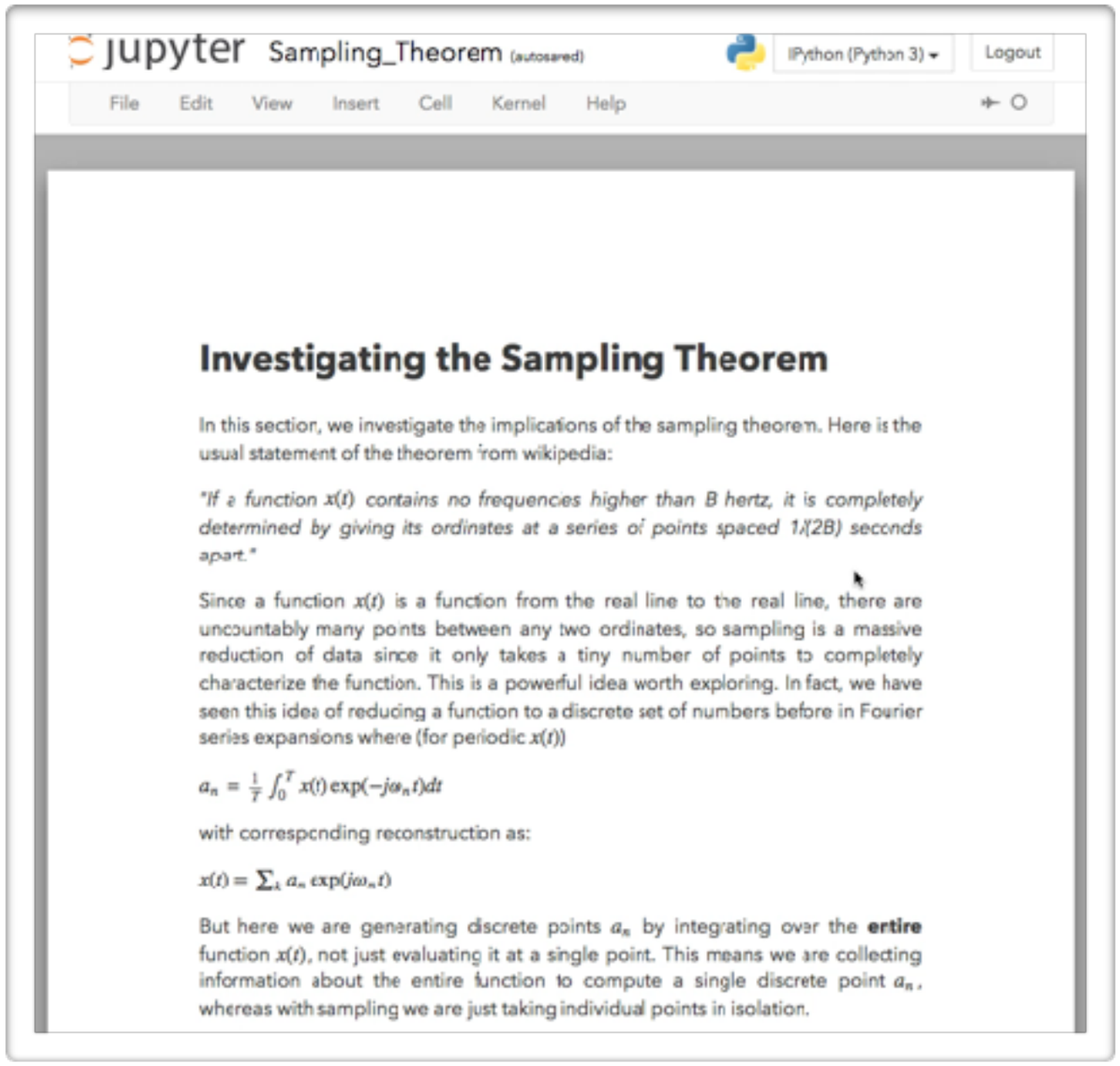
is language agnostic

Jupyter Protocol

is language agnostic

Jupyter Notebooks

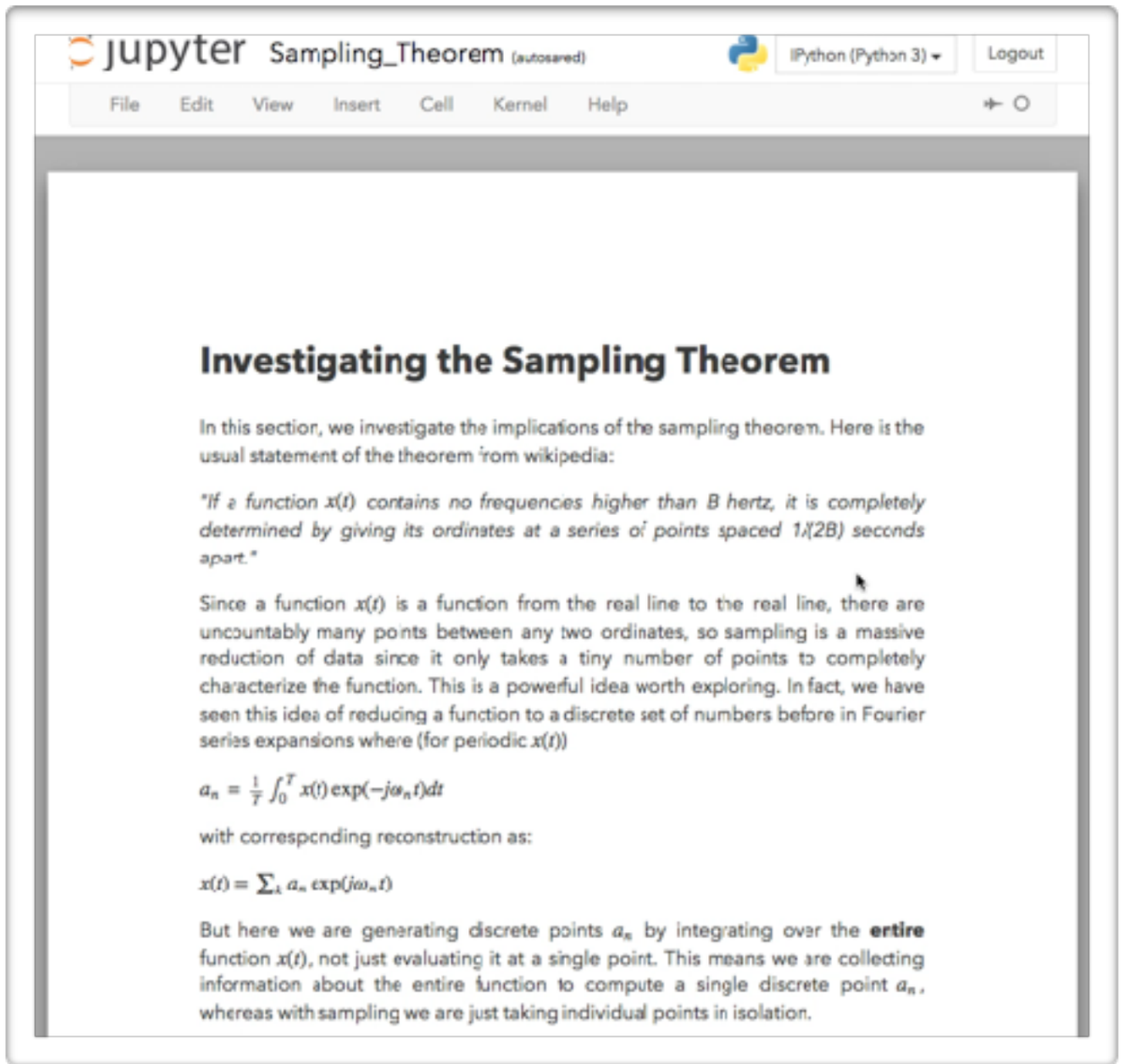
- notebook = sequence of cells



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

Jupyter Notebooks

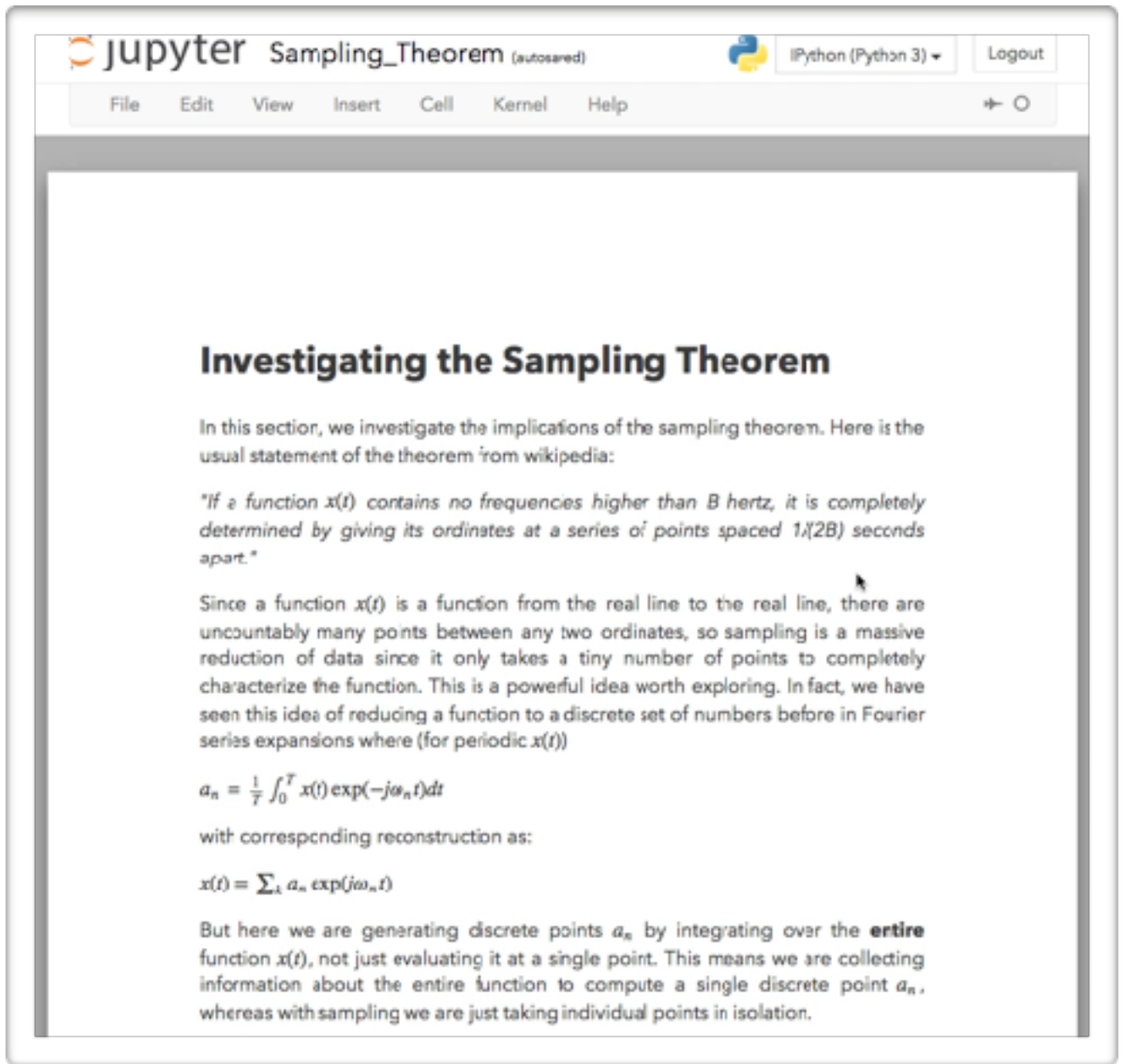
