Introduction to Jupyter Notebook

Ward Fisher

Unidata Python Workshop

July 2016

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?

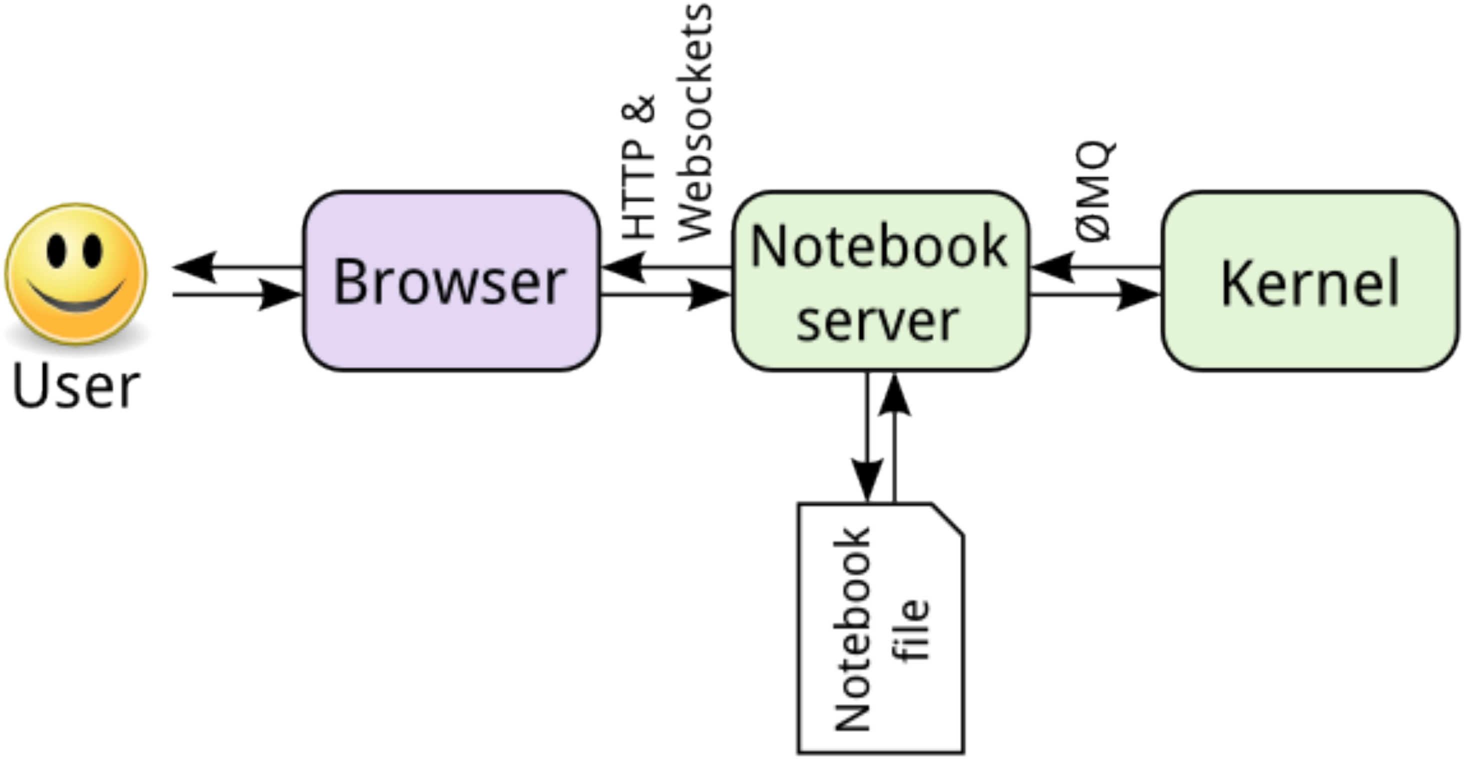
Alternatively: Jupyter Notebook is an interactive computing environment that enables users to author notebook documents that include:

- 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.

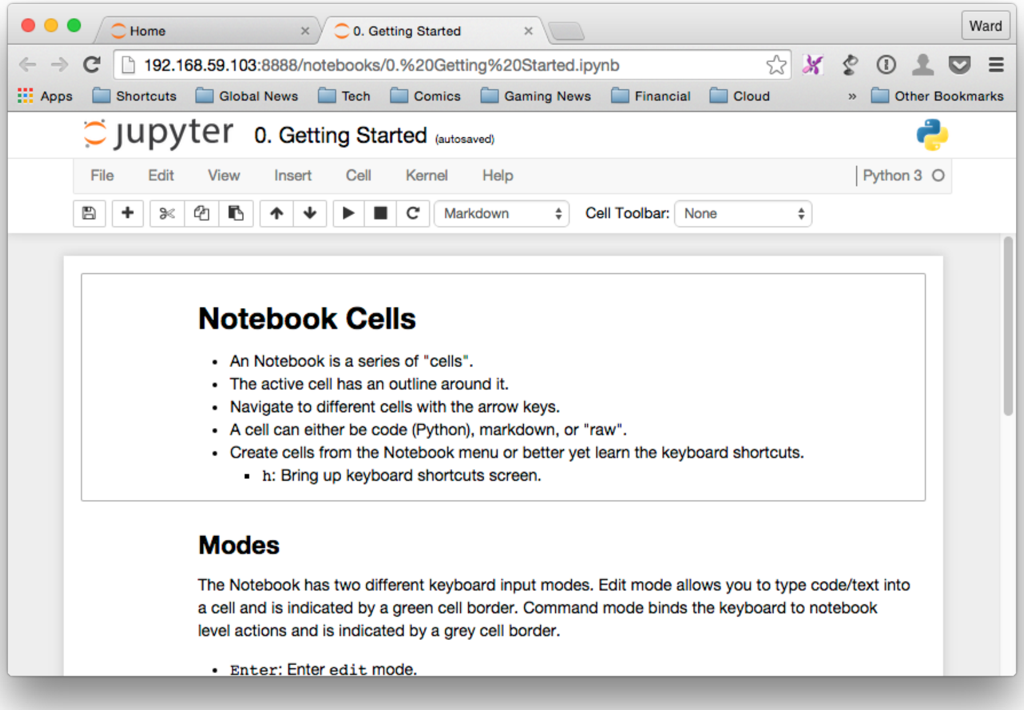
Open source, interactive data science and scientific computing across over 40 programming languages.

