

# Introduction to Jupyter Notebook

Ward Fisher

Unidata Python Workshop

October 17-19, 2016

# Overview

- Brief overview of Jupyter Notebook
  - What is it?
  - How do you use it?

# Overview

- Brief overview of Jupyter Notebook
  - What is it?
  - How do you use it?
- Examples of basic Jupyter Notebook Usage.

# Overview

- Brief overview of Jupyter Notebook
  - What is it?
  - How do you use it?
- Examples of basic Jupyter Notebook Usage.
- Discuss some Advanced Jupyter Notebook Uses.

# Overview

- We will not be going too in-depth with what you can do in regards to using Jupyter Notebooks for actual science.

# What is Jupyter Notebook?

The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more. (<http://jupyter.org>)



Open source, interactive data science and scientific computing  
across over 40 programming languages.

# What is Jupyter Notebook?

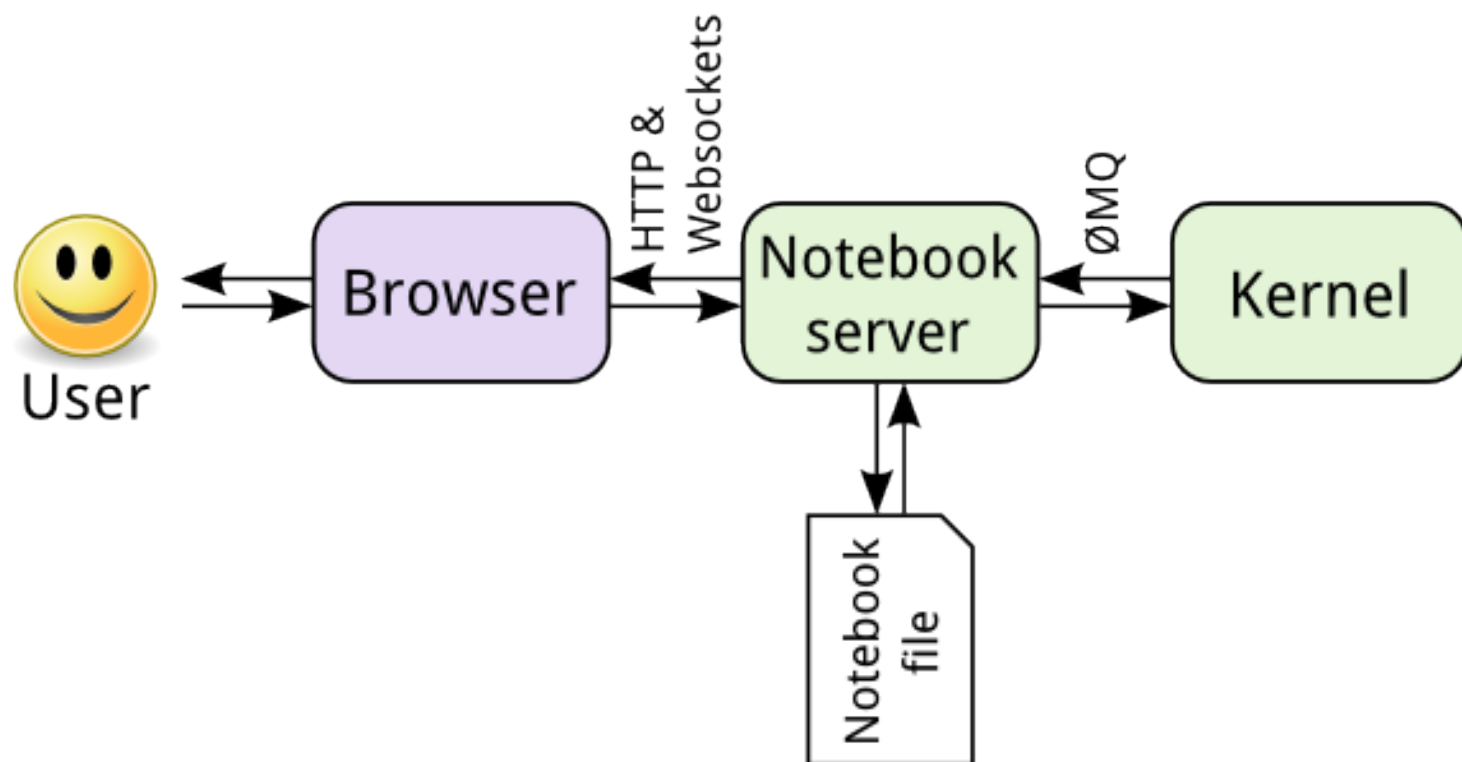
Alternatively: Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include:

- Live code
- Interactive widgets
- Plots
- Narrative text
- Equations
- Images
- Video

These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others using email, Dropbox, version control systems (like git/GitHub) or [nbviewer.jupyter.org](http://nbviewer.jupyter.org).

Open source, interactive data science and scientific computing  
across over 40 programming languages.