- notebook = sequence of cells
- text cell = markdown + latex)



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

Jupyter Notebooks

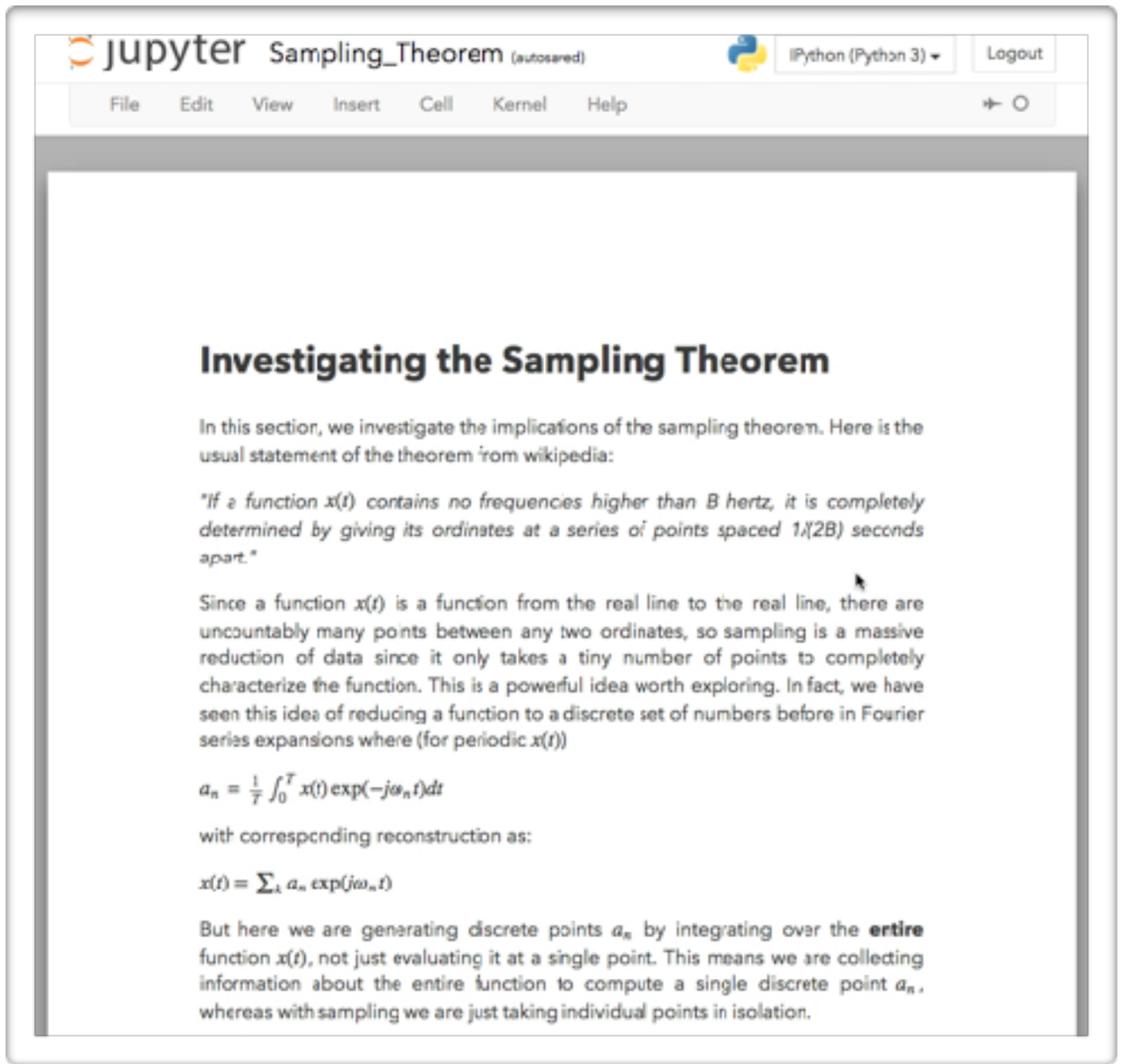
- notebook = sequence of cells
- text cell = markdown + latex)
- code cell = REP (input + output)



