

Python: Its Past, Present, and Future in Meteorology

7th Symposium on Advances in
Modeling and Analysis Using Python
23 January 2016 | Seattle, WA

Ryan May (@dopplershift)
UCAR/Unidata

Outline

- The Past
 - What is Python?
 - A History Lesson
- The Present
 - Recent events
 - Today's Python Landscape
- The Future
 - What you should keep an eye on
 - What's coming down the pipe
 - ~~Ryan predicts the weather~~

A long time ago in an office
far, far away...

What is Python?

- Dynamic, strongly-typed language
 - Automatic memory management
 - Exception handling
 - “Fits your brain”
 - Large standard library (batteries included)
 - **Interpreted (i.e. slow)**
- Free and open-source
- **Quick to develop**
- “Glue language”

Zen of Python (`import this`)

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than **right** now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

A Brief History of Python

Python started by Guido van Rossum

Python Software Foundation created

Python 1.0 Released

Python 2.0 Released

1990

1992

1994

1996

1998

2000

2002

First PyCon Held

Python 2.7 Released

Python 3.0 Released

Python 3.6 Released

2004

2006

2008

2010

2012

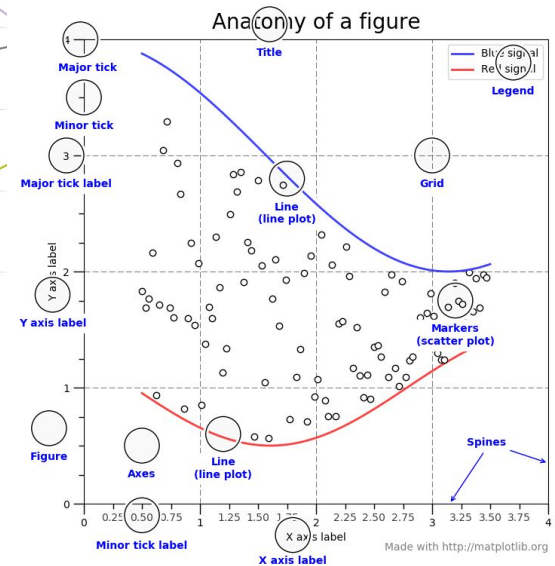
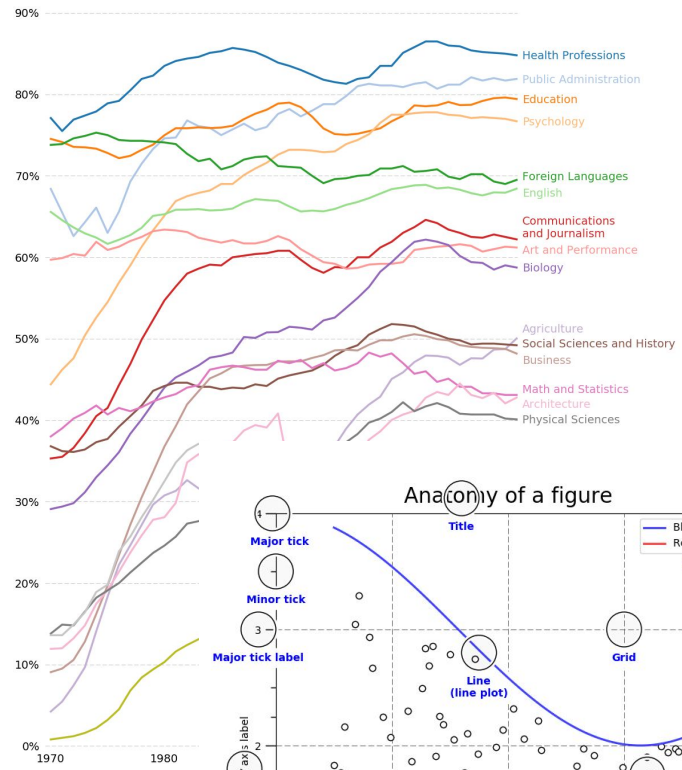
2014

2016

Scientific Python: Matplotlib

- Matplotlib
 - Provide MATLAB-like plotting interface
 - Works with wide array of GUI interfaces
 - Produces publication-quality figures

Percentage of Bachelor's degrees conferred to women in the U.S.A. by major (1970-2011)