Jupyter Notebook Components



Jupyter Notebook document

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



Notebook Cells

- An Notebook is a series of "cells".
- The active cell has an outline around it.
- Navigate to different cells with the arrow keys.
- A cell can either be code (Python), markdown, or "raw".
- Create cells from the Notebook menu or better yet learn the keyboard shortcuts.
  - `h`: Bring up keyboard shortcuts screen.

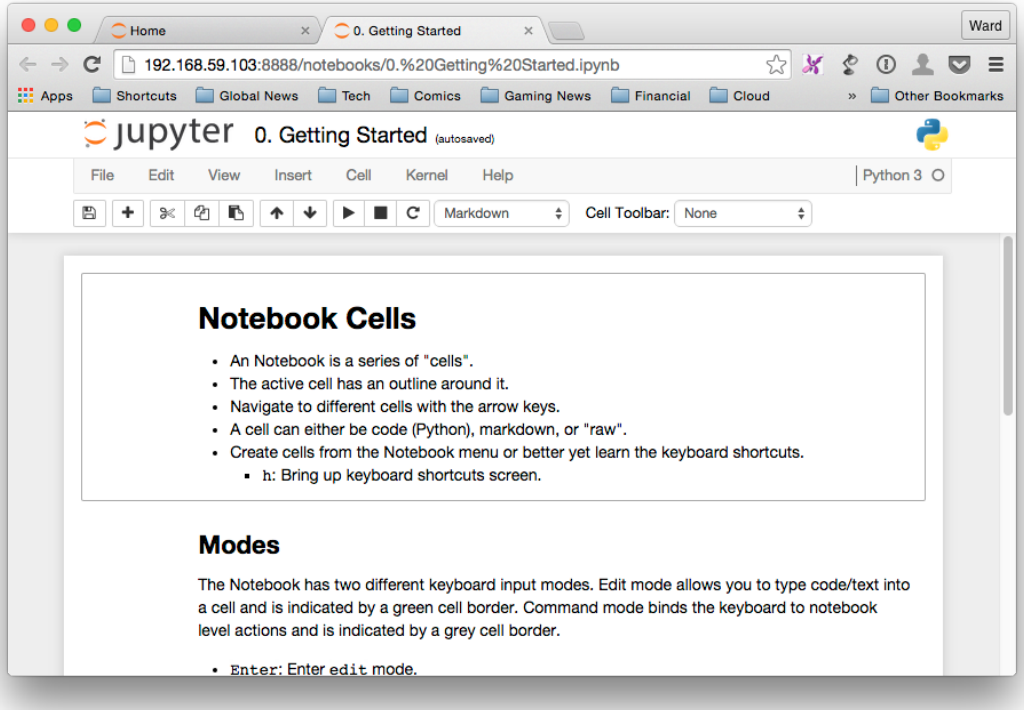
Modes

The Notebook has two different keyboard input modes. Edit mode allows you to type code/text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level actions and is indicated by a grey cell border.

- `Enter`: Enter `edit` mode.

Jupyter Notebook document

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



Notebook Cells

- An Notebook is a series of "cells".
- The active cell has an outline around it.
- Navigate to different cells with the arrow keys.
- A cell can either be code (Python), markdown, or "raw".
- Create cells from the Notebook menu or better yet learn the keyboard shortcuts.
  - `h`: Bring up keyboard shortcuts screen.

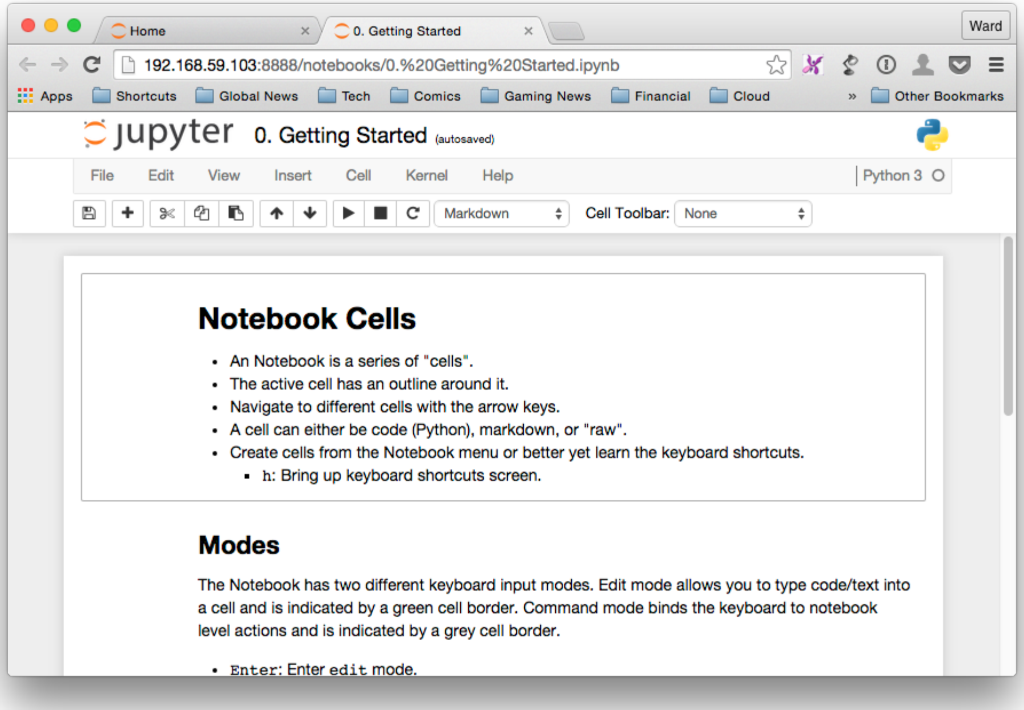
Modes

The Notebook has two different keyboard input modes. Edit mode allows you to type code/text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level actions and is indicated by a grey cell border.

- `Enter`: Enter `edit` mode.

Jupyter Notebook document

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



Notebook Cells

- An Notebook is a series of "cells".
- The active cell has an outline around it.
- Navigate to different cells with the arrow keys.
- A cell can either be code (Python), markdown, or "raw".
- Create cells from the Notebook menu or better yet learn the keyboard shortcuts.
  - `h`: Bring up keyboard shortcuts screen.