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

Jupyter Notebooks

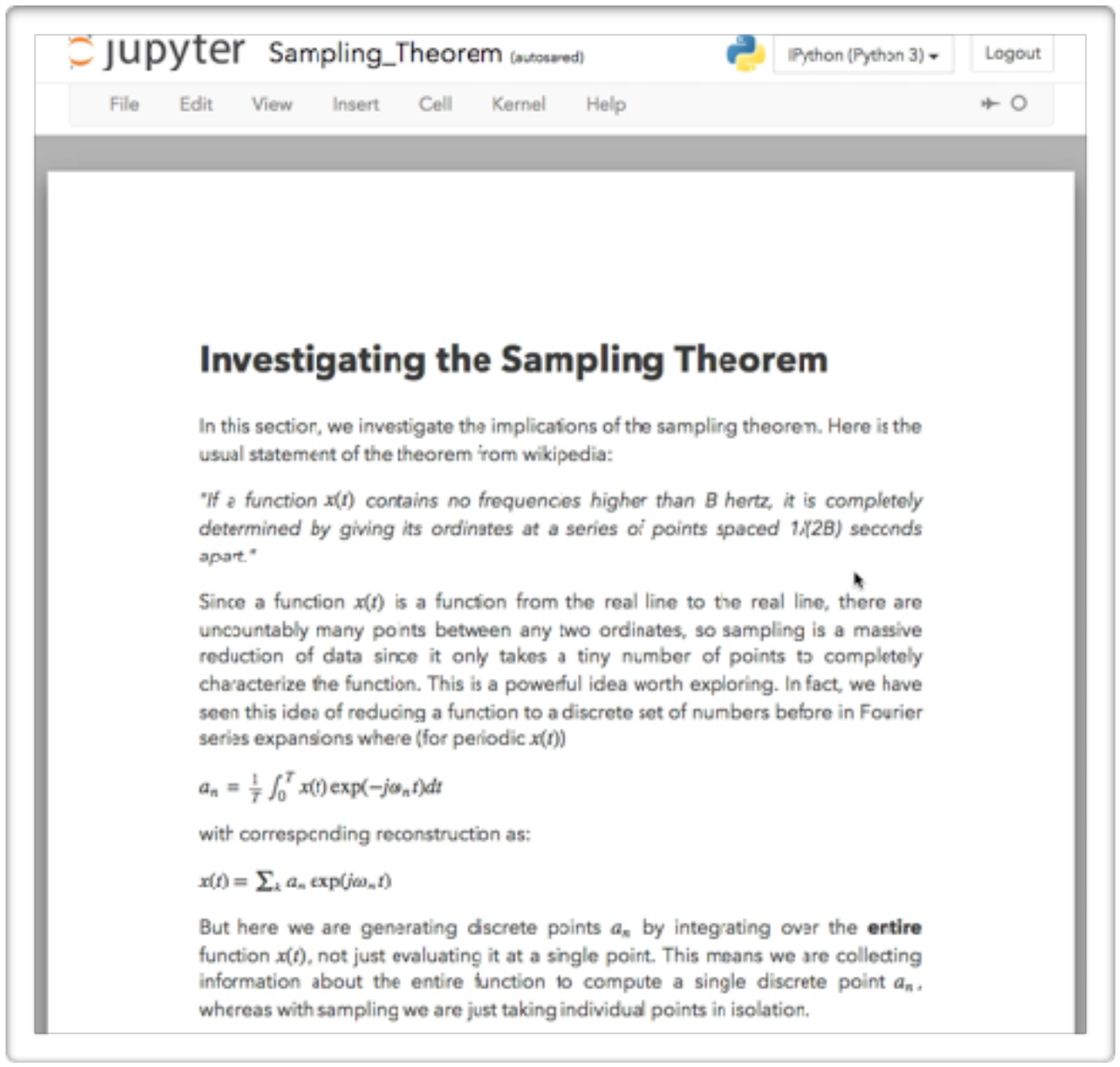
- notebook = sequence of cells
- text cell = markdown + latex)
- code cell = REP (input + output)
- metadata everywhere



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

Jupyter Notebooks

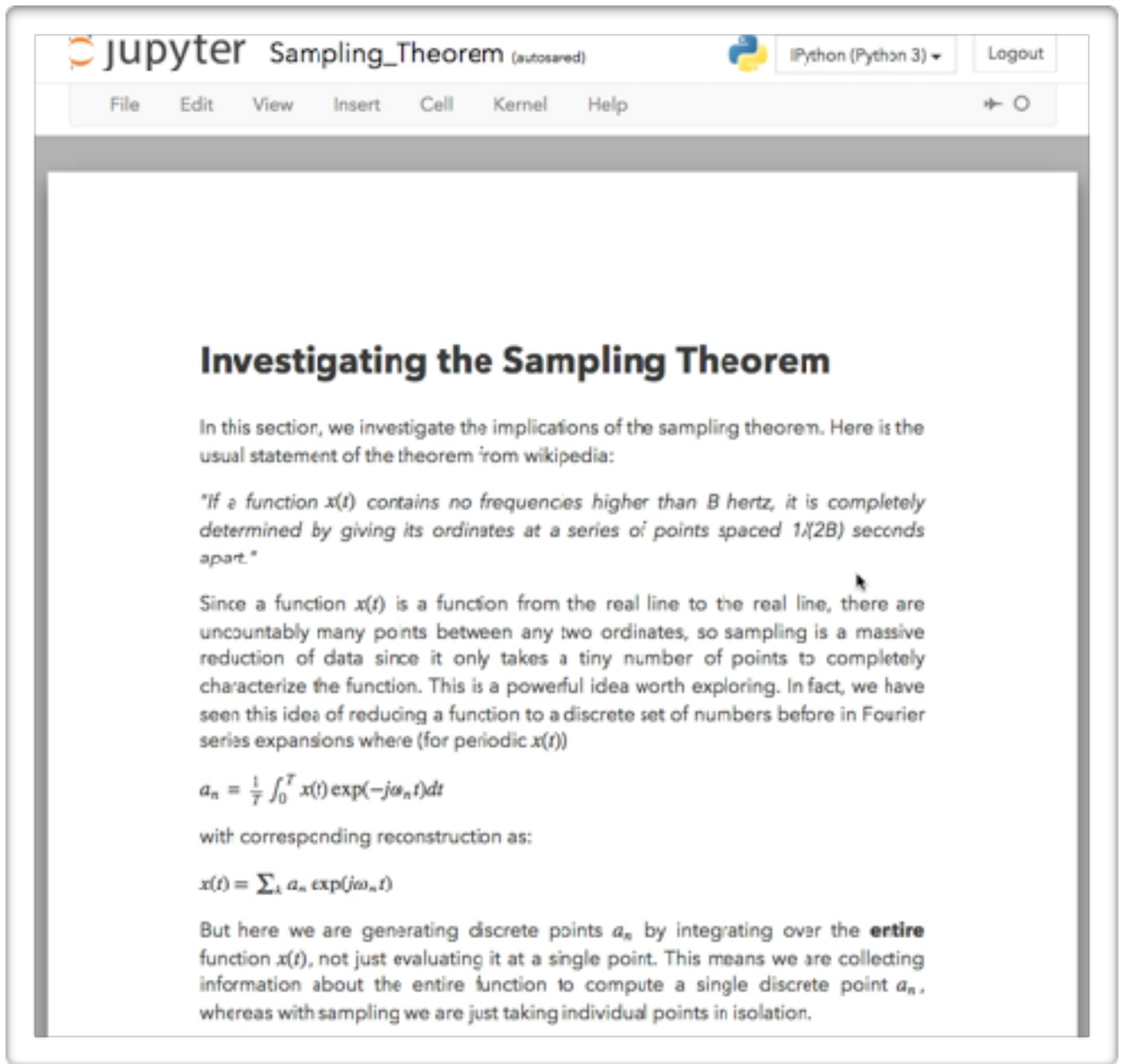
- Plain Text (JSON)