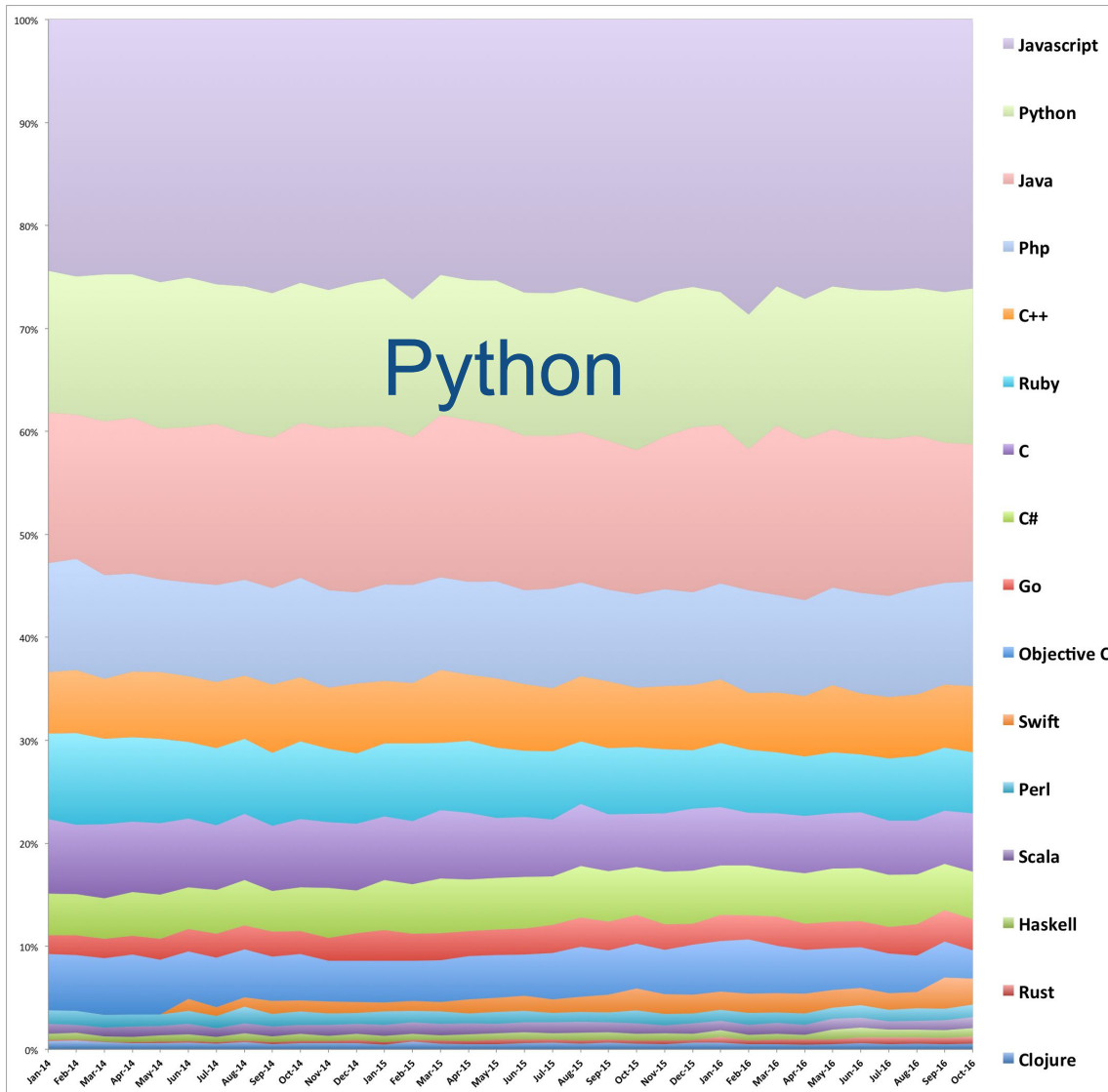


Scientific Python: The Core

- NumPy: Fast mathematical arrays
 - Array math
 - Avoid python loops
- SciPy: interfaces to general algorithms
 - Integration, linear algebra, FFT
 - Spatial algorithms, random numbers, stats
- IPython: interactive analysis sandbox
 - Interactive plotting with matplotlib
 - Various improvements for working with data interactively

Where do we stand today?

Second Most Popular?



Based on active GitHub repositories. (October 2016)

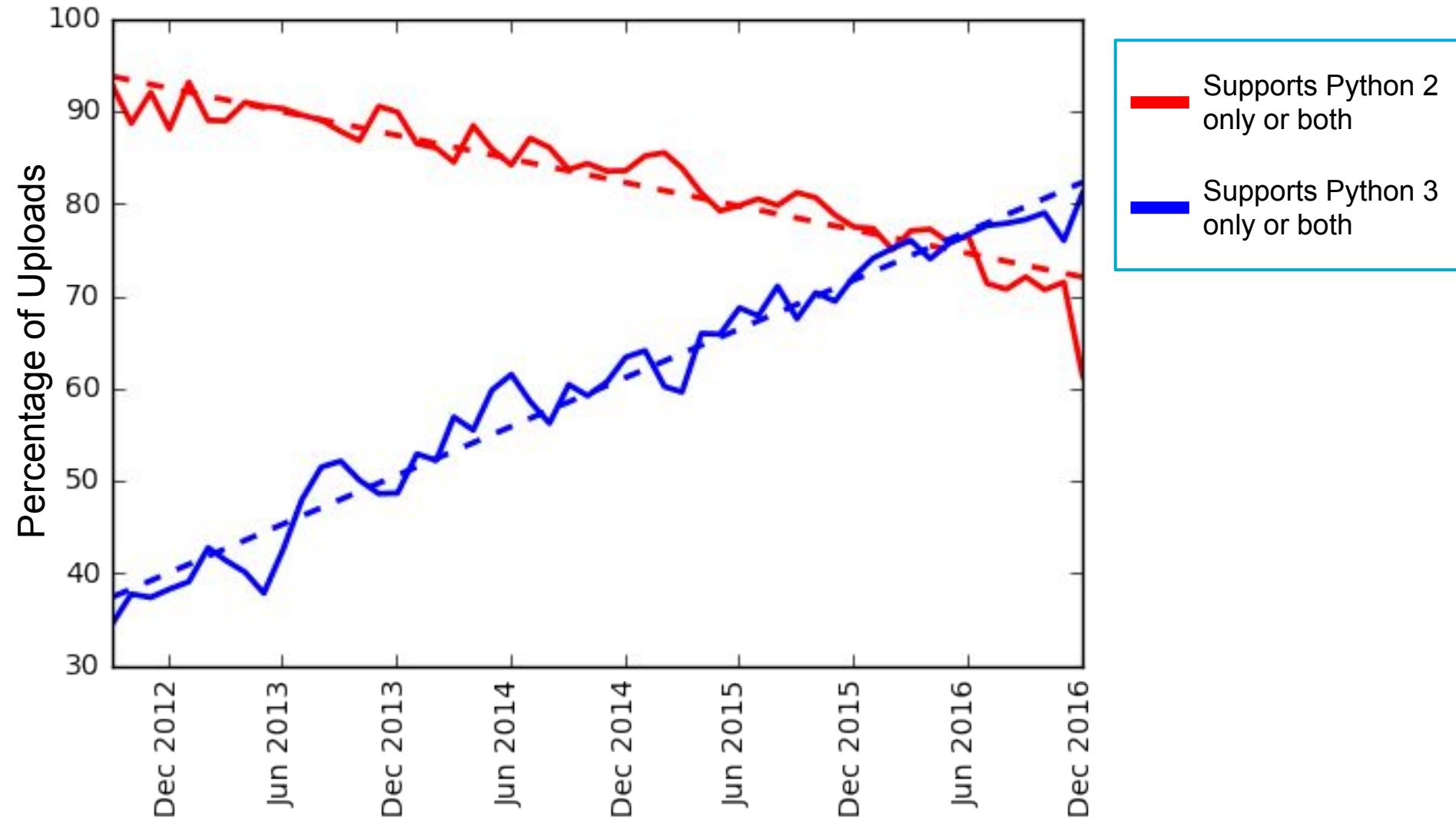
Python is also #5 in the Tiobe programming language index.