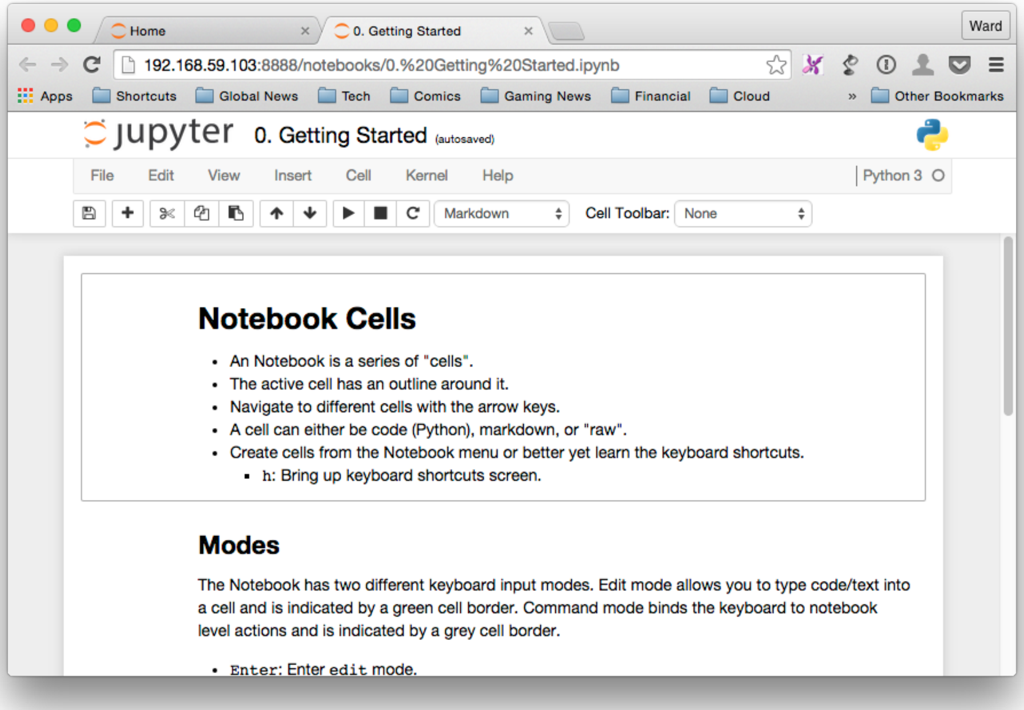
Modes

The Notebook has two different keyboard input modes. Edit mode allows you to type code/text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level actions and is indicated by a grey cell border.

- `Enter`: Enter `edit` mode.

Jupyter Notebook document

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



Notebook Cells

- An Notebook is a series of "cells".
- The active cell has an outline around it.
- Navigate to different cells with the arrow keys.
- A cell can either be code (Python), markdown, or "raw".
- Create cells from the Notebook menu or better yet learn the keyboard shortcuts.
  - `h`: Bring up keyboard shortcuts screen.

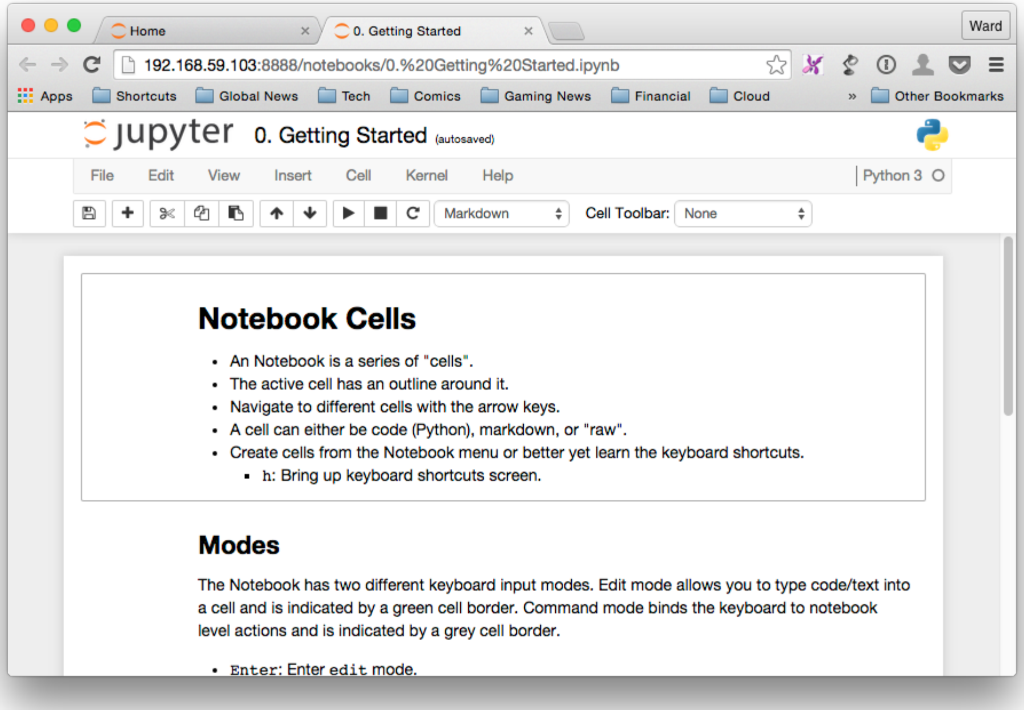
Modes

The Notebook has two different keyboard input modes. Edit mode allows you to type code/text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level actions and is indicated by a grey cell border.

- `Enter`: Enter `edit` mode.