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

Jupyter Notebooks

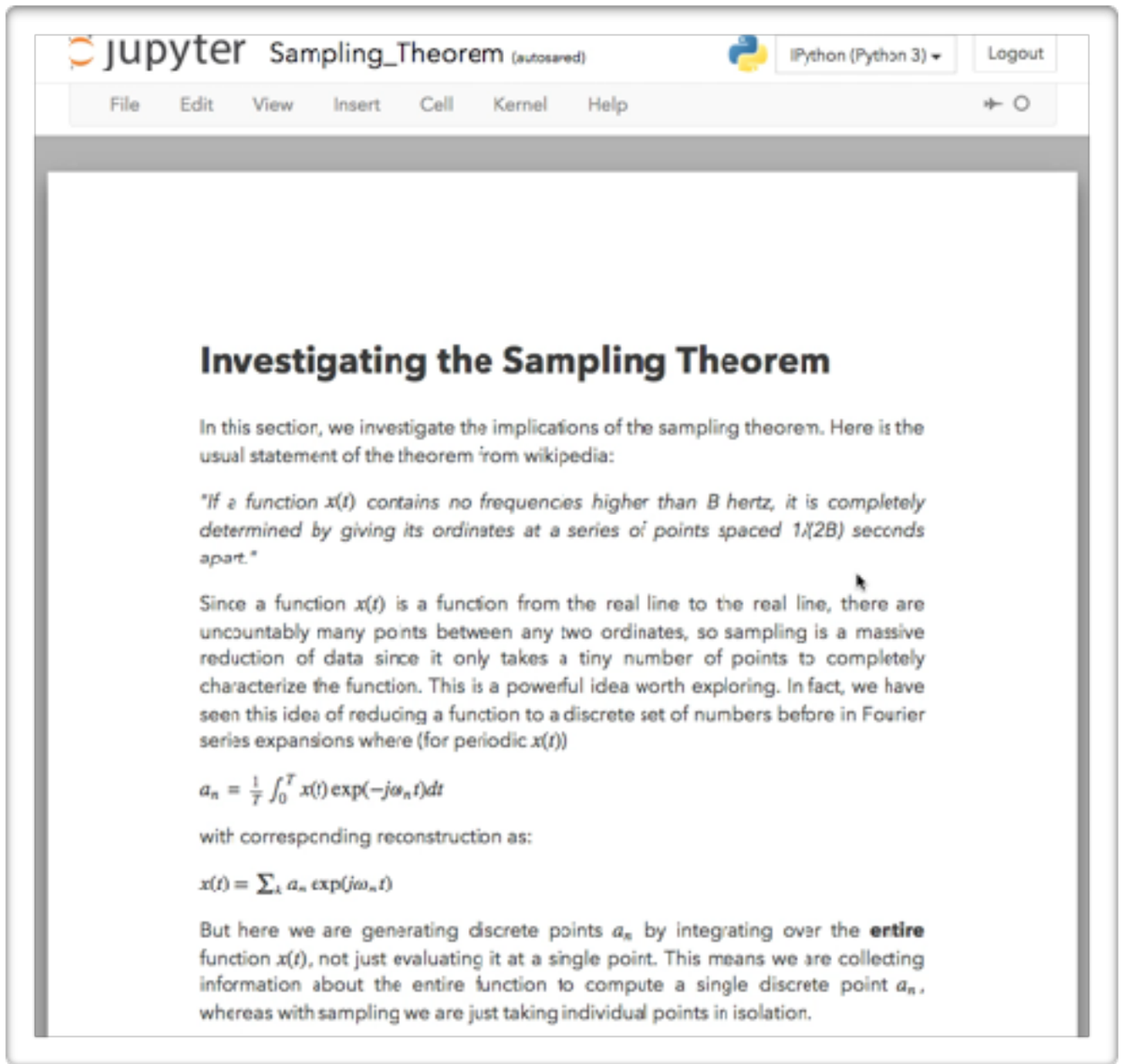
- Plain Text (JSON)
- Publicly documented schema



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

Jupyter Notebooks

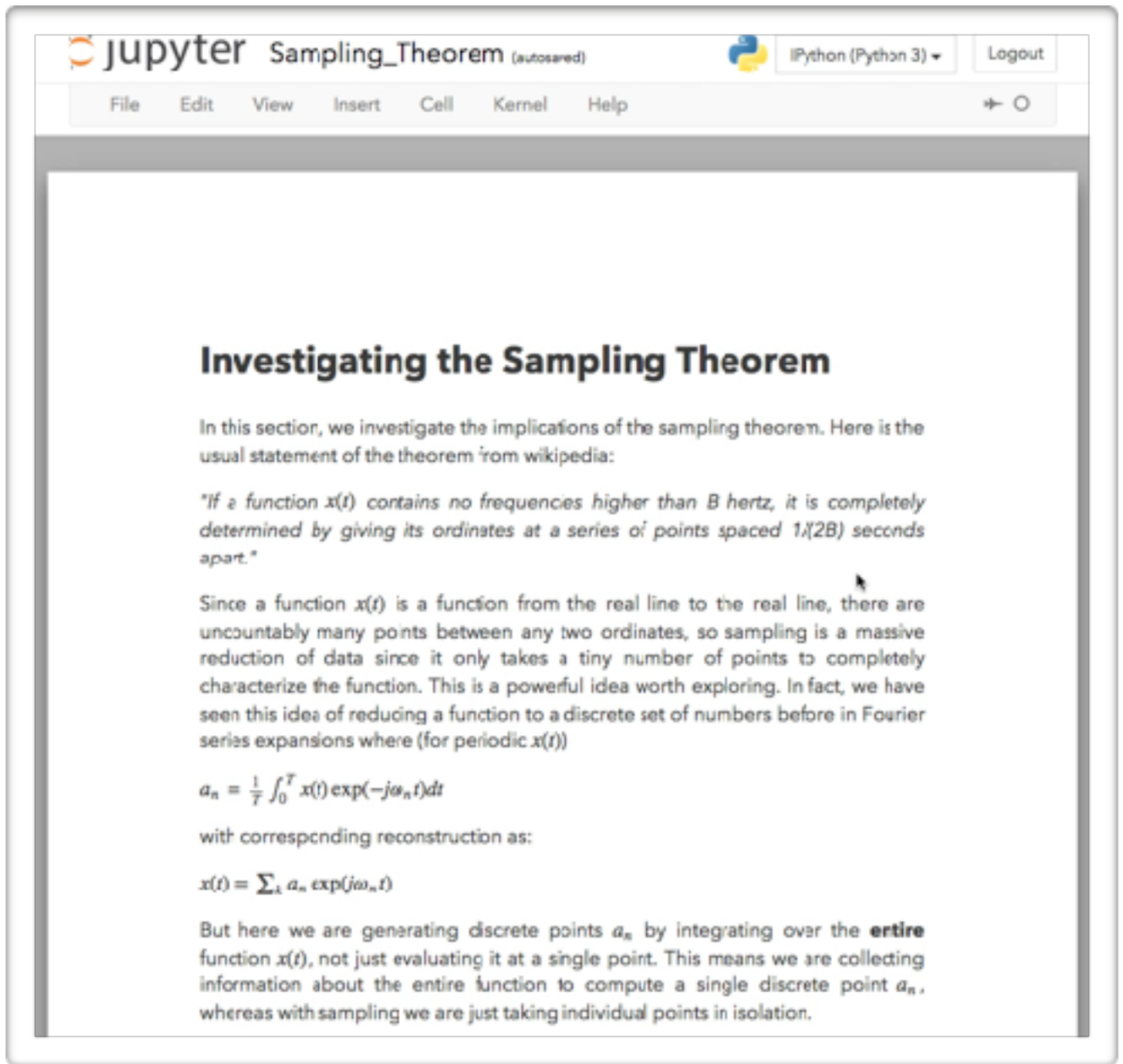
- Plain Text (JSON)
- Publicly documented schema
- Machine readable, easy to understand



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

Jupyter Notebooks

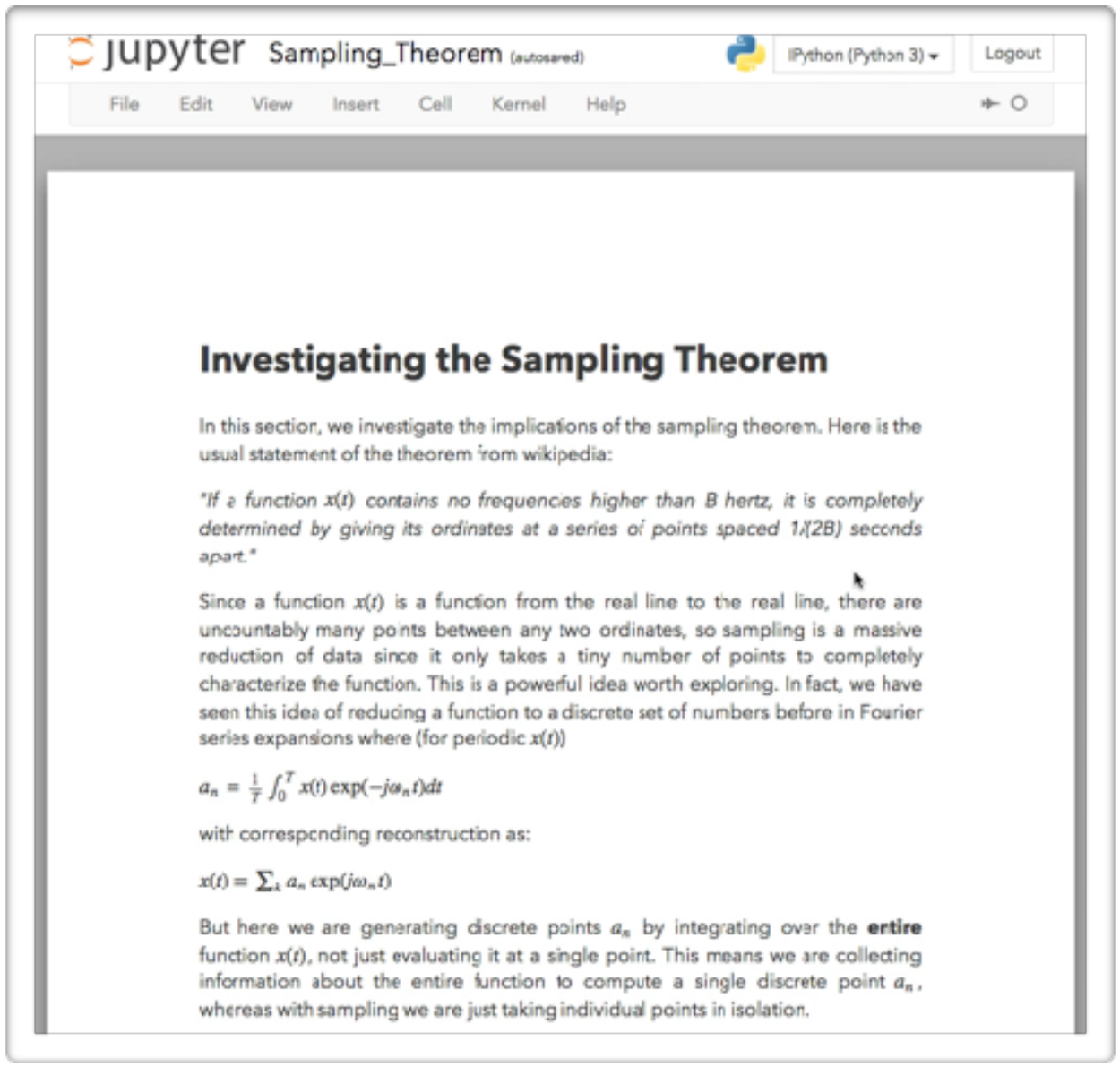
- Plain Text (JSON)
- Publicly documented schema
- Machine readable, easy to understand
- Transformable (nbconvert)