<http://www.tiobe.com/tiobe-index>

Python: Healthy as Ever

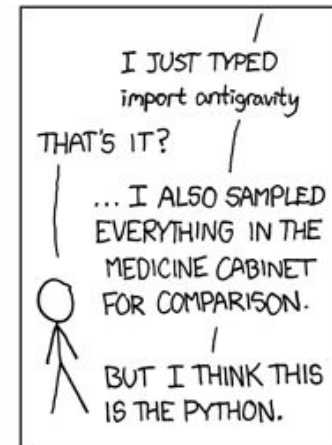
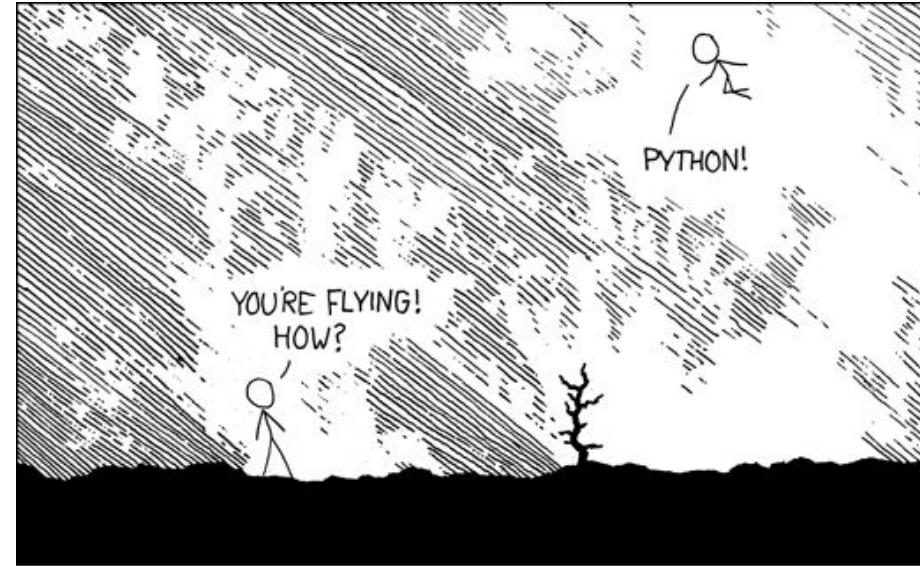
- Growing migration from Python 2 to 3
- End-of-life for 2.7 announced for 2020--Jupyter/IPython, Matplotlib, Pandas all following suit
- 3.6 released 23 December 2016
 - Many small additions
 - Some performance optimization
 - 3.6 up to 20% faster than 2.7

Python Upload Statistics



What About for Science?

- Numpy 1.12
 - released 15 January
- Matplotlib 2.0
 - released 16 January
 - **Tutorial Tuesday**
- SciPy 0.18.1
 - Released 19 Sept.
- IPython has grown into Jupyter



<https://xkcd.com/353/>

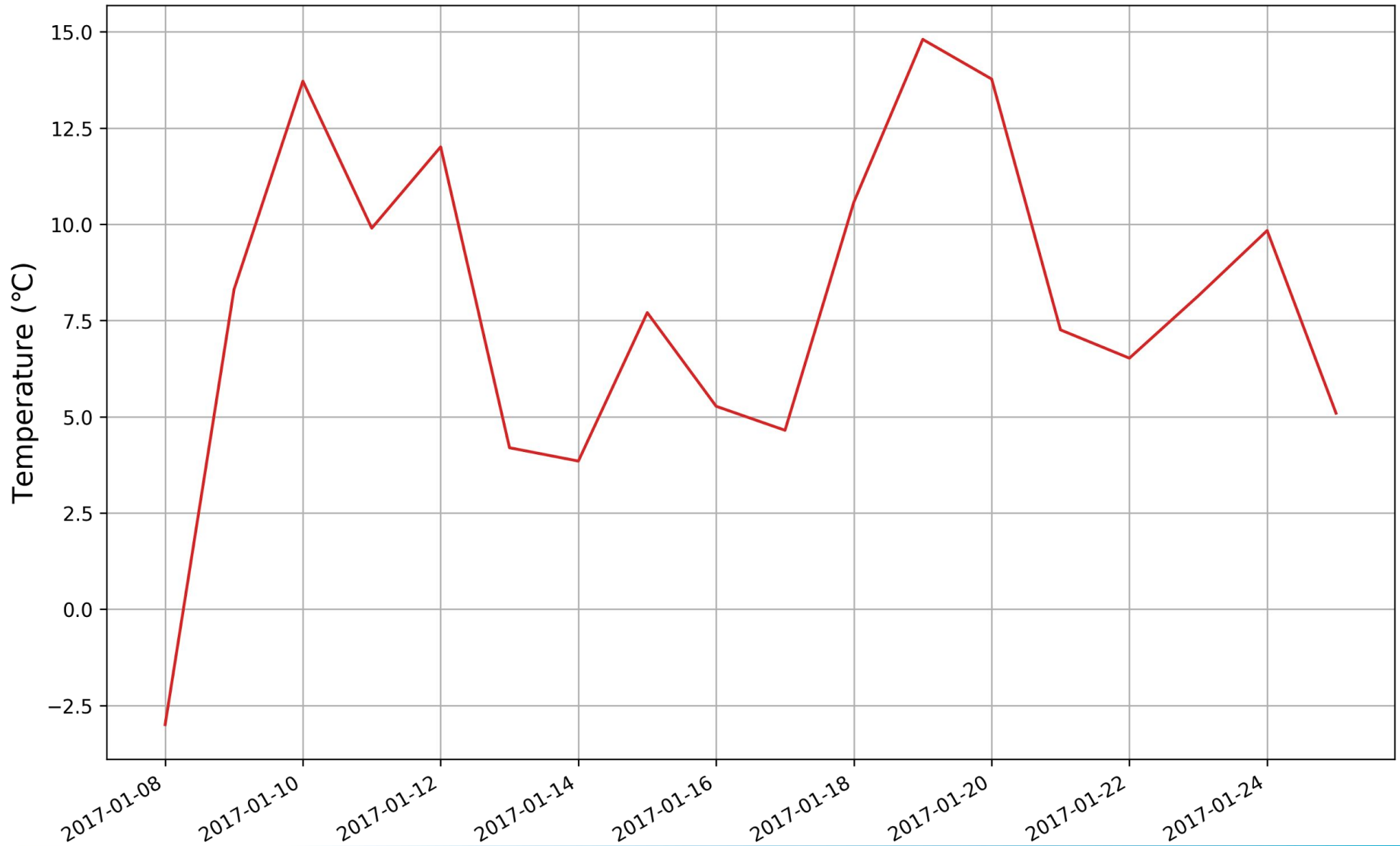
What's It Like?

```
import matplotlib.pyplot as plt
import netCDF4

# Read netCDF file
ds = netCDF4.Dataset('ndfd.nc')
time_var = ds['time']
time = netCDF4.num2date(time_var[:], time_var.units)
temps = ds['Temperature'][:].squeeze()

# Plot time series
fig, ax = plt.subplots(figsize=(12, 8), dpi=300)
ax.plot(time.squeeze(), temps - 273, 'tab:red')
ax.grid()
ax.set_ylabel('Temperature (\N{DEGREE CELSIUS})',
              fontsize='x-large')
fig.autofmt_xdate()
```