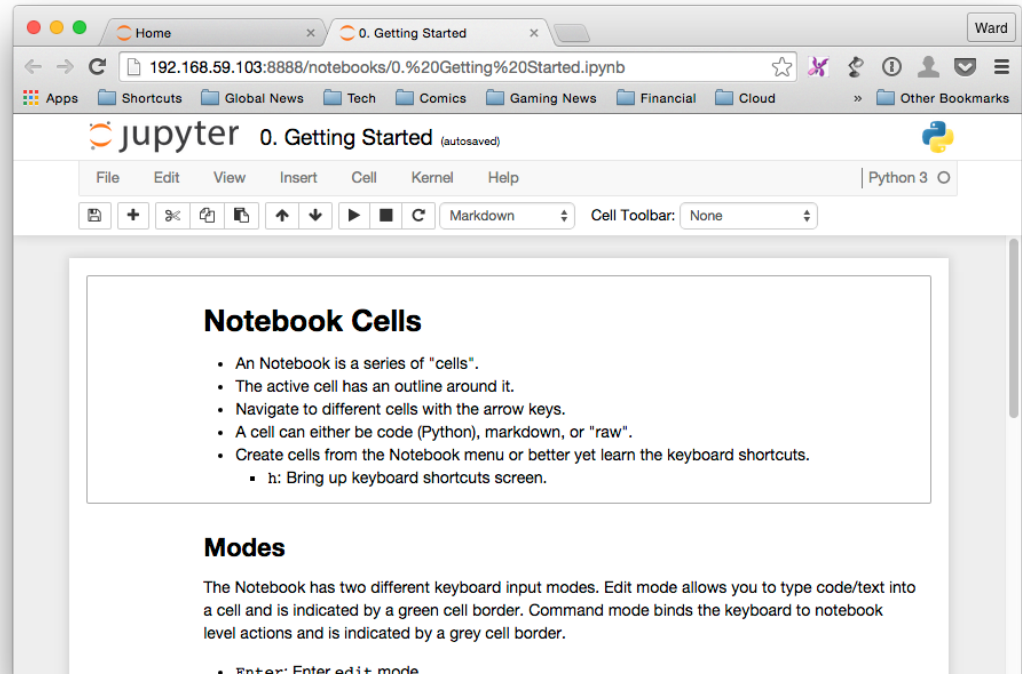
# Jupyter Notebook Components





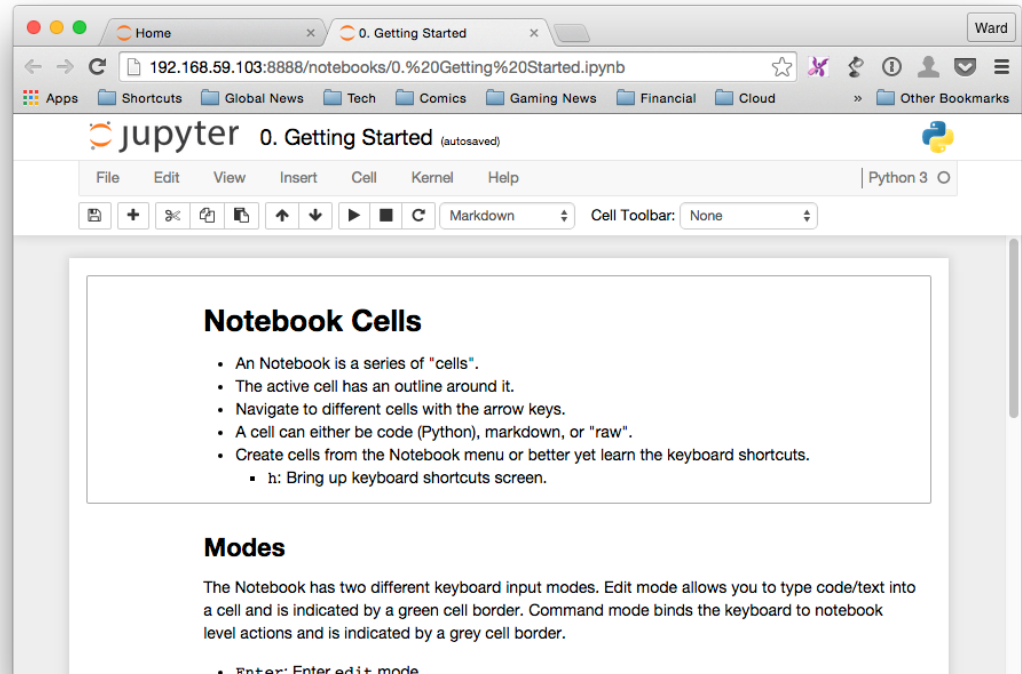
# Jupyter Notebook document

- A Jupyter Notebook is a collection of *cells*.