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

Jupyter Notebooks

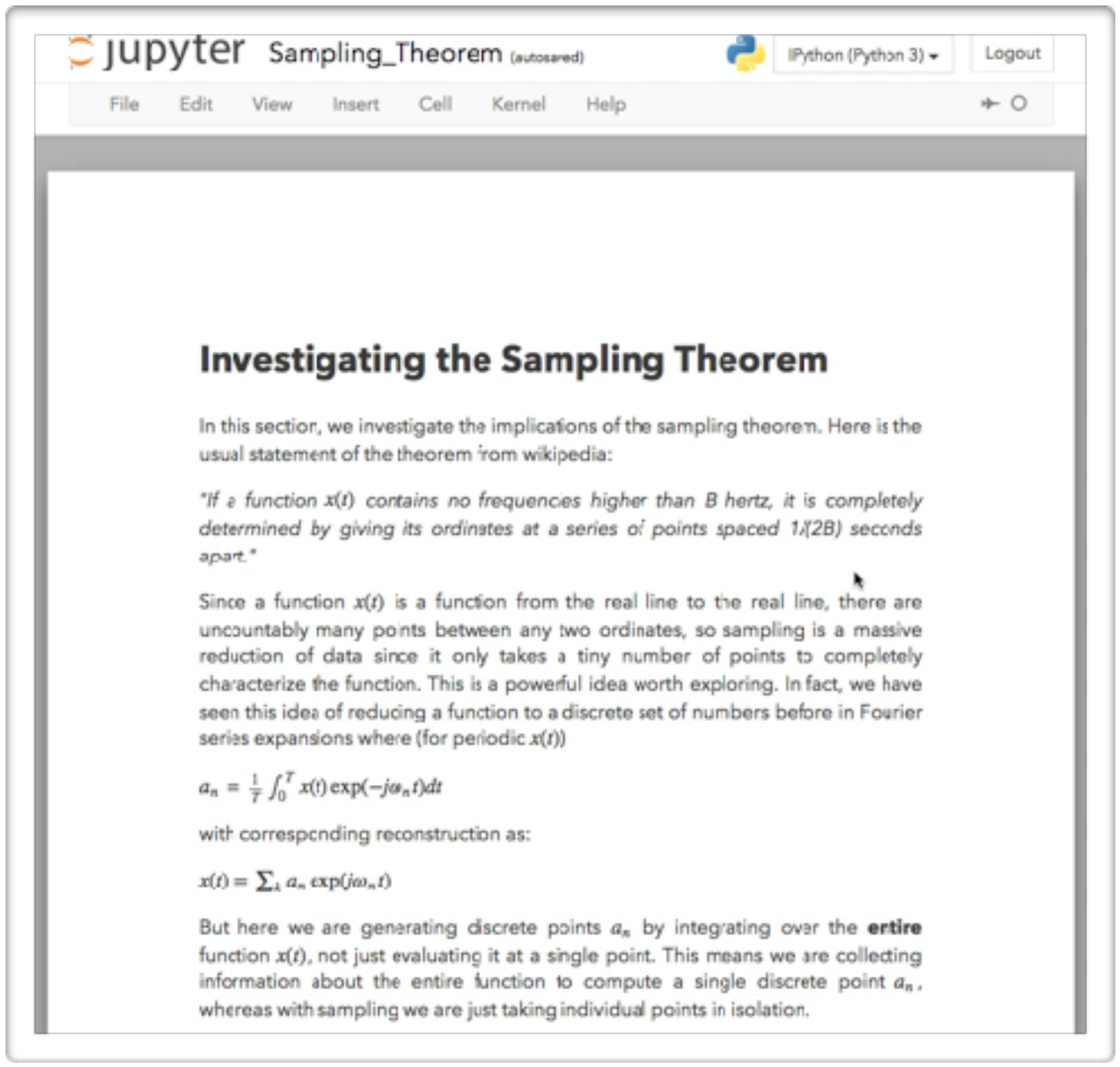
- interactive environment



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

Jupyter Notebooks

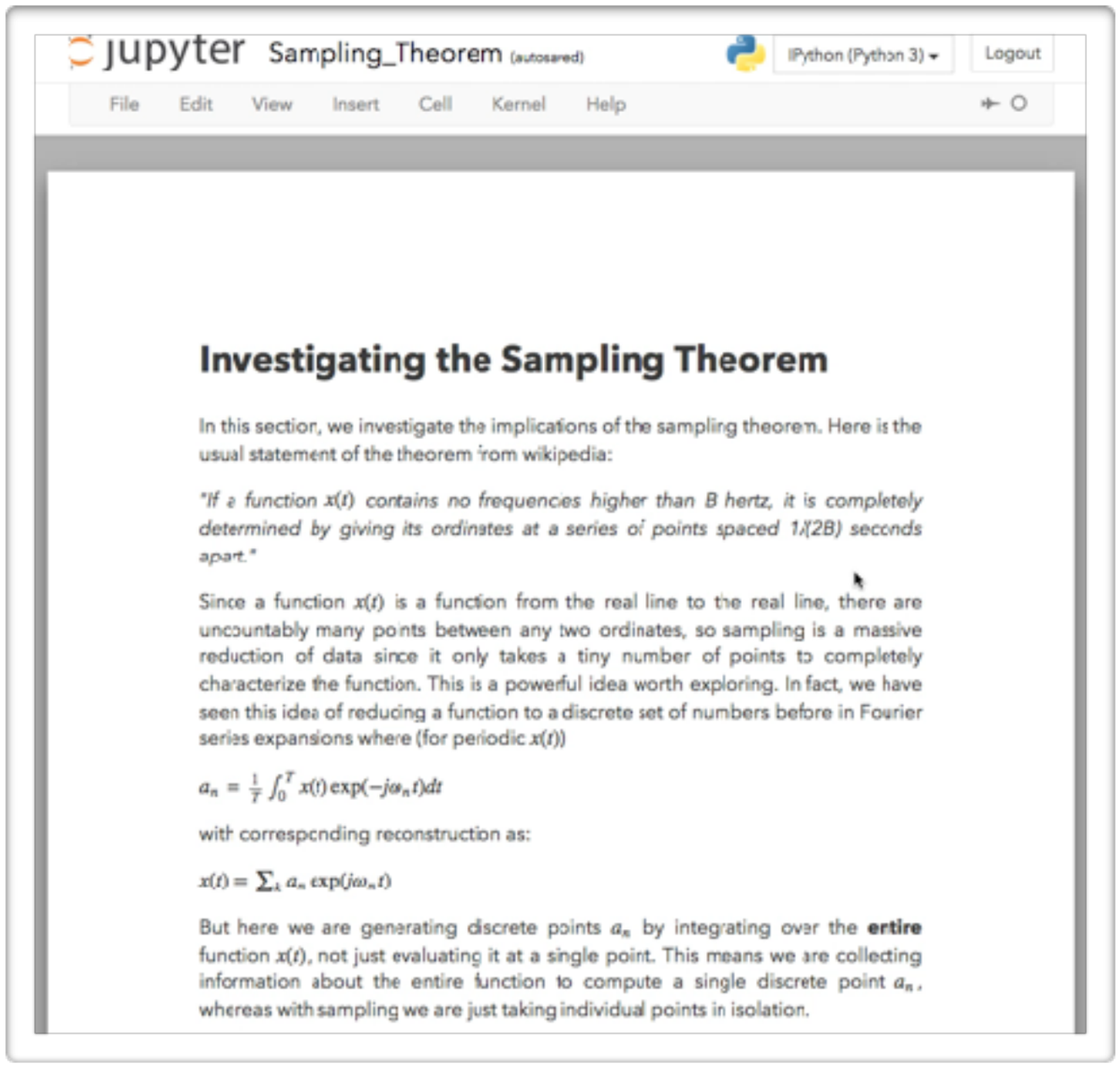
- interactive environment
- input format