Jupyter Notebook document

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



Notebook Cells

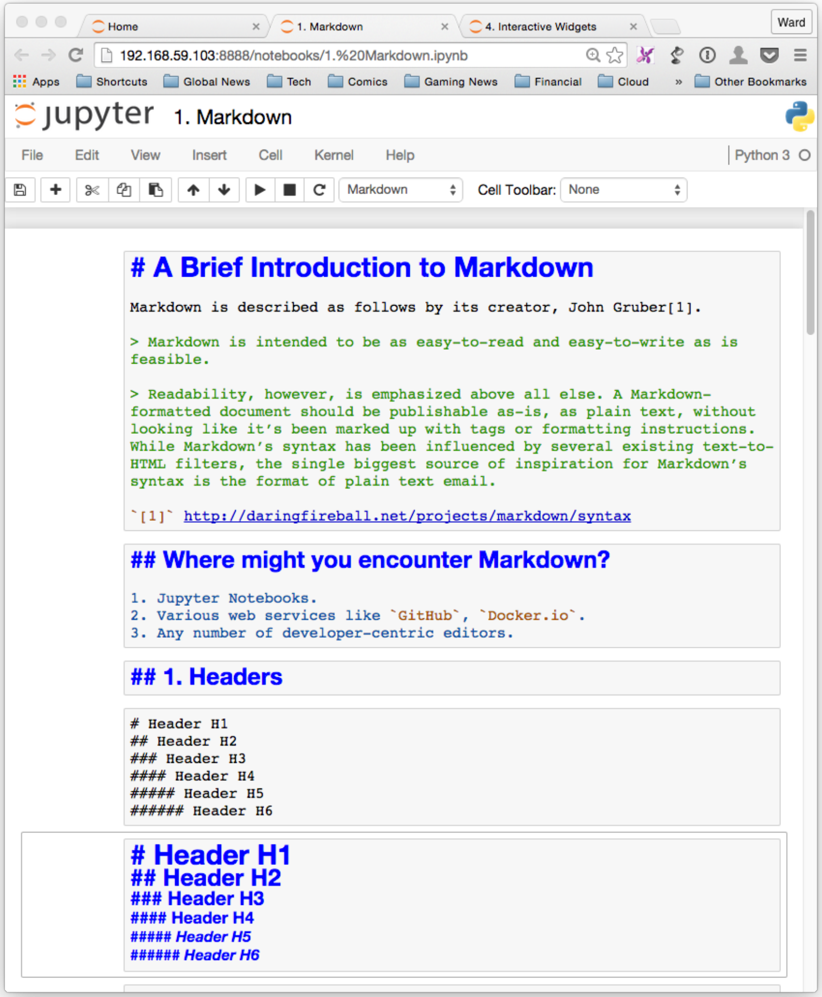
- An Notebook is a series of "cells".
- The active cell has an outline around it.
- Navigate to different cells with the arrow keys.
- A cell can either be code (Python), markdown, or "raw".
- Create cells from the Notebook menu or better yet learn the keyboard shortcuts.
  - `h`: Bring up keyboard shortcuts screen.

Modes

The Notebook has two different keyboard input modes. Edit mode allows you to type code/text into a cell and is indicated by a green cell border. Command mode binds the keyboard to notebook level actions and is indicated by a grey cell border.

- `Enter`: Enter `edit` mode.

Markdown Cells



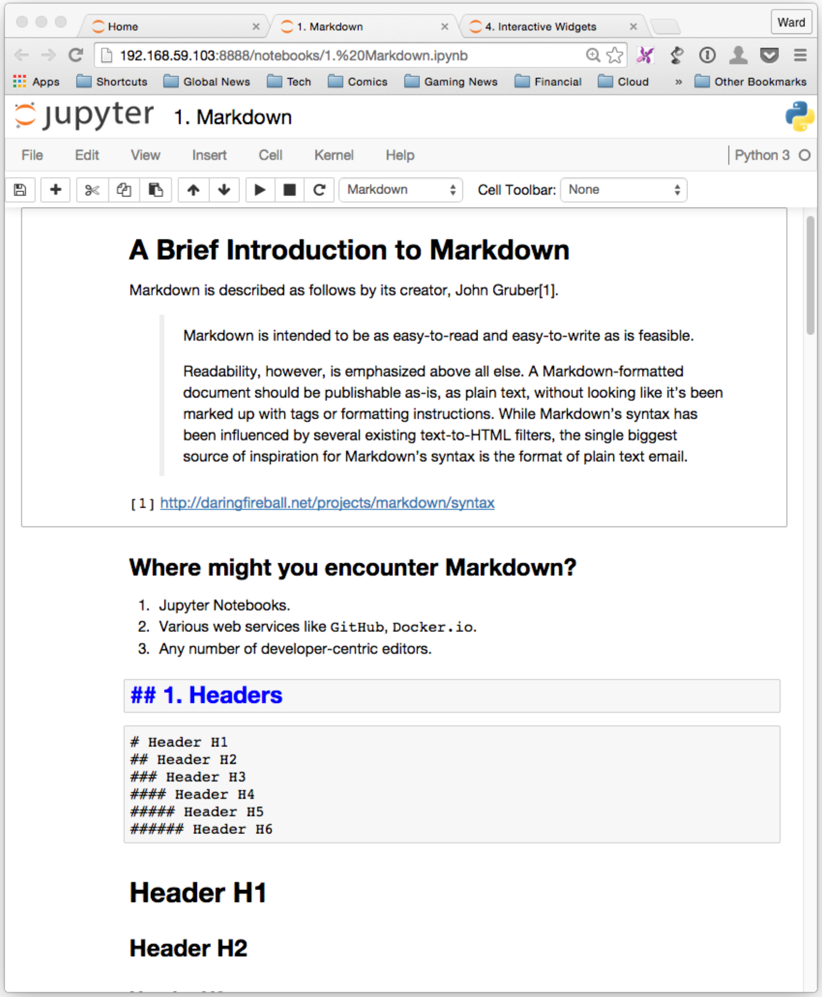
```
# A Brief Introduction to Markdown

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
```



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.