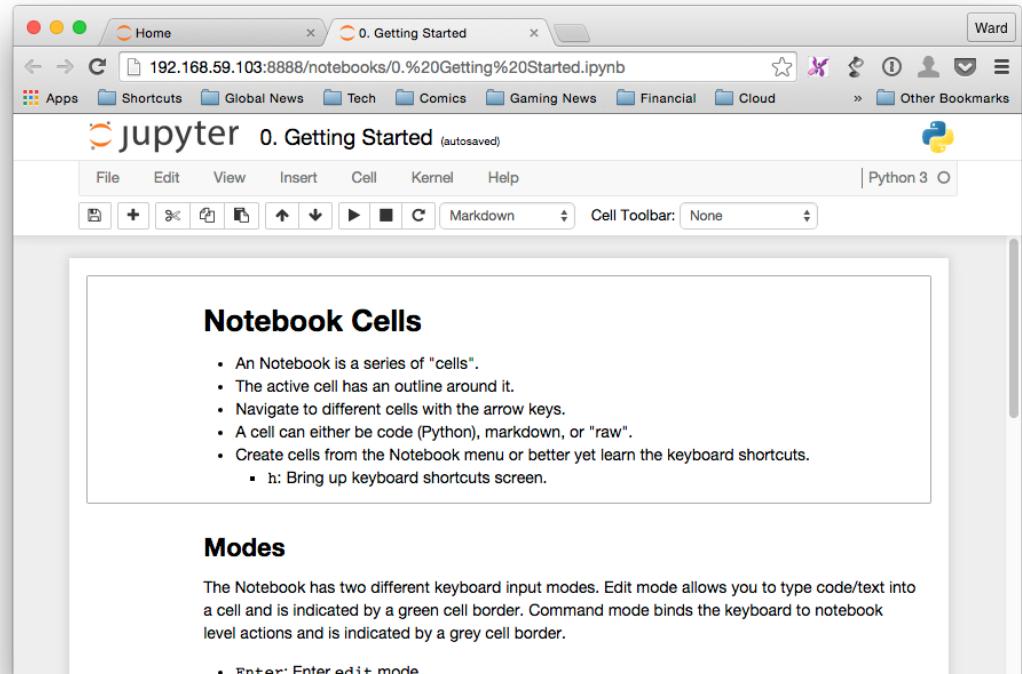
# Jupyter Notebook document

- A Jupyter Notebook is a collection of *cells*.
  - Markdown



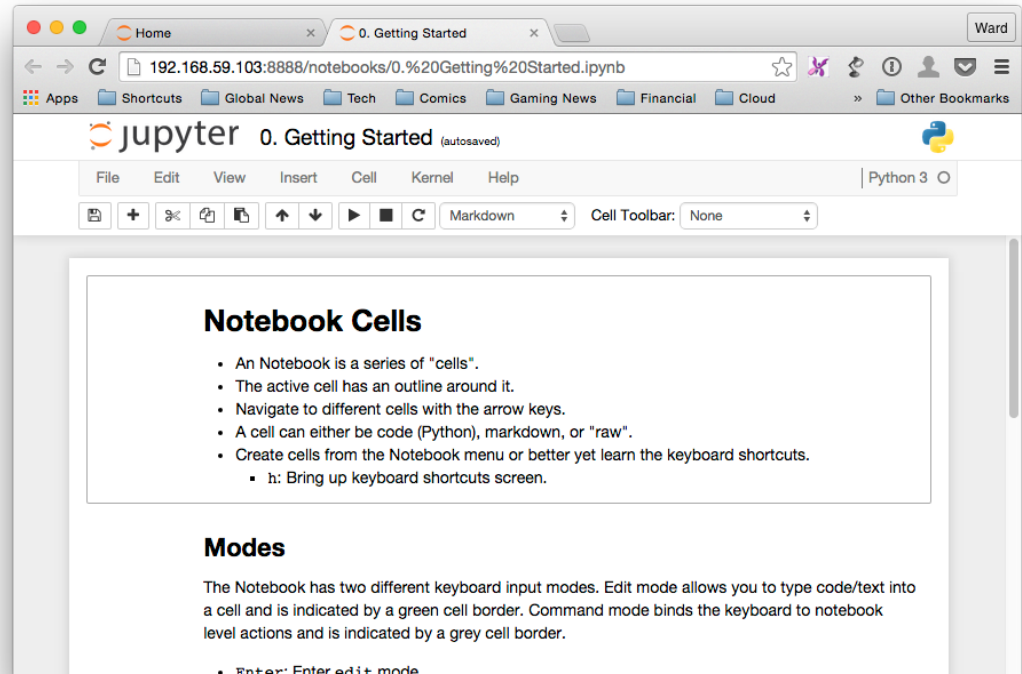
# Jupyter Notebook document

- A Jupyter Notebook is a collection of *cells*.
  - Markdown
  - Code

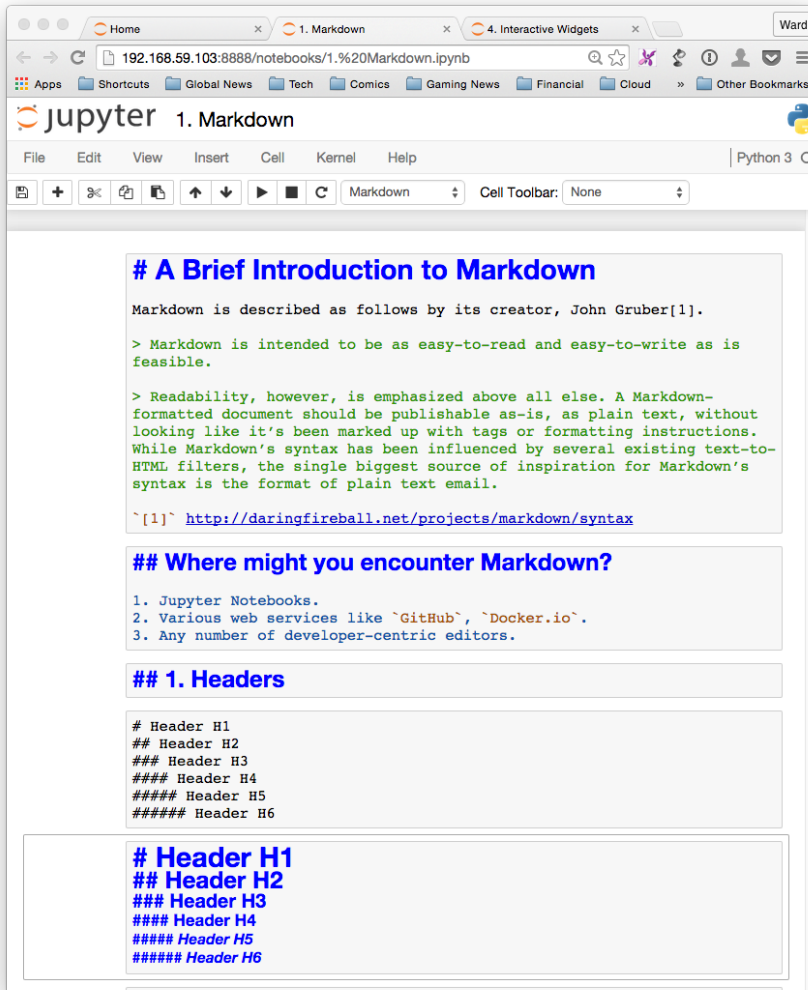


# Jupyter Notebook document

- A Jupyter Notebook is a collection of *cells*.
  - Markdown
  - Code
  - “Raw” - Raw cells are left ‘as is’ and are not processed.



# Markdown Cells



The screenshot shows a Jupyter Notebook browser interface with the title "1. Markdown". The toolbar indicates the current cell type is "Markdown". The content of the cell is raw Markdown source code, including a first-level heading, a paragraph, a code block, and a list. The code is as follows:

```
# A Brief Introduction to Markdown

Markdown is described as follows by its creator, John Gruber[1].

> Markdown is intended to be as easy-to-read and easy-to-write as is
feasible.

> Readability, however, is emphasized above all else. A Markdown-