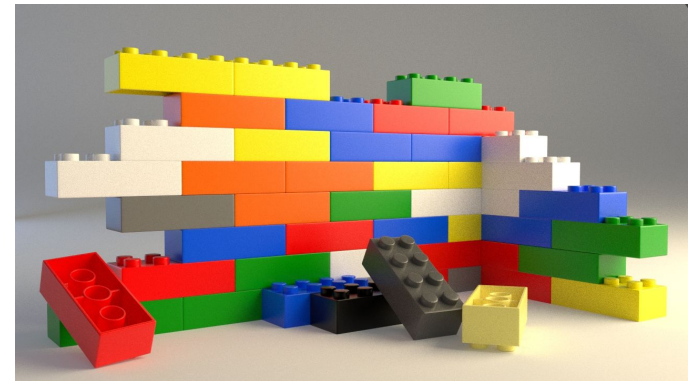
The Result



Tools That Fit Together

- **NumPy, Matplotlib, SciPy, IPython**
- **CartoPy**: Maps to go with Matplotlib
- **Pandas & Xarray**: Array data structures
- **Sympy**: Symbolic mathematics
- **Scikit-learn**: Machine learning
- **Scikit-image**: Image processing
- **Cython/f2py**: Optimize and link to compiled code

And many, many more!