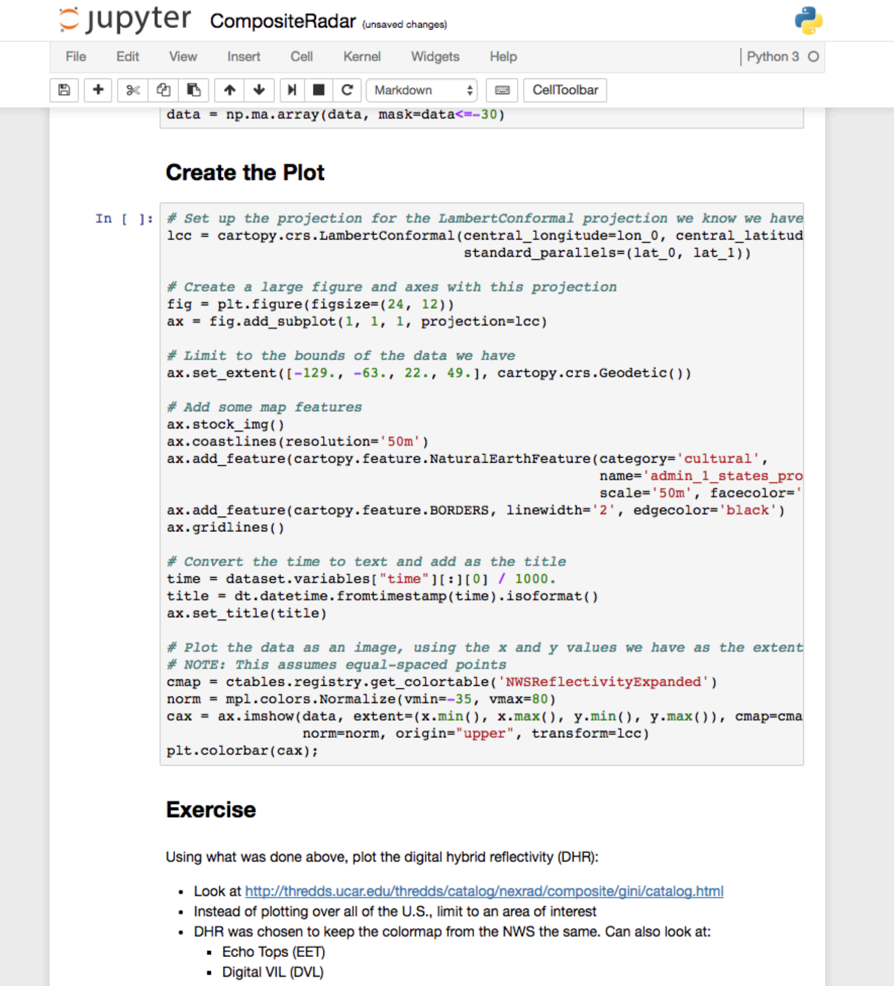
```
## 1. Headers
```

```
# 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)
```

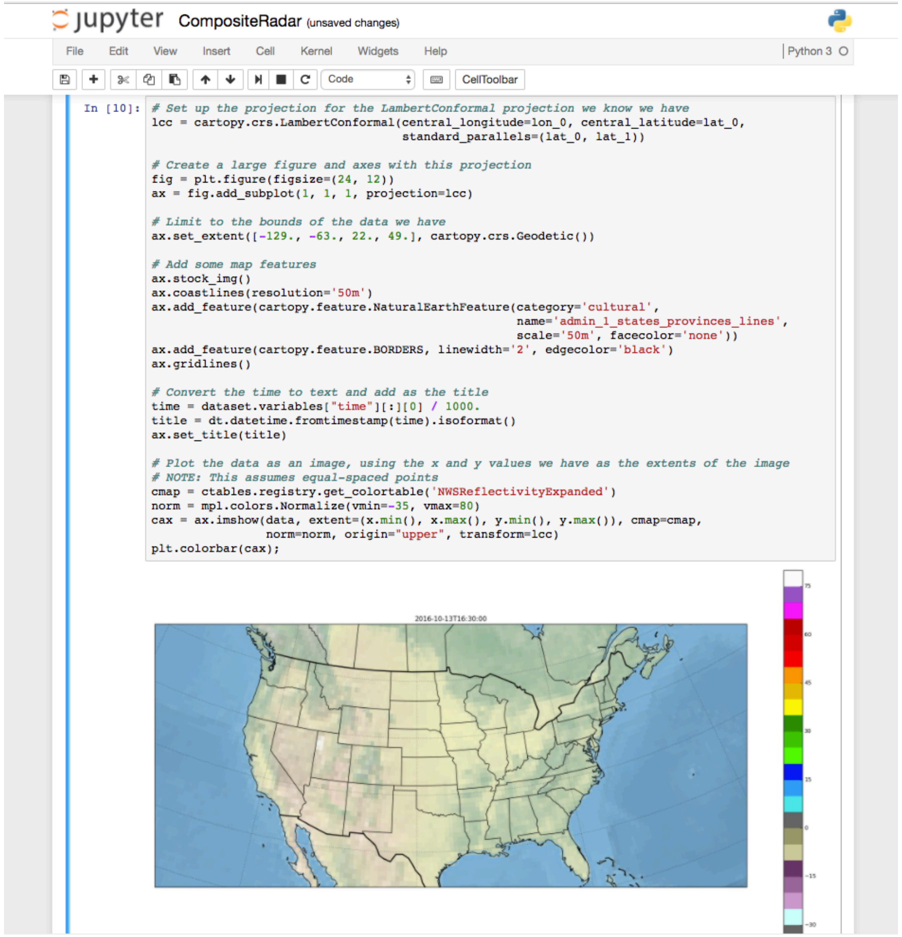
Create the Plot

```
In [ ]: # Set up the projection for the LambertConformal projection we know we have
        lcc = cartopy.crs.LambertConformal(central_longitude=lon_0, central_latitud
                                           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_pro
                                                           scale='50m', facecolor='
        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=cma
                        norm=norm, origin="upper", transform=lcc)
        plt.colorbar(cax);
```