formatted document should be publishable as-is, as plain text, without
looking like it's been marked up with tags or formatting instructions.
While Markdown's syntax has been influenced by several existing text-to-
HTML filters, the single biggest source of inspiration for Markdown's
syntax is the format of plain text email.

`[1]` http://daringfireball.net/projects/markdown/syntax

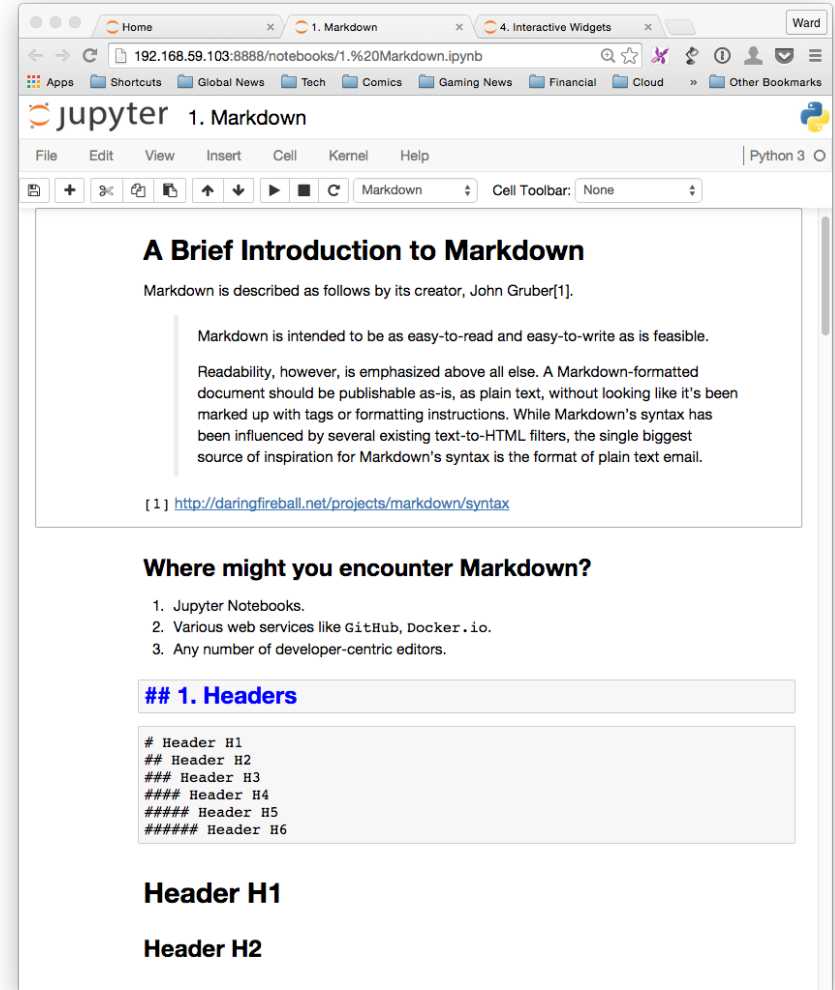
## Where might you encounter Markdown?

1. Jupyter Notebooks.
2. Various web services like `GitHub`, `Docker.io`.
3. Any number of developer-centric editors.

## 1. Headers

# Header H1
## Header H2
### Header H3
#### Header H4
##### Header H5
##### Header H6

# Header H1
## Header H2
### Header H3
#### Header H4
##### Header H5
##### Header H6
```



The screenshot shows the same Jupyter Notebook browser interface, but the cell type is now "Text", and the content is the rendered output of the Markdown source code. The rendered content is as follows:

## A Brief Introduction to Markdown

Markdown is described as follows by its creator, John Gruber[1].