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

Jupyter Notebooks

- interactive environment
- input format
- output format



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

Lifecycle of a Computational Idea

Lifecycle of a Computational Idea

1. Explore an idea interactively in a Notebook

Lifecycle of a Computational Idea

1. Explore an idea interactively in a Notebook

2. *Build/add to a library based on what you learn*

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

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

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

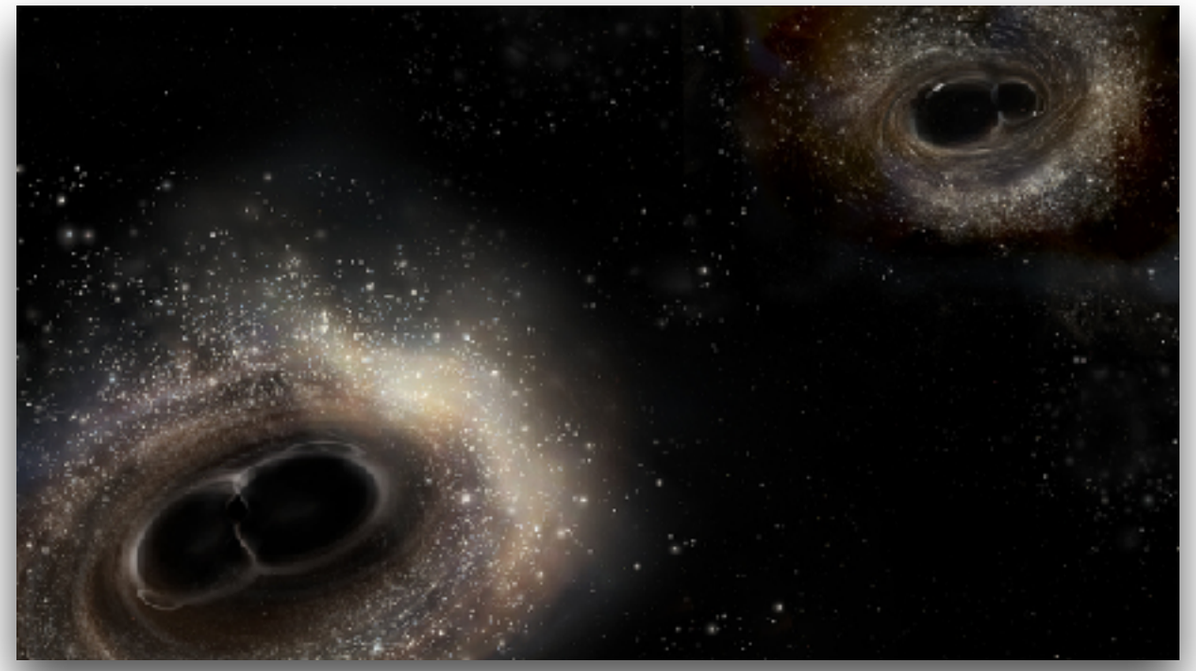
4. Document, demonstrate, and share in Notebooks

5. Computational companions, reproducible papers

Applications of Jupyter

- **jupyterlab** - new UI, javascript APIs for notebooks
- **nbconvert** - convert notebooks to other formats (rst, html, latex/pdf, markdown, script, reveal.js slides)
- **nbviewer** - nbconvert to html on the web
- **jupyterhub** - multi-user notebook server for classes, groups
- **nbgrader** - automated grading of notebooks
- **tmpnb** - containerized (docker) transient deployments of notebooks
- **thebe** - transient kernels on the web, without notebooks
- **binder** - turn a GitHub repo of notebooks into a free executable environment
- **nbdime** - diffing and merging notebooks
- **nteract** - javascript implementation of Jupyter for electron- and react-based applications
- **dashboards** - 2D layout, hiding input for presentations of notebook results
- **dexy** - reproducible document-based workflows

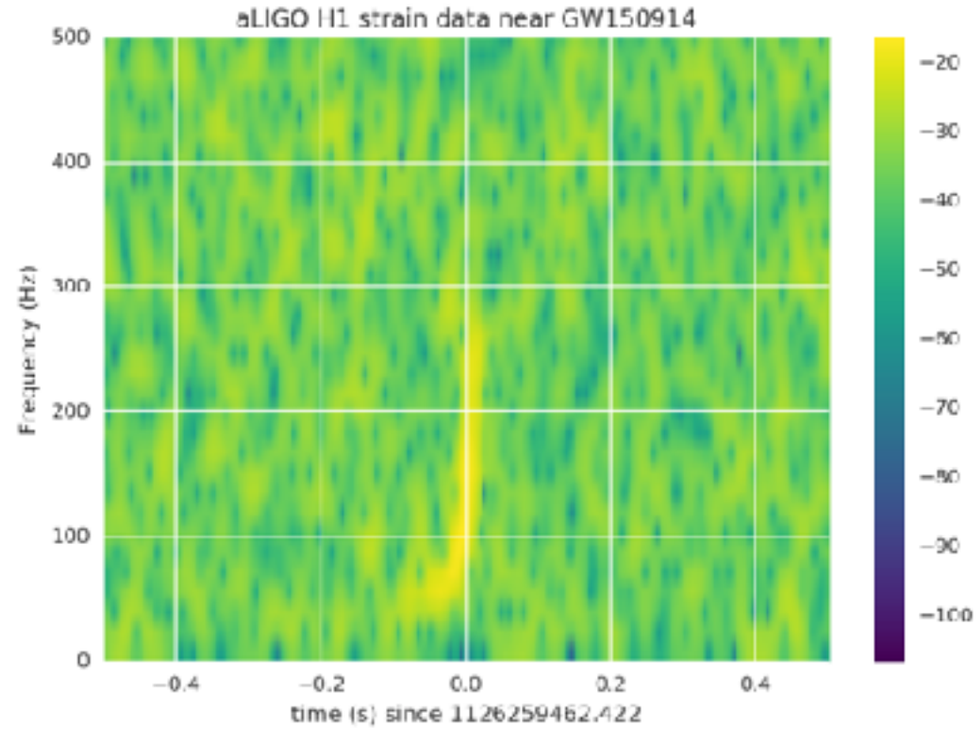
Jupyter in Science



LIGO

- Direct Observation of Gravitational Waves
- Published data and analysis as Jupyter Notebooks

```
# Plot the L1 whitened spectrogram around the signal
plt.figure()
spec_H1, freqs, bins, im = plt.specgram(strain_L1_whiten[indxt], N
                                        noverlap=NOVL, cmap=spec_cm
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('Frequency (Hz)')
plt.colorbar()
plt.axis([-0.5, 0.5, 0, 500]);
```