Exercise

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

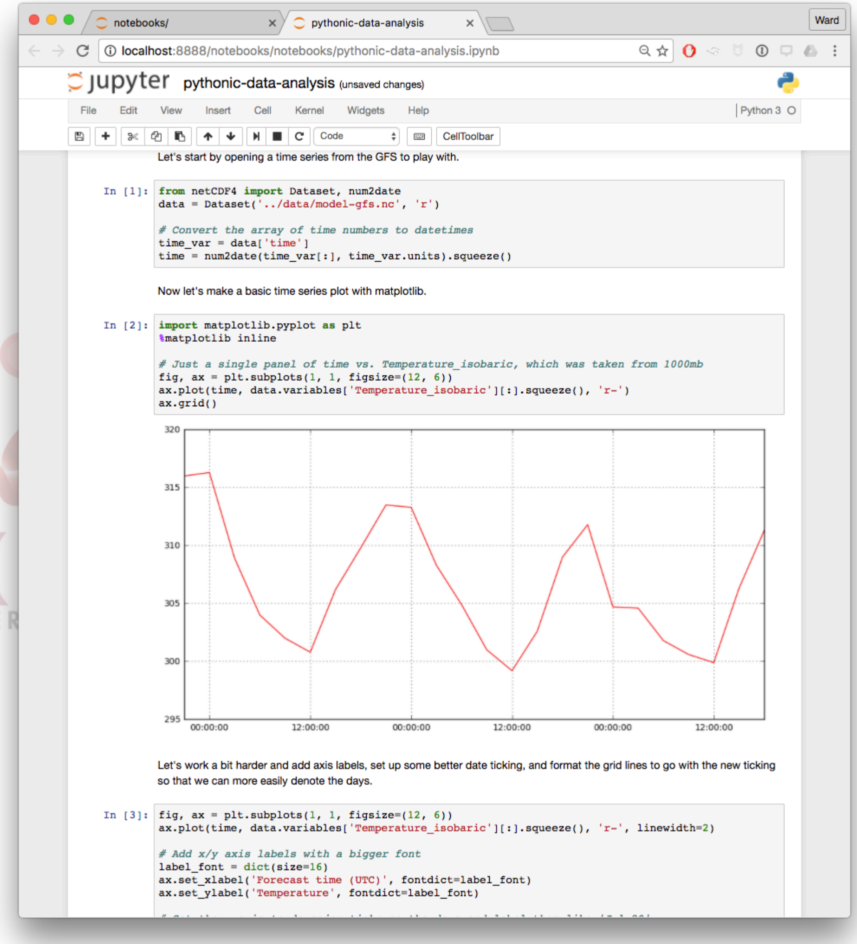
- Look at http://thredds.ucar.edu/thredds/catalog/nexrad/composite/gini/catalog.html
- Instead of plotting over all of the U.S., limit to an area of interest
- DHR was chosen to keep the colormap from the NWS the same. Can also look at:
    - Echo Tops (EET)
    - Digital VIL (DVL)



```
In [10]: # 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 extents of the image
         # 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);
```

What does this get you?

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



Let's start by opening a time series from the GFS to play with.

```
In [1]: from netCDF4 import Dataset, num2date
        data = Dataset('../data/model-gfs.nc', 'r')

        # Convert the array of time numbers to datetimes
        time_var = data['time']
        time = num2date(time_var[:], time_var.units).squeeze()
```

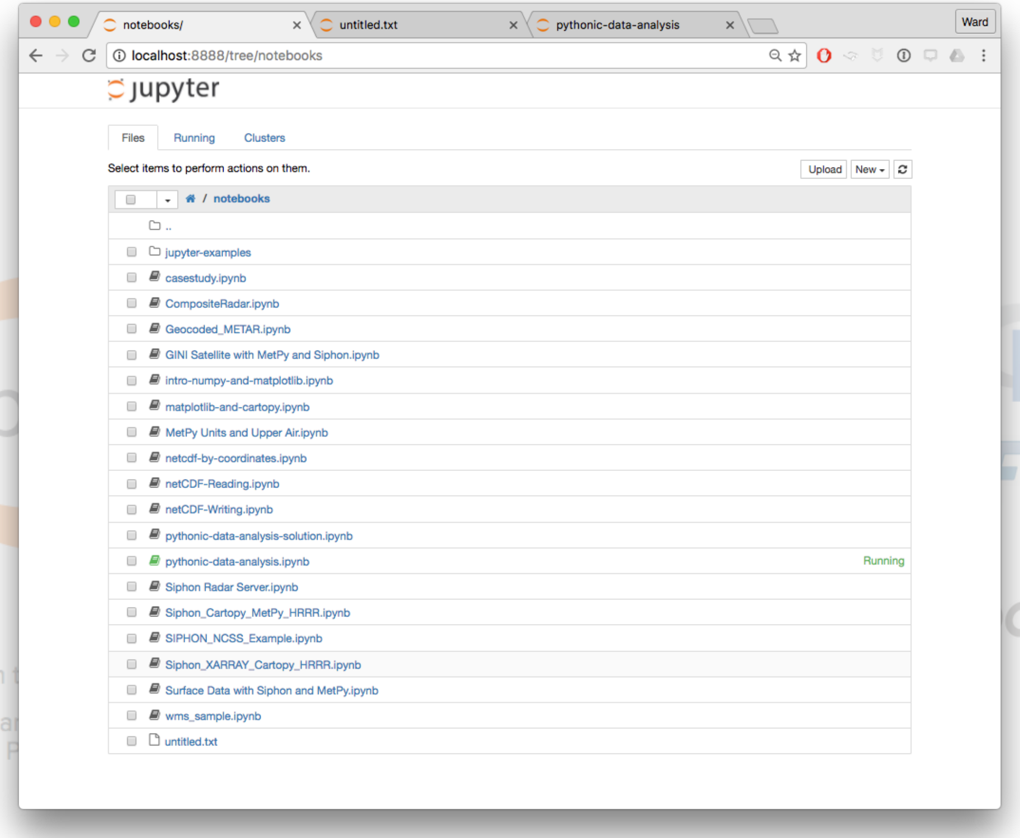
Now let's make a basic time series plot with matplotlib.

```
In [2]: import matplotlib.pyplot as plt
        %matplotlib inline

        # Just a single panel of time vs. Temperature_isobaric, which was taken from 1000mb
        fig, ax = plt.subplots(1, 1, figsize=(12, 6))
        ax.plot(time, data.variables['Temperature_isobaric'][:].squeeze(), 'r-')
        ax.grid()
```

Let's work a bit harder and add axis labels, set up some better date ticking, and format the grid lines to go with the new ticking so that we can more easily denote the days.

```
In [3]: fig, ax = plt.subplots(1, 1, figsize=(12, 6))
        ax.plot(time, data.variables['Temperature_isobaric'][:].squeeze(), 'r-', linewidth=2)

        # Add x/y axis labels with a bigger font
        label_font = dict(size=16)
        ax.set_xlabel('Forecast time (UTC)', fontdict=label_font)
        ax.set_ylabel('Temperature', fontdict=label_font)
```