Markdown is intended to be as easy-to-read and easy-to-write as is feasible.

Readability, however, is emphasized above all else. A Markdown-formatted document should be publishable as-is, as plain text, without looking like it's been marked up with tags or formatting instructions. While Markdown's syntax has been influenced by several existing text-to-HTML filters, the single biggest source of inspiration for Markdown's syntax is the format of plain text email.

[1] <http://daringfireball.net/projects/markdown/syntax>

## Where might you encounter Markdown?

1. Jupyter Notebooks.
2. Various web services like GitHub, Docker.io.
3. Any number of developer-centric editors.

### 1. Headers

#### Header H1

#### Header H2

#### Header H3

#### Header H4

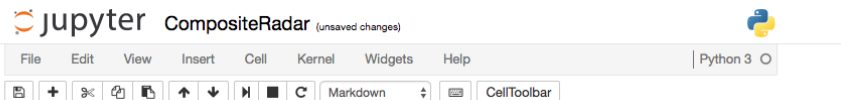
#### Header H5

#### Header H6

## Header H1

## Header H2

# Code Cells



```
data = np.ma.array(data, mask=data<=-30)
```

## Create the Plot

```
In [ ]: # Set up the projection for the LambertConformal projection we know we have
lcc = cartopy.crs.LambertConformal(central_longitude=lon_0, central_latitude=lat_0,
                                   standard_parallels=(lat_0, lat_1))

# Create a large figure and axes with this projection
fig = plt.figure(figsize=(24, 12))
ax = fig.add_subplot(1, 1, 1, projection=lcc)

# Limit to the bounds of the data we have
ax.set_extent([-129., -63., 22., 49.], cartopy.crs.Geodetic())

# Add some map features
ax.stock_img()
ax.coastlines(resolution='50m')
ax.add_feature(cartopy.feature.NaturalEarthFeature(category='cultural',
                                                    name='admin_1_states_provinces_lines',
                                                    scale='50m', facecolor='none'))

ax.add_feature(cartopy.feature.BORDERS, linewidth='2', edgecolor='black')
ax.gridlines()

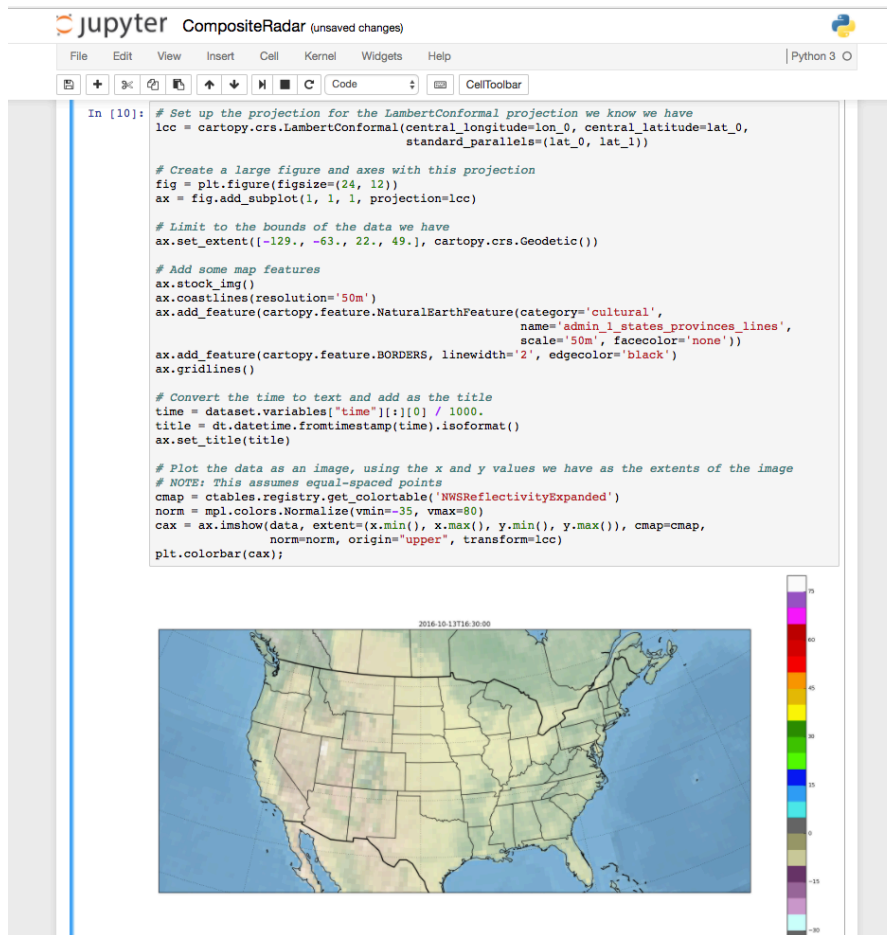
# Convert the time to text and add as the title
time = dataset.variables["time"][0] / 1000.
title = dt.datetime.fromtimestamp(time).isoformat()
ax.set_title(title)

# Plot the data as an image, using the x and y values we have as the extent
# NOTE: This assumes equal-spaced points
cmap = ctables.registry.get_colortable('NWSReflectivityExpanded')
norm = mpl.colors.Normalize(vmin=-35, vmax=80)
cax = ax.imshow(data, extent=(x.min(), x.max(), y.min(), y.max()), cmap=cmap,
                norm=norm, origin="upper", transform=lcc)
plt.colorbar(cax);
```