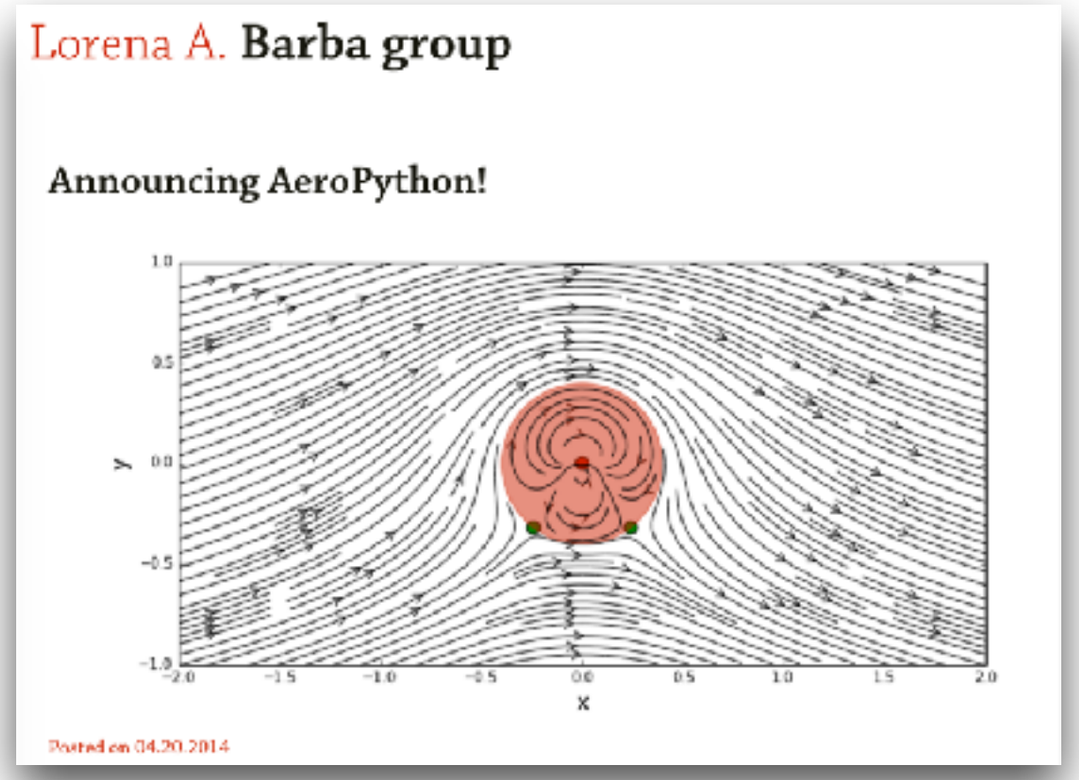


https://losc.ligo.org/events/GW150914/
https://github.com/minrk/ligo-binder

image: https://www.ligo.caltech.edu/news/ligo20160615

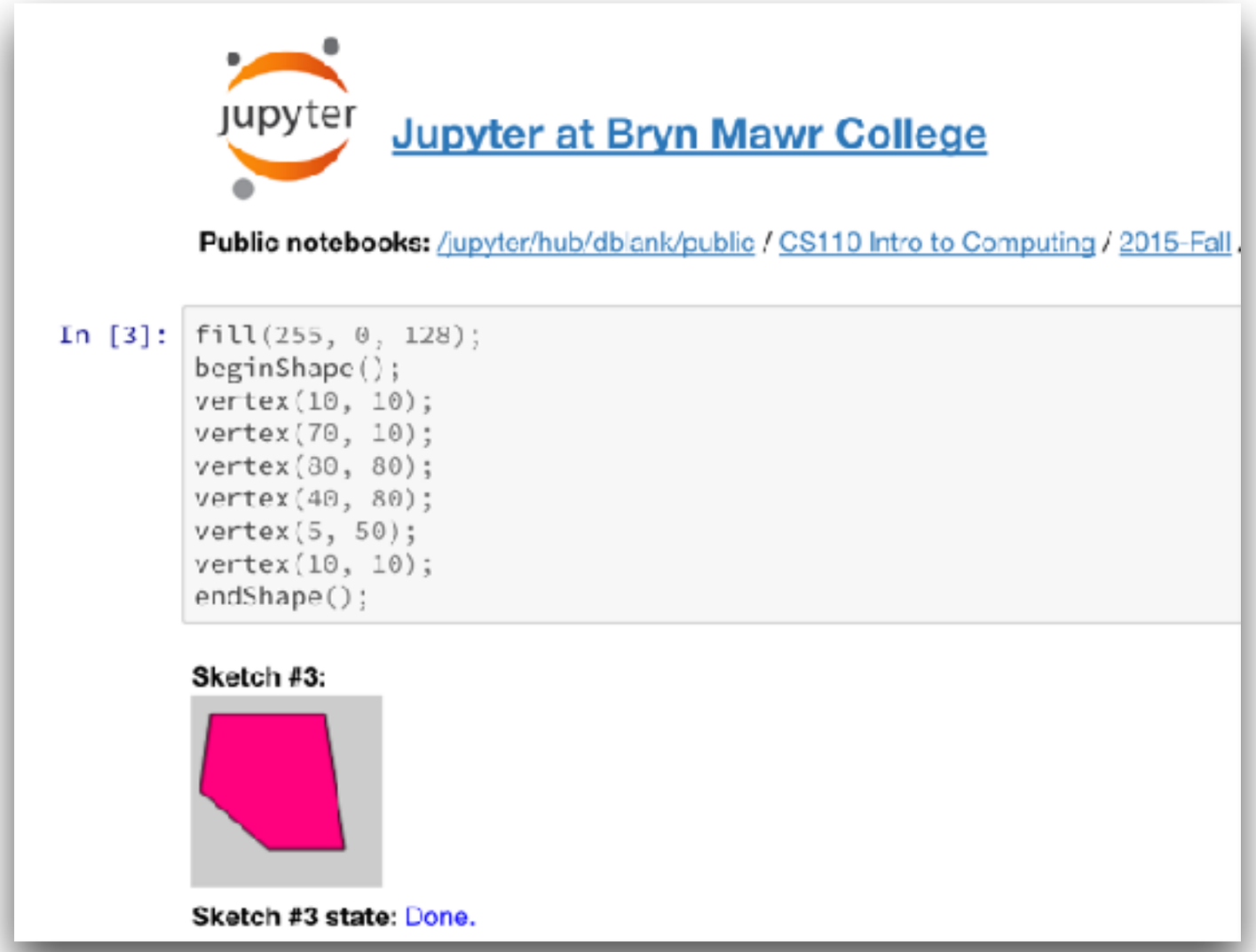
Jupyter in the Classroom

Lorena Barba
AeroPython
Aerodynamics, George Washington U
Flipped Classroom



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

Jupyter in the Classroom

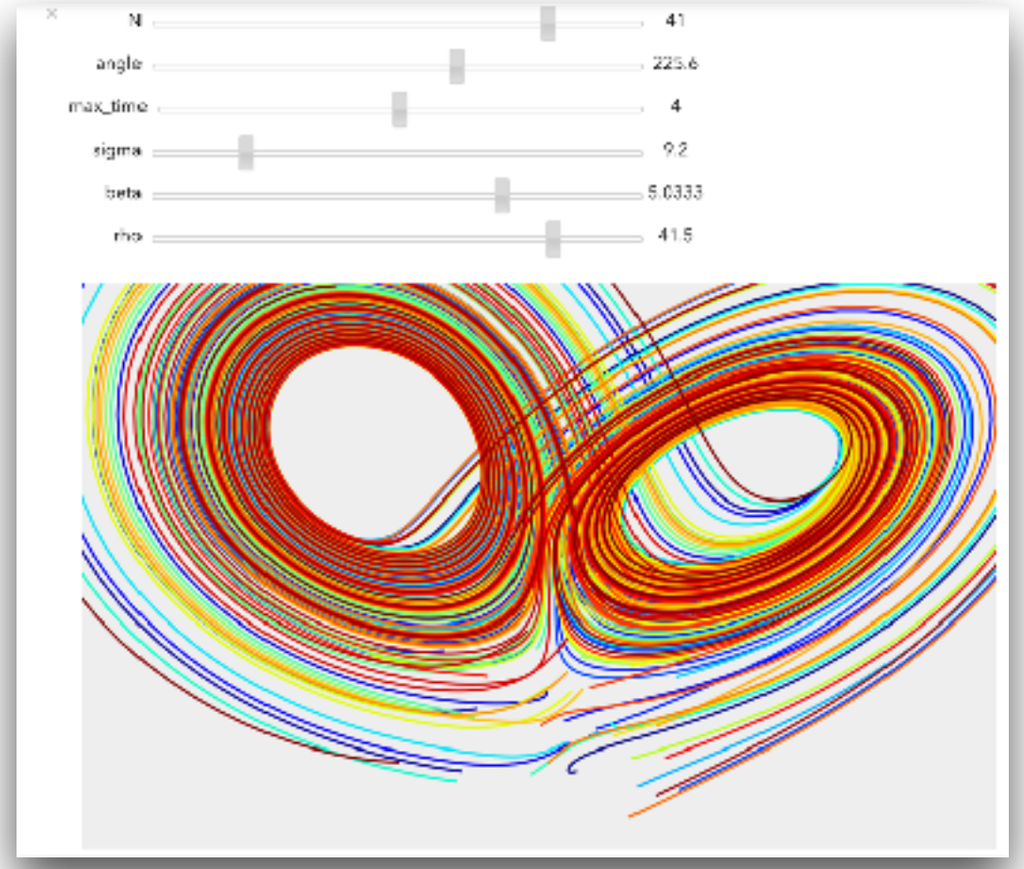