Jupyter Web Application



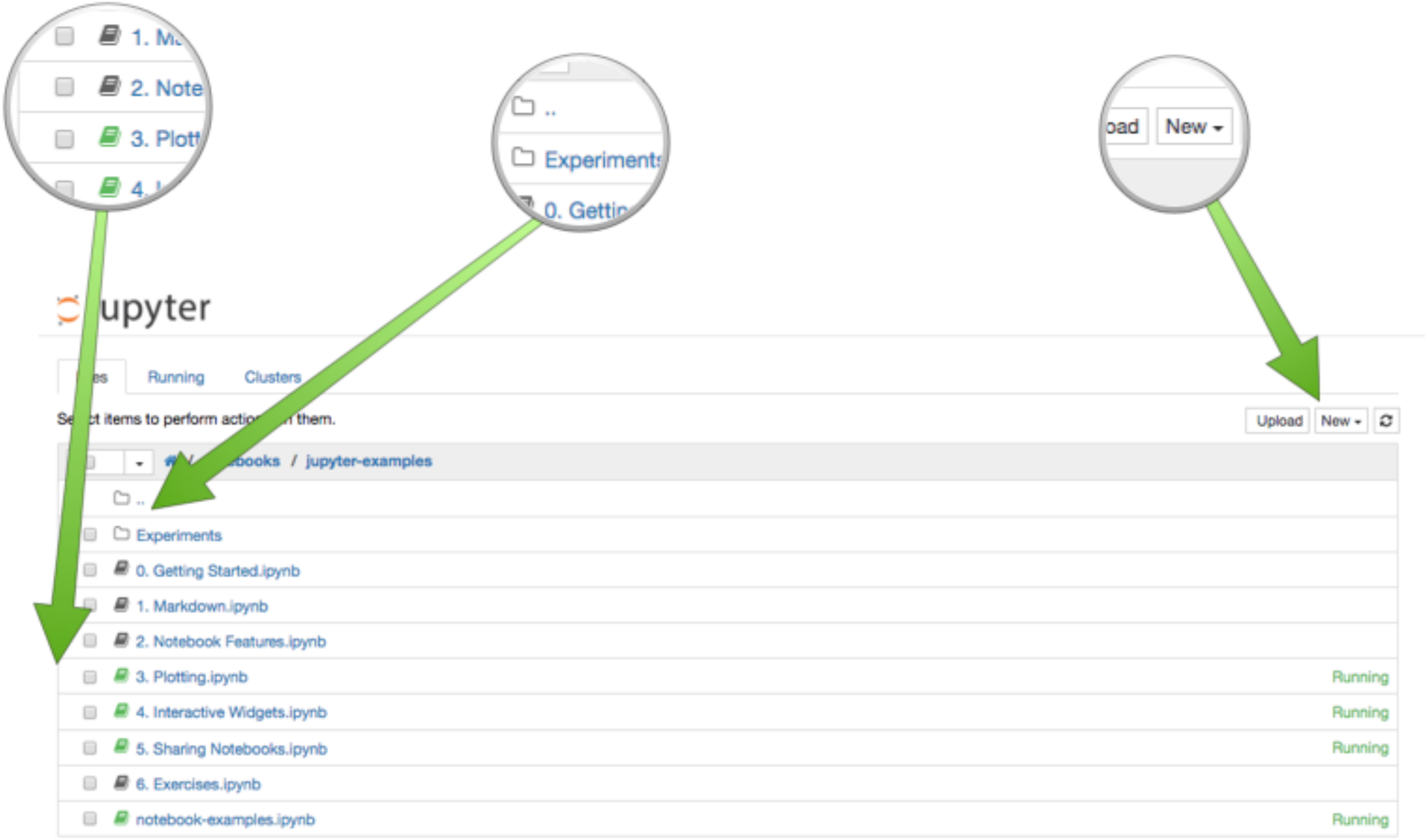
Select items to perform actions on them.

/ notebooks

- ..
- jupyter-examples
- casestudy.ipynb
- CompositeRadar.ipynb
- Geocoded_METAR.ipynb
- GINI Satellite with MetPy and Siphon.ipynb
- intro-numpy-and-matplotlib.ipynb
- matplotlib-and-cartopy.ipynb
- MetPy Units and Upper Air.ipynb
- netcdf-by-coordinates.ipynb
- netCDF-Reading.ipynb
- netCDF-Writing.ipynb
- pythonic-data-analysis-solution.ipynb
- pythonic-data-analysis.ipynb Running
- Siphon Radar Server.ipynb
- Siphon_Cartopy_MetPy_HRRR.ipynb
- SIPHON_NCSS_Example.ipynb
- Siphon_XARRAY_Cartopy_HRRR.ipynb
- Surface Data with Siphon and MetPy.ipynb
- wms_sample.ipynb
- untitled.txt

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

Jupyter Web Application



jupyter

Files Running Clusters

Select items to perform actions on them.

Upload New

# / notebooks / jupyter-examples	
..	
□ Experiments	
□ 0. Getting Started.ipynb	
□ 1. Markdown.ipynb	
□ 2. Notebook Features.ipynb	
□ 3. Plotting.ipynb	Running
□ 4. Interactive Widgets.ipynb	Running
□ 5. Sharing Notebooks.ipynb	Running
□ 6. Exercises.ipynb	
□ notebook-examples.ipynb	Running

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 is launched via the command line.