## Exercise

Using what was done above, plot the digital hybrid reflectivity (DHR):

- Look at <http://thredds.ucar.edu/thredds/catalog/hexrad/composite/gini/catalog.html>
- Instead of plotting over all of the U.S., limit to an area of interest
- DHR was chosen to keep the colormap from the NWS the same. Can also look at:
  - Echo Tops (EET)
  - Digital VIL (DVL)

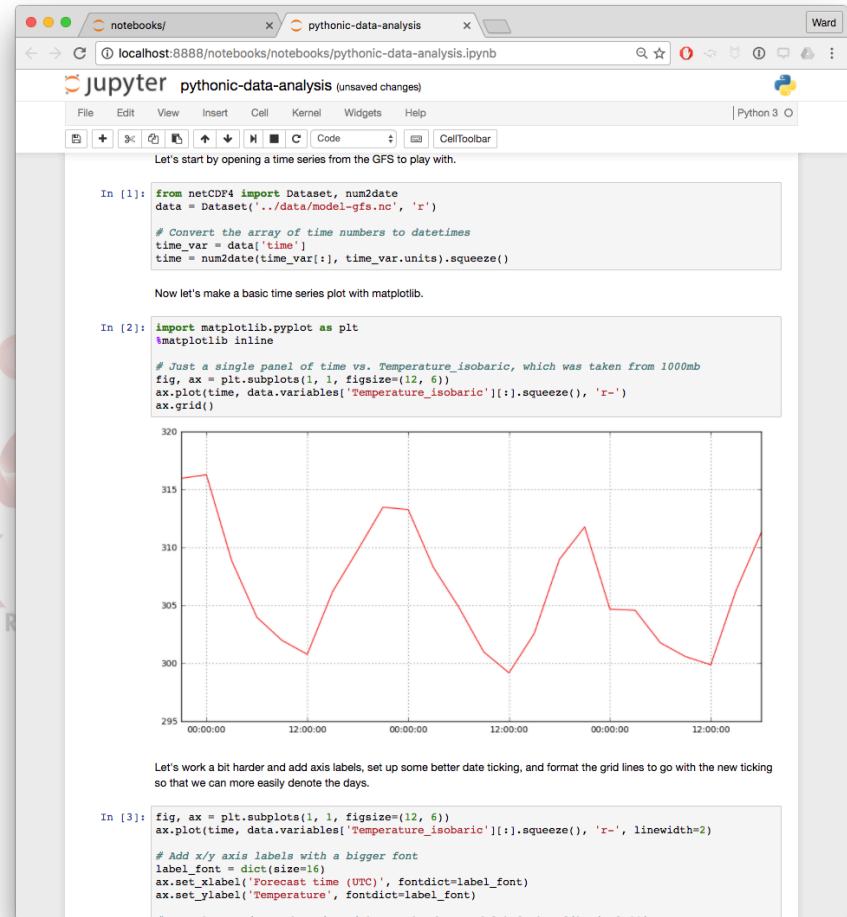


# What does this get you?

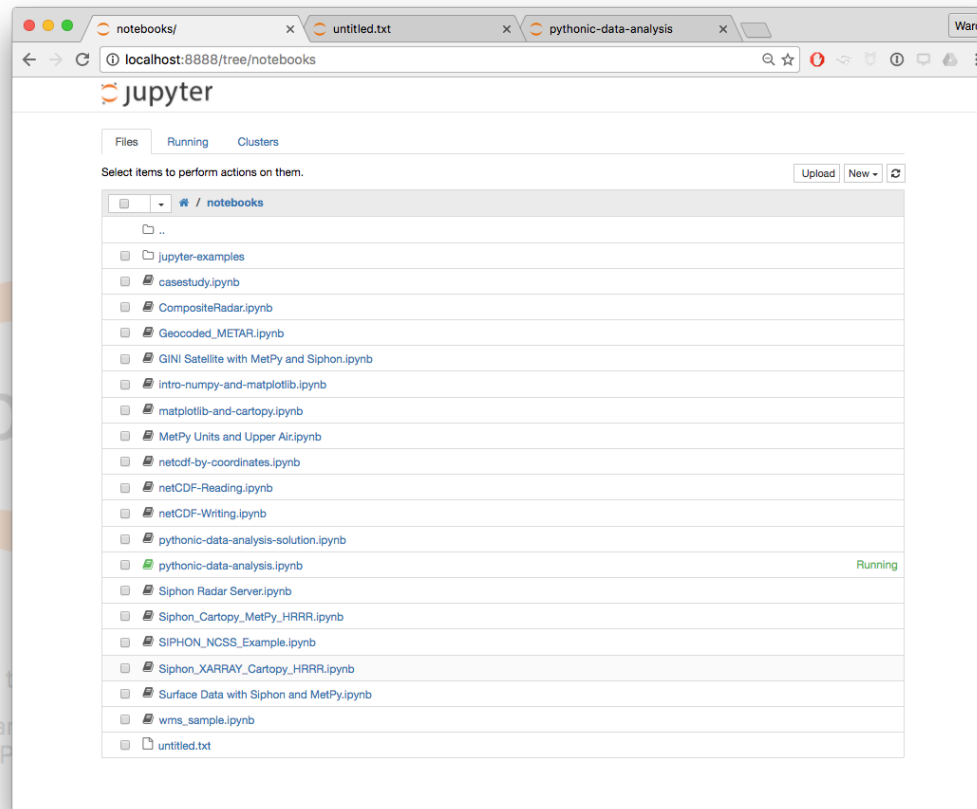
- A sharable document with embedded, reproducible experimental data analysis.