Doug Blank
CS, Brynmawr
Calysto (Multiple languages, not Python)
JupyterHub

Jupyter in the Classroom

Brian Granger
Computational Physics, Cal Poly

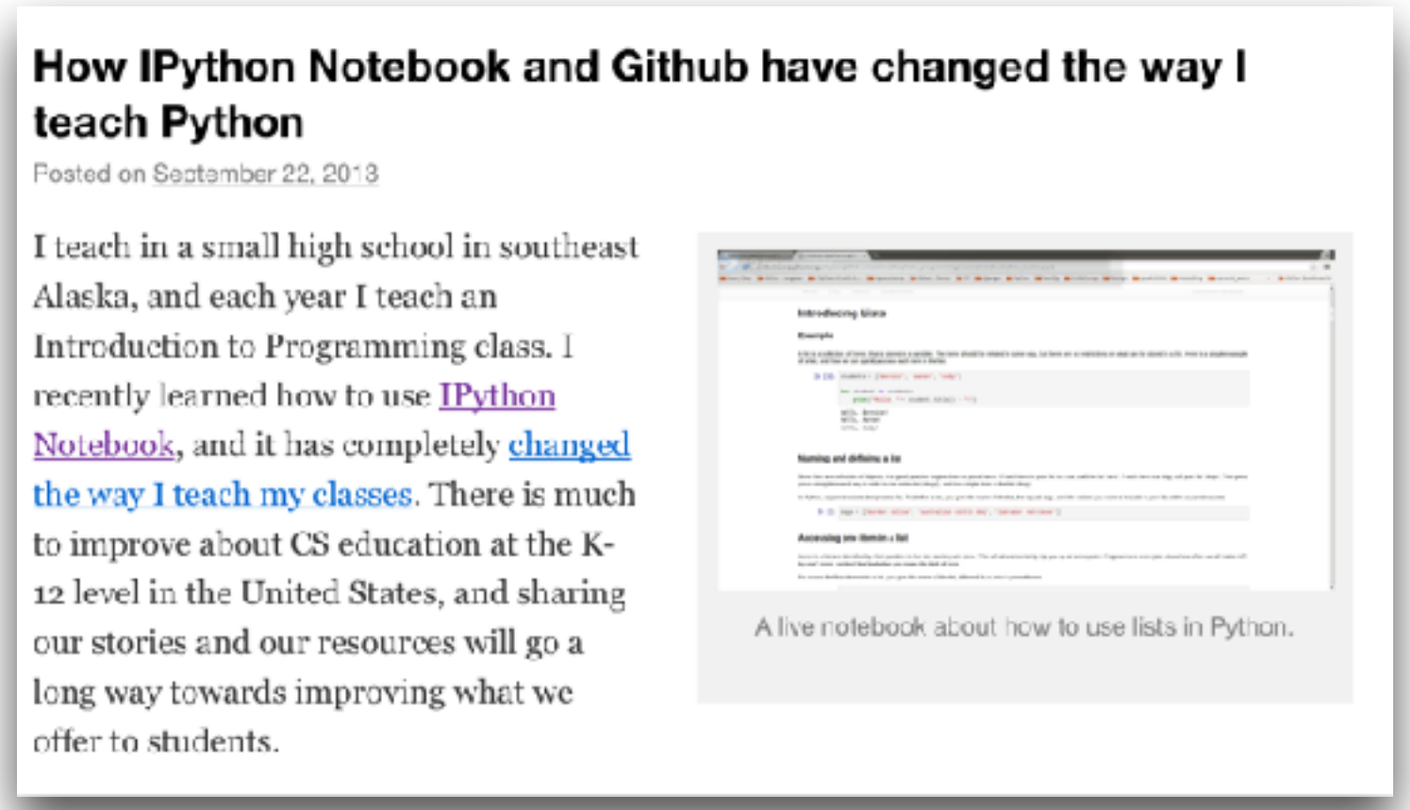
JupyterHub



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.

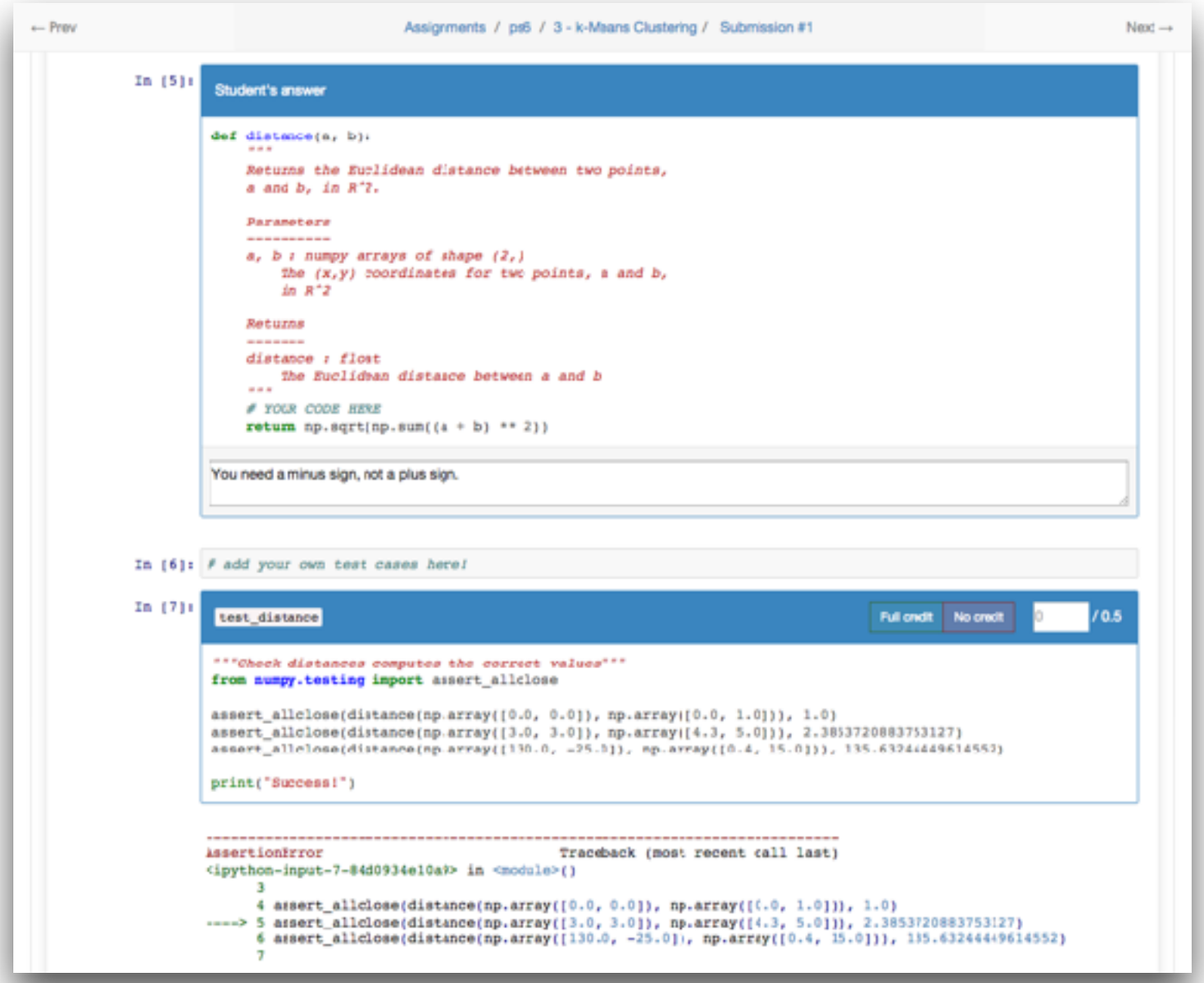
A live notebook about how to use lists in Python.

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




jupyter