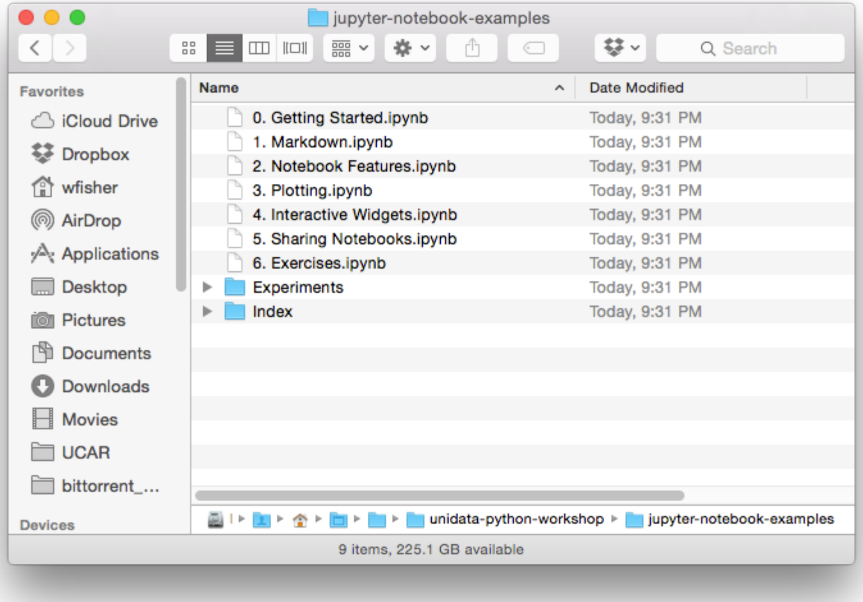
```
$ jupyter notebook
```

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

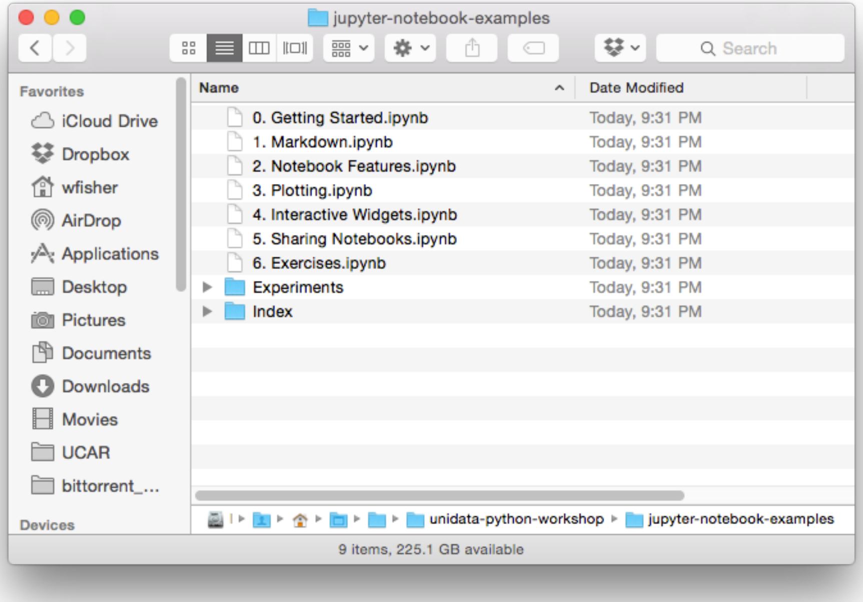


Name	Date Modified
0. Getting Started.ipynb	Today, 9:31 PM
1. Markdown.ipynb	Today, 9:31 PM
2. Notebook Features.ipynb	Today, 9:31 PM
3. Plotting.ipynb	Today, 9:31 PM
4. Interactive Widgets.ipynb	Today, 9:31 PM
5. Sharing Notebooks.ipynb	Today, 9:31 PM
6. Exercises.ipynb	Today, 9:31 PM
▶ Experiments	Today, 9:31 PM
▶ Index	Today, 9:31 PM

9 items, 225.1 GB available

- Notebooks are arranged in a directory.
- You launch Jupyter Notebook from the root of this directory structure.

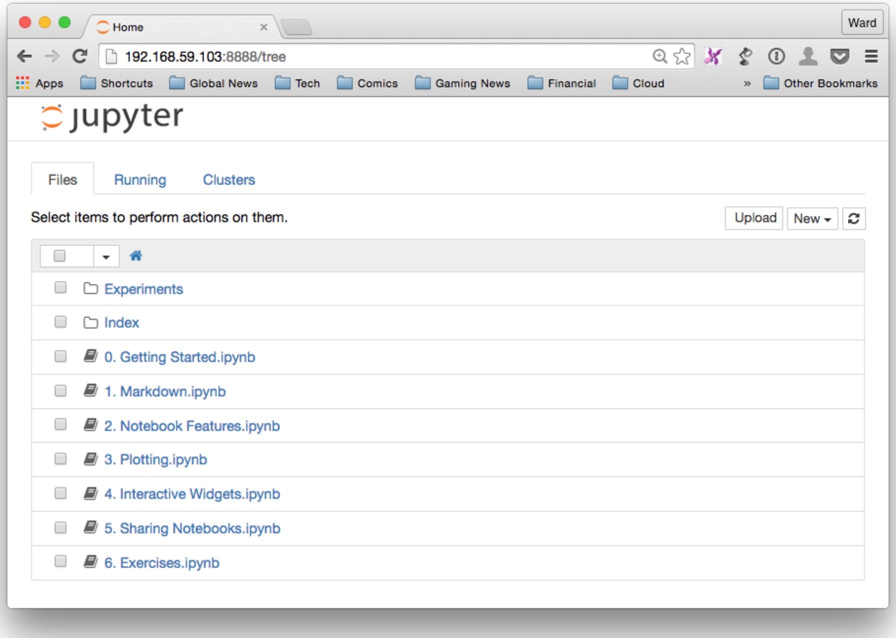
Using the Jupyter Web Application



Name	Date Modified
0. Getting Started.ipynb	Today, 9:31 PM
1. Markdown.ipynb	Today, 9:31 PM
2. Notebook Features.ipynb	Today, 9:31 PM
3. Plotting.ipynb	Today, 9:31 PM
4. Interactive Widgets.ipynb	Today, 9:31 PM
5. Sharing Notebooks.ipynb	Today, 9:31 PM
6. Exercises.ipynb	Today, 9:31 PM
Experiments	Today, 9:31 PM
Index	Today, 9:31 PM

unidata-python-workshop ▸ jupyter-notebook-examples

9 items, 225.1 GB available



192.168.59.103:8888/tree

Select items to perform actions on them.

- □ Experiments
- □ Index
- □ 0. Getting Started.ipynb
- □ 1. Markdown.ipynb
- □ 2. Notebook Features.ipynb
- □ 3. Plotting.ipynb
- □ 4. Interactive Widgets.ipynb
- □ 5. Sharing Notebooks.ipynb
- □ 6. Exercises.ipynb

Using the Jupyter Web Application

Switching to the Browser.