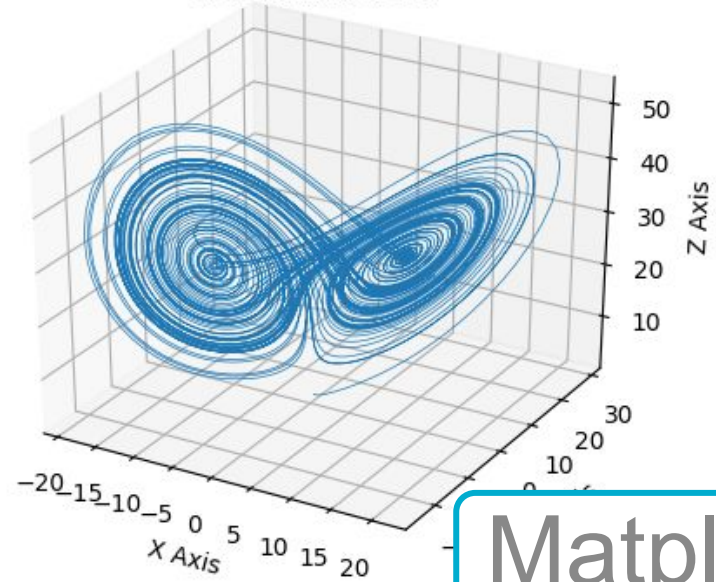
Examples

Sobel filter computed on the converted grayscale image

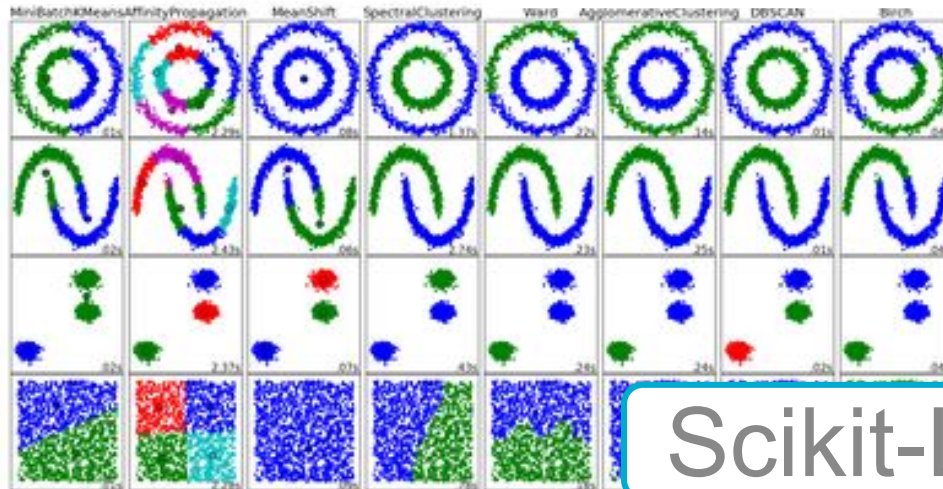


Scikit-Image

Lorenz Attractor

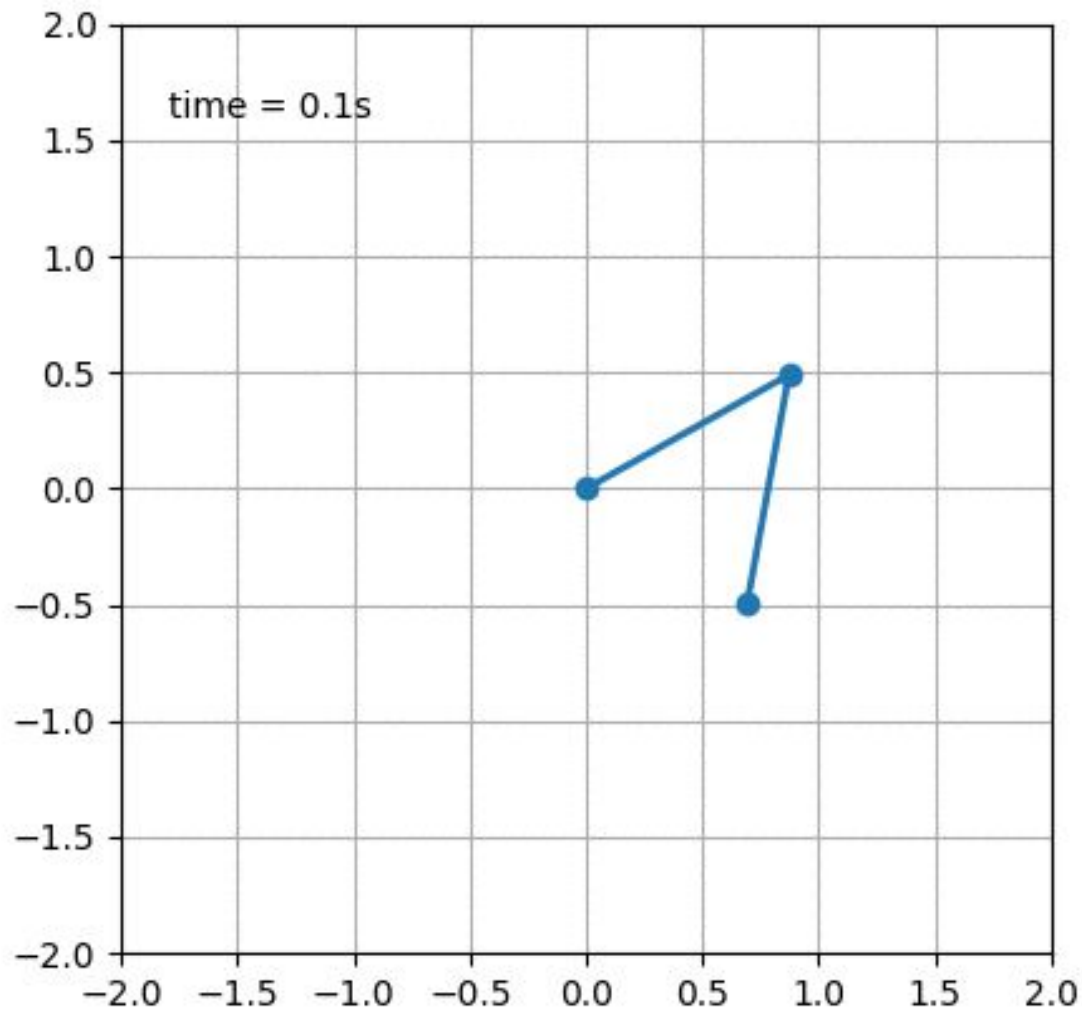


Matplotlib



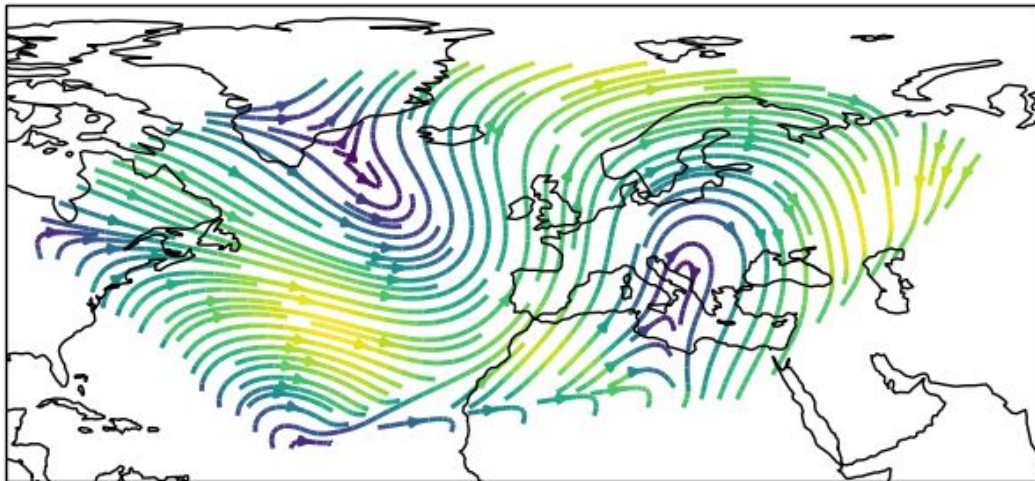
Scikit-Learn

The Double Pendulum System

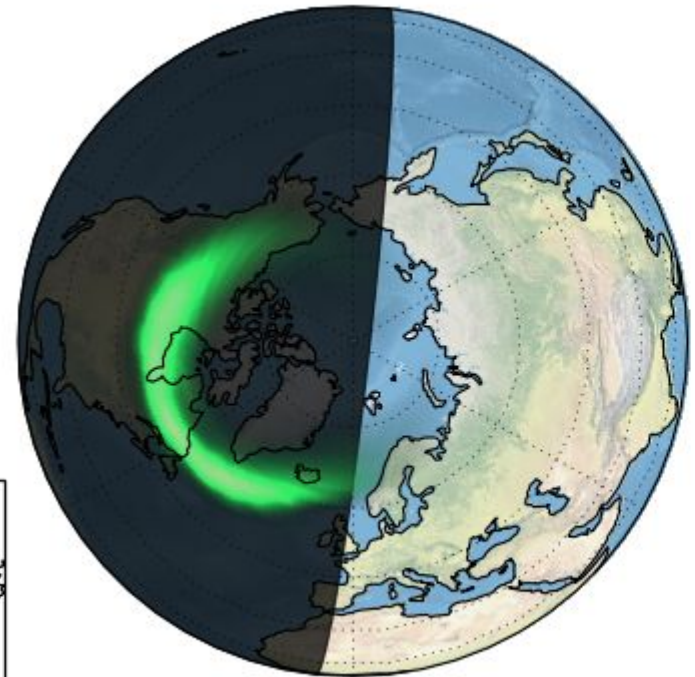


Examples (CartoPy)

US States which intersect the track of Hurricane Katrina (2005)



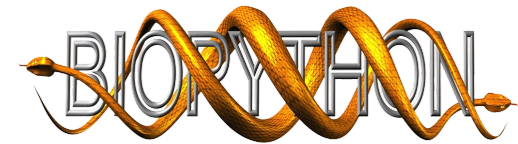
NOAA Aurora Forecast



Others Using Python

- Domains
 - Quantitative Finance
 - Astronomy
 - LIGO gravitational wave experiment
 - Oceanography
 - Genomics
- Web Applications
- Scientific Python consulting
- Many, many others

