The Architecture of Jupyter

Fernando Pérez
(@fperez_org & fperez@lbl.gov)

LBL & UC Berkeley
Mandatory COI disclosure
“The purpose of computing is insight, not numbers”

–Hamming’62
Every research discipline is now awash in data

Astronomy: LSST

Sociology: The Web

Biology: Sequencing

Economics: POS terminals

Physics: LHC

Personalized, data-driven medicine

Neuroscience: EEG, fMRI
IPython: Interactive Python, 2001

- Object Introspection (TAB!)
- OS Integration
- Rich terminal client
- GUI support (plots, ...)
- %magic commands
- Embeddable
Team today: where all the credit goes

Plus ~ 500 more Open source contributors!
Funding and partnerships
Pragmatic abstractions: vocabulary
Interactivity as a protocol

- The REPL as a network protocol
- Kernels
  - execute code
- Clients
  - Read input
  - Present output

Simple abstractions enable rich, sophisticated clients
The IPython/Jupyter Notebook

- Rich web client
- Text & math
- Code
- Results
- Share, reproduce.
From IPython to Project Jupyter
IPython

- Interactive Python shell at the terminal
- Kernel for this protocol in Python
- Tools for Interactive Parallel computing

Jupyter

- Network protocol for interactive computing
- Clients for protocol
  - Console
  - Qt Console
  - Notebook
- Notebook file format & tools (nbconvert...)
- Nbviewer

Language Agnostic
Protocols: kernels & clients
Jupyter Protocol is language agnostic

~75 different kernels: https://github.com/ipython/ipython/wiki/IPython-kernels-for-other-languages
Alternate clients (nb): nteract

- Local desktop application
- Written in node.js (uses React)
- Uses:
  - Jupyter messaging protocols
  - Notebook file format.
- [https://github.com/nteract/nteract](https://github.com/nteract/nteract)

Kyle Kelley
Safia Abdalla
Alternate clients (editor): hydrogen

```python
df = pd.DataFrame({
    'A': 1.,
    'B': pd.Timestamp('20130102'),
    'C': pd.Series(1, index=list(range(4)), dtype='float32'),
    'D': np.array([3] * 4, dtype='int32'),
    'E': pd.Categorical(['test', 'train', 'test', 'train']),
    'F': 'foo'
})
```
Alternate clients (IDE): spyder

https://github.com/spyder-ide/spyder
Jupyter Kernel Gateway

- Web server for spawning and communicating with kernels over HTTP/Websocket
-Defaults to letting web clients talk the Jupyter protocol
- Extensible with other modes / personalities

Slides/credit: Peter Parente, @parente
Notebooks: structured files (metadata!)
Part A (2 points)

Write code to compute the mean of a list of numbers.

```python
In [ ]:
def mean(x):
    """Compute the mean of a list of numbers given in `x`."""
    BEGIN SOLUTION
    return sum(x) / len(x)
    END SOLUTION
```

```python
In [ ]:
"""Check that the `mean` function is correct."""
assert mean([1]) == 1.0
assert mean([1, 2]) == 1.5
assert mean([5.5, 0, 2, 3.4]) == 2.725
assert mean(range(100)) == 49.5
assert mean(range(100, 0, -1)) == 50.5
```

Part B (3 points)

Describe the difference between an arithmetic mean, a harmonic mean, and a geometric mean.

Arithmetic mean:

\[
\text{Arithmetic mean: } \frac{1}{N} \sum_{i=1}^{N} x_i
\]
Notebooks as dashboards

- Same file format
- Metadata based
- Live dashboard with Jupyter kernel
- Web view with hidden details (code, setup, etc)

https://github.com/jupyter-incubator/dashboards
“Executable books”

By Cameron Davidson-Pilon

By Jose Unpingco

By Matthew Russell

By Cyrille Rossant

By Cameron Davidson-Pilon
“The Notebook”: reusable web application
Oriole: executable, video-narrated tutorials

Oriole Online Tutorials
Learn alongside smart people solving hard problems

Oriole is a unique new medium that blends code, data, text, and video into a narrated learning experience with executable content.

Led by some of the most brilliant minds in technology, each lesson is an easily digestible and engaging thought-by-thought tour of the instructor’s approach to the problem in both narrative and executable code. No set-up or installation is necessary. Oriole Online Tutorials require nothing more than an internet connection and a laptop. You can write and run code within the environment. Make a mistake? Change it, and try again.

Oriole combines the expert insight and hands-on learning of in-person or online courses with the on-demand, at-your-own-desk, back-up-and-run-it-again convenience of video training. You learn by doing, on your own schedule, and at your own pace.

In Oriole, we get the complete integration of video synchronized with the flow of the text, as well as the ability to execute the code: this is probably as close as we can get to learning side-by-side with Peter himself.

Fernando Perez, creator of IPython, which evolved into Project Jupyter.
Microsoft, IBM, Google, Continuum...
JupyterHub: multiuser support

JupyterHub is a multiuser version of the notebook designed for centralized deployments in companies, university classrooms and research labs.

- **Pluggable authentication**: Manage users and authentication with PAM, OAuth or integrate with your own directory service system. Collaborate with others through the Linux permission model.
- **Centralized deployment**: Deploy the Jupyter Notebook to all users in your organization on centralized servers on- or off-site.
- **Container friendly**: Use Docker containers to scale your deployment and isolate user processes using a growing ecosystem of prebuilt Docker containers.
- **Code meets data**: Deploy the Notebook next to your data to provide unified software management and data access within your organization.
Berkeley’s *Foundations of Data Science*

- New curriculum aimed at all freshmen at UC Berkeley
- Interactive textbook is Jupyter Notebooks
- Course deployment is JupyterHub
- Off Jess Hamrick’s work

data.berkeley.edu, data8.org
Data Science: Connector Courses

BERKELEY DATA SCIENCE EDUCATION PROGRAM
Fall 2016 Connector Course Offerings
Designed to Complement
Data 8: Foundations of Data Science

data.berkeley.edu
JupyterHub: interactive HPC

Cori @ NERSC: Department of Energy Supercomputing Center (LBNL)

Shreyas Cholia
Rollin Thomas
JupyterLab: the notebook, evolved...
The “Notebook”?
JupyterLab: unifying these ideas

Jason Grout
(here)

+Brian, Steven, Darian, Sylvain, Carol, Cameron, Farica, Paul, Reese, Kyle, Chris, Ian, Matthias, ...
Live Demo!

Demo credits / thank you:
Brian Granger (Cal Poly SLO)
Jason Grout (Bloomberg)
Thank You