Evolved from the IPython Project

The language-agnostic parts of IPython are getting a new home in Project Jupyter



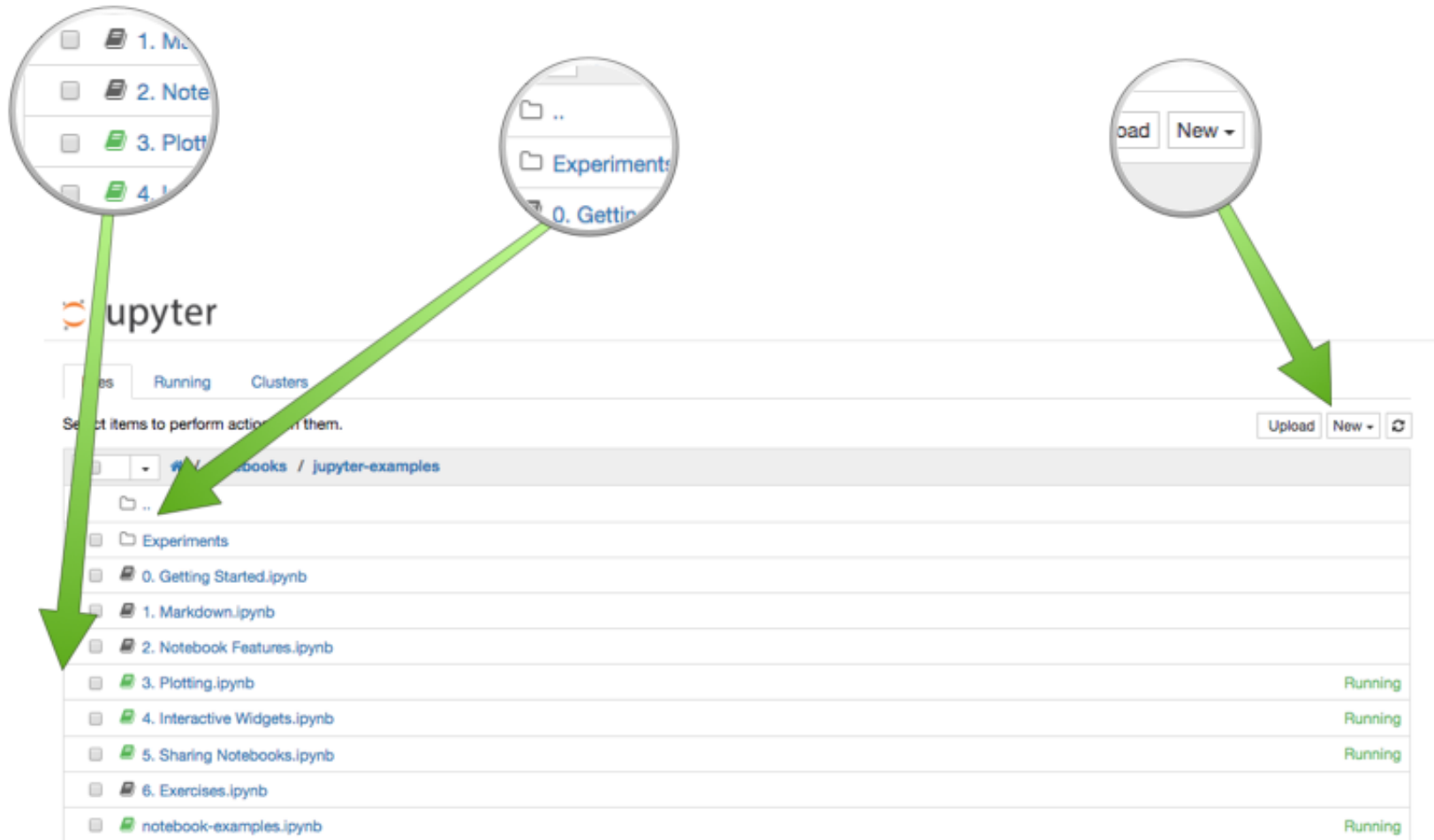
# Jupyter Web Application



- The Jupyter Web application acts as a dashboard for collections of individual notebooks.



# Jupyter Web Application



# Installing Jupyter

- The easiest way to install Jupyter notebook is with a package manager like “miniconda”
  - <http://conda.pydata.org/miniconda.html>

```
$ conda install jupyter
```

# Using the Jupyter Web Application

- Once installed, `jupyter notebook` is launched via the command line.