Bloomberg



YouTube



CONTINUUM[®]
ANALYTICS

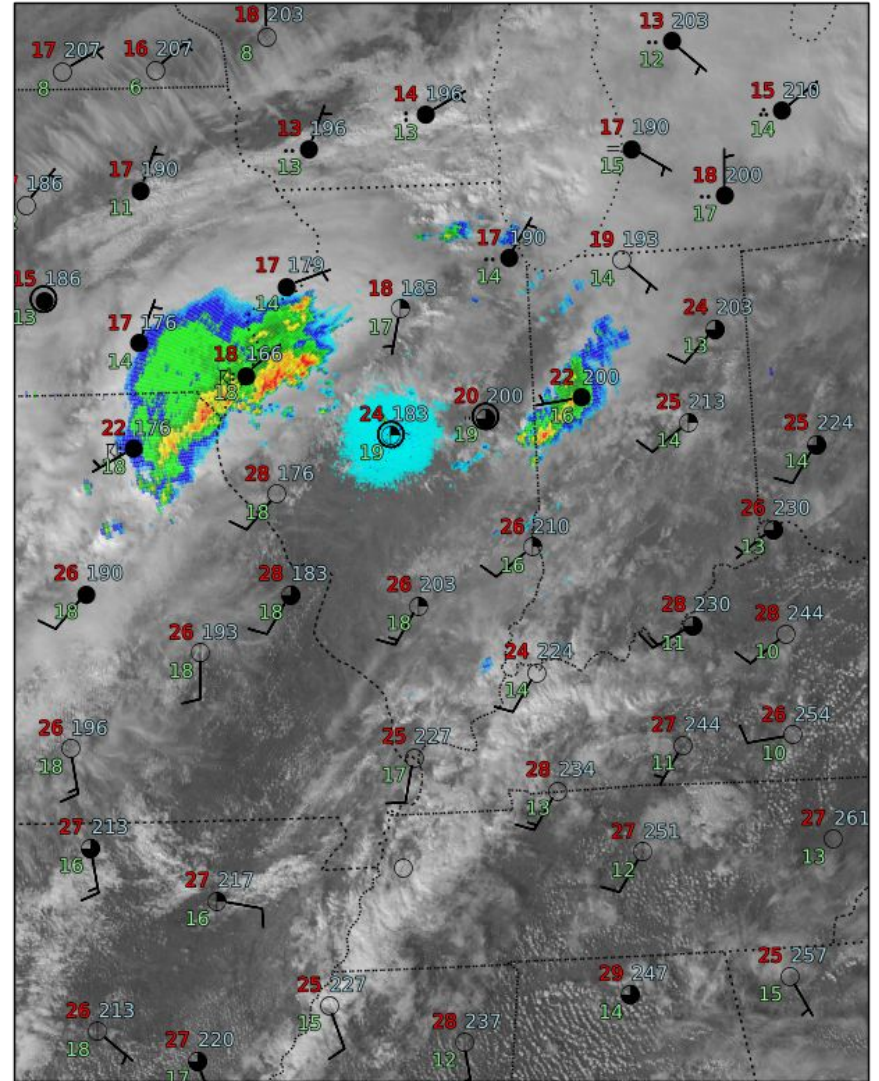
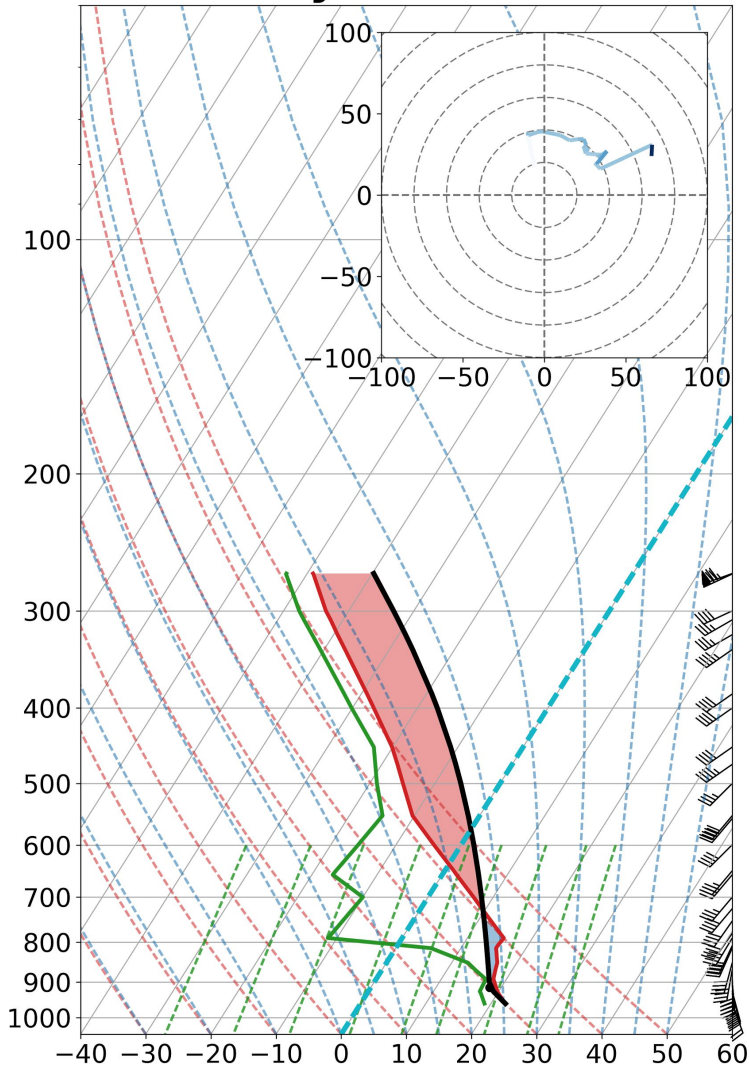
ENTHOUGHT
SCIENTIFIC COMPUTING SOLUTIONS

And Meteorology!

- **MetPy**: General metr. tools (GEMPAK)
- **Siphon**: THREDDS Python API
- **netcdf4-Python**: netCDF reader/writer
- **Pyart**: ARM Radar Toolkit
- **SHARPy**: NSHARP ported to Python
- **Pygrib**: GRIB format reader
- **PyNIO/PyNGL**: NCL wrappers
- **UV-CDAT**: Ultrascale Visualization
Climate Data Analysis Tools
- Aospy, windrose, windspharm,

Examples (MetPy)

04 May 1999 0000Z



It's Not Just Pretty Pictures

My projects that have used Python:

- Radar simulation
- Feeding real-time NEXRAD data into Amazon S3 for NOAA Big Data Project
<https://aws.amazon.com/noaa-big-data/nexrad/>
- Mass data conversion
- Website scraping
- Algorithm prototypes

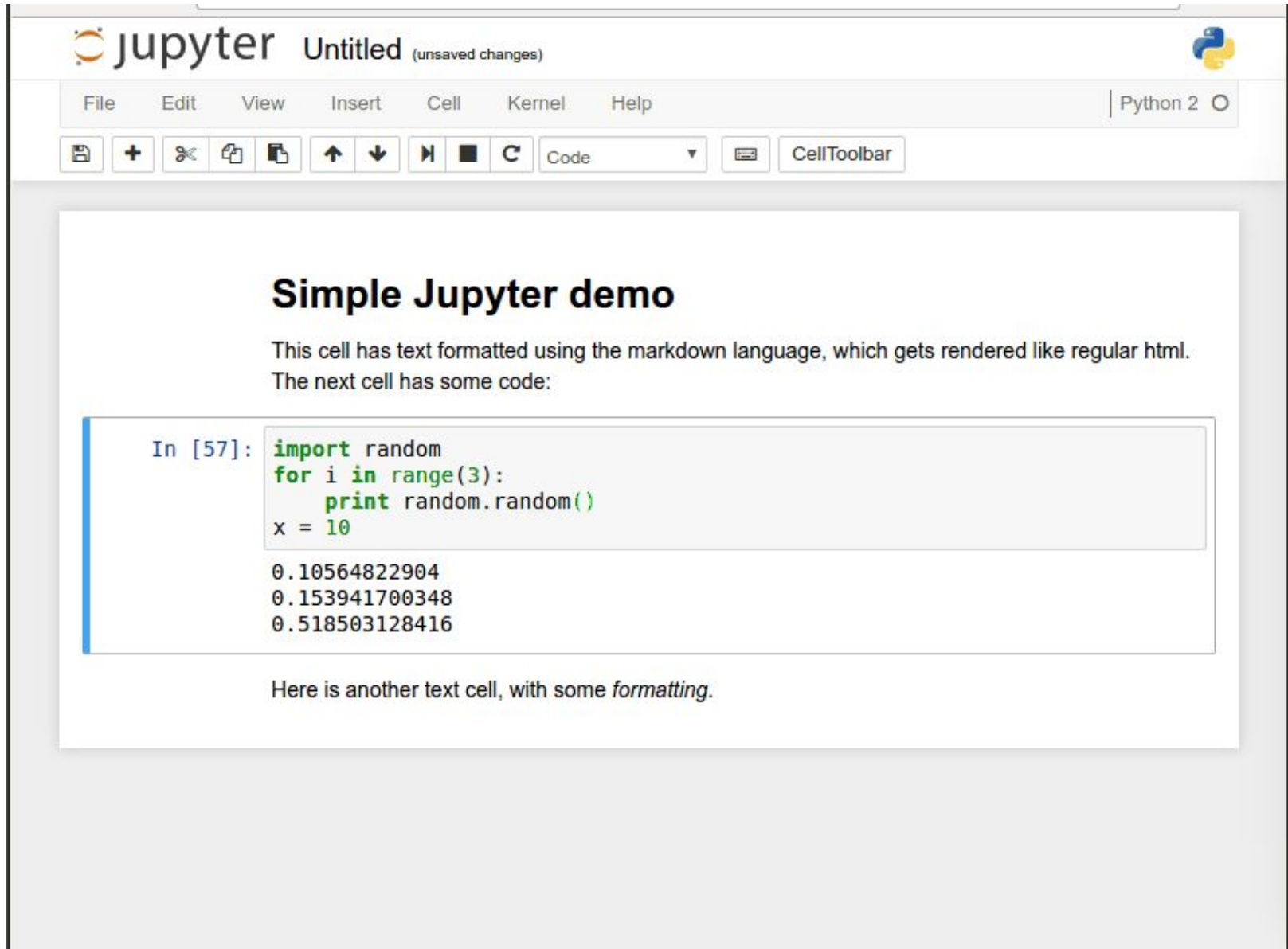
Getting Python

- Conda (from Continuum Analytics)
- Python distributions
 - Anaconda: Download full collection
 - Miniconda: Download what you need
- Simplifies installing libraries with compiled extensions
- Conda-forge community channel with over 1500 packages

Jupyter Notebooks

- Notebooks combine text, code, and rich output
 - HTML and Javascript
 - Images
- Share analysis and code
- Learning resources
- Runs on its own server over http -> accessed through a web browser
- Notebook server can be remote--close to data!
- JupyterHub authenticated, multi-user server

Notebook Demo



The screenshot shows a Jupyter Notebook window titled "Untitled (unsaved changes)". The interface includes a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", and "Help". A toolbar contains icons for file operations and cell execution. The notebook content consists of two cells: a text cell with a heading and explanatory text, and a code cell with Python code and its output.

jupyter Untitled (unsaved changes) Python 2

File Edit View Insert Cell Kernel Help

Code CellToolbar

Simple Jupyter demo

This cell has text formatted using the markdown language, which gets rendered like regular html.
The next cell has some code:

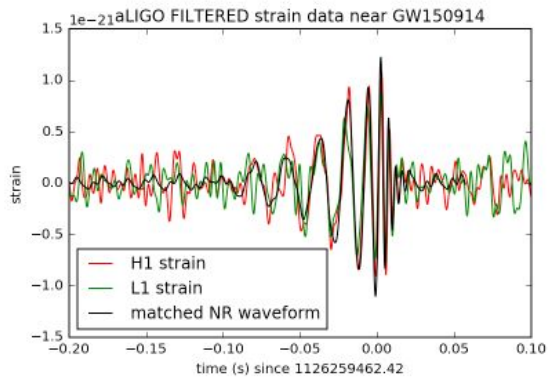
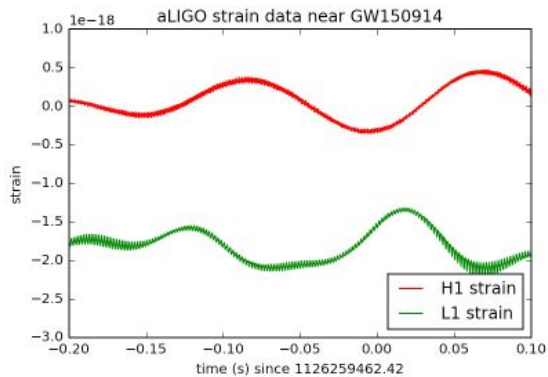
```
In [57]: import random
for i in range(3):
    print random.random()
x = 10
```

0.10564822904
0.153941700348
0.518503128416

Here is another text cell, with some *formatting*.

Python for Gravitational Waves?

```
# First, shift L1 by 1 ms, and invert. See the GW150914 detection paper for why!
strain_L1_filt = -np.roll(strain_L1_filt, int(0.007*fs))
# We also have to shift the NR template by 2 ms to get it to line up properly with the data
plt.figure()
plt.plot(time-tevent, strain_H1_filt, 'r', label='H1 strain')
plt.plot(time-tevent, strain_L1_filt, 'g', label='L1 strain')
plt.plot(NRtime+0.002, NR_H1_filt, 'k', label='matched NR waveform')
plt.xlim([-0.2, 0.1])
plt.ylim([-1.5e-21, 1.5e-21])
plt.xlabel('time (s) since '+str(tevent))
plt.ylabel('strain')
plt.legend(loc='lower left')
plt.title('aLIGO FILTERED strain data near GW150914')
plt.savefig('GW150914_H1_strain_filtered.png')
```



Community

- All of this work is free and open-source
- **ANYONE** can contribute
- Pyaos Mailing list
- Community conferences and workshops
 - Unidata Training Workshops
 - SciPy (10-16 July Austin, TX)
 - PyCon (17-25 May Portland, OR)
 - AnacondaCon (7-9 February Austin, TX)
 - PyData