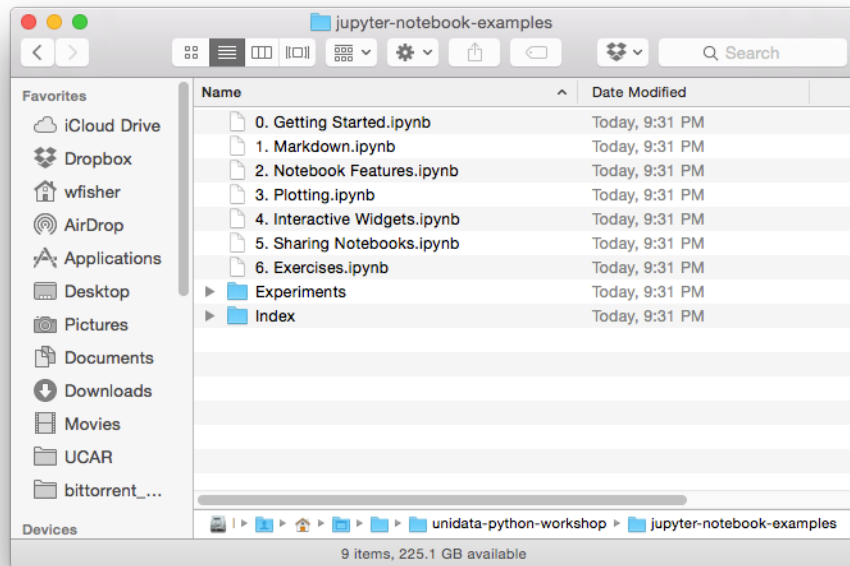
```
$ jupyter notebook
```

# Using the Jupyter Web Application

- There are a number of command-line options for advanced usage.
  - Security-related options.
  - Working directory.
  - Default behavior.
  - etc.

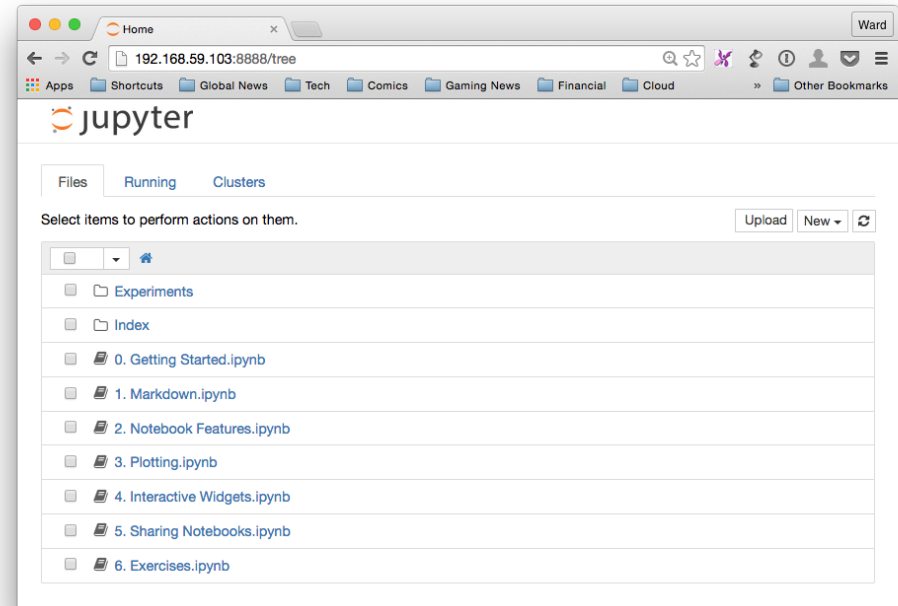
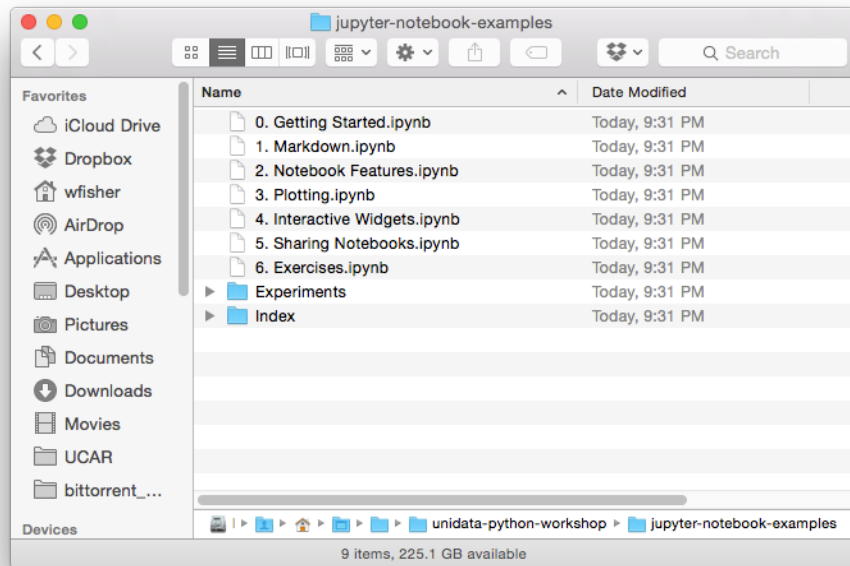
```
$ jupyter notebook [options]
```

# Using the Jupyter Web Application



- Notebooks are arranged by directory.
- Jupyter Notebook is invoked at the root of this directory structure.

# Using the Jupyter Web Application



# Using the Jupyter Web Application

Switching to the Browser.