Unidata and MetPy

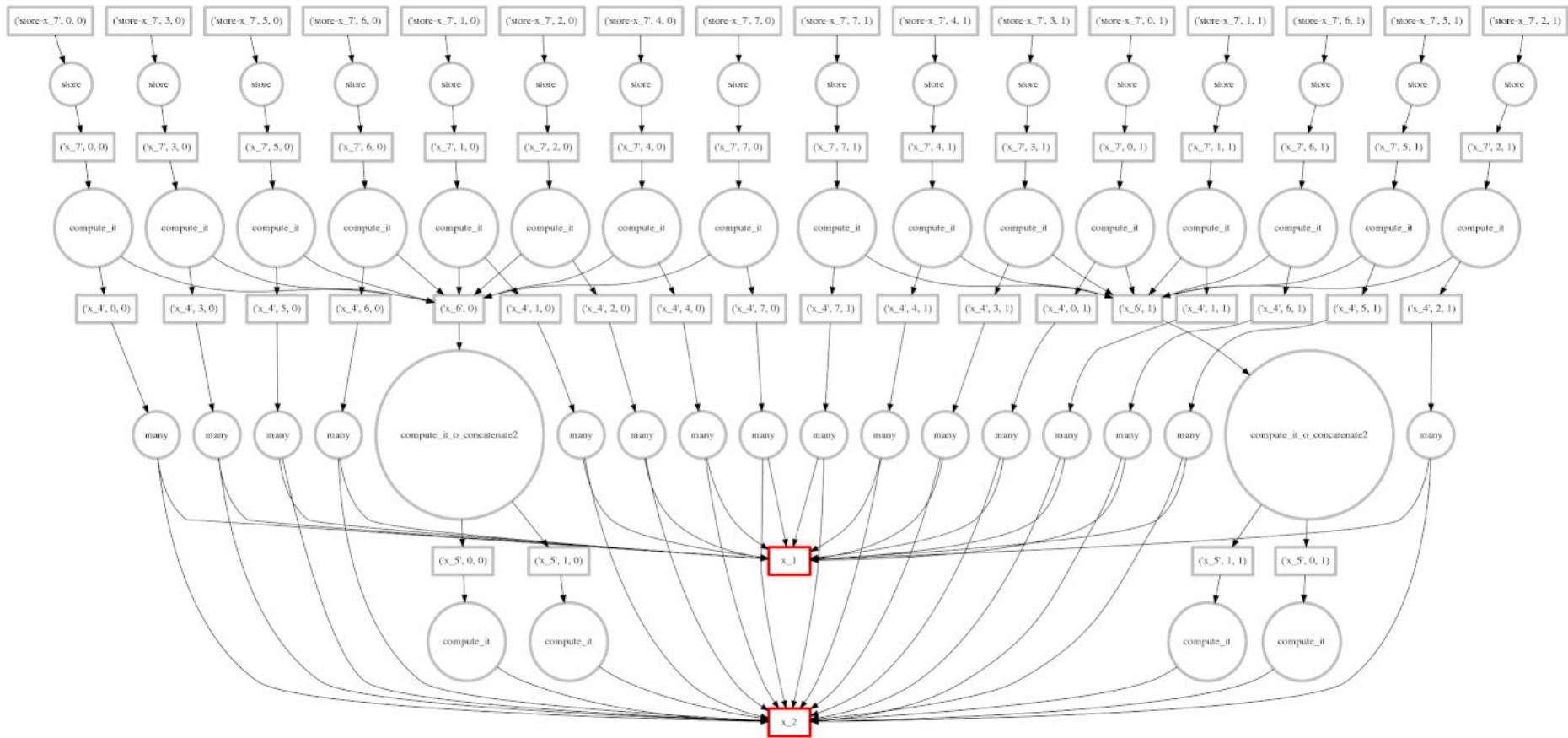
- Unidata is here to help support the use of Python in the atmospheric sciences
- Python projects on GitHub: MetPy, Siphon, netcdf4-python, notebook-gallery
 - <https://github.com/Unidata>
- Join us!
 - Bug reports and feature requests
 - Contributions: code, examples, documentation
- Follow along:
 - [@metpy](#) on Twitter
 - python-users-join@unidata.ucar.edu

Coming soon to a python
environment near you...

Python Performance

- Dask
 - “Versatile parallel programming with task scheduling”
- Numba
 - JIT compile parts of Python code with LLVM
- PyPy
 - Full JIT-compiled Python interpreter
 - Only recent support for C-extensions and numpy

Task Demo



```
expr = x.T.dot(y) - y.mean(axis=0)
```


Jupyter and More Jupyter

- Notebook Widgets
 - Make your notebooks interactive without editing
- Dashboards
 - Layout notebooks outputs on a grid or cell
 - Works really well with widgets
- Dashboard server
 - Deploy a notebook as a web application
- JupyterLab

Notebook Widgets

jupyter GFS_Widget Last Checkpoint: 09/26/2016 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Python [conda env:py35]

```
time_var = data.variables[time_var(data.variables['temperature_surface'])]
time_var = num2date(time_var[:], time_var.units).tolist()
time_strings = [t.strftime('%m/%d %H:%M') for t in time_var]

# Combine 1D latitude and longitudes into a 2D grid of locations
lon_2d, lat_2d = np.meshgrid(lon, lat)

In [ ]: def plot(varname='', time=0, colormap=''):
    variable = data.variables[varname][:]
    fig = plt.figure(figsize=(10,8))
    ax = fig.add_subplot(111, projection=ccrs.PlateCarree())
    ax.set_extent([235., 290., 20., 55.])
    ax.set_title('GFS 12-Hour Forecast', size=16)

    # Add state boundaries to plot
    states_provinces = cfeature.NaturalEarthFeature(category='cultural', name='admin_1_states_provinces_lines',
                                                    scale='50m', facecolor='none')
    ax.add_feature(states_provinces, edgecolor='black', linewidth=1)

    # Add country borders to plot
    country_borders = cfeature.NaturalEarthFeature(category='cultural', name='admin_0_countries',
                                                    scale='50m', facecolor='none')
    ax.add_feature(country_borders, edgecolor='black', linewidth=1)

    if varname == 'Temperature_surface':
        variable = (variable * units.kelvin).to('degF')

    # Contour based on variable chosen
    c = ax.contourf(lon_2d, lat_2d, variable[time_strings.index(time)], cmap=colormap)
    cb = fig.colorbar(c, ax=ax, shrink=0.7)

    if varname == 'Temperature_surface':
        cb.set_label(r'$^{o}F$', size='large')
    if varname == 'Relative_humidity_entire_atmosphere_single_layer':
        cb.set_label(r'%', size='large')
    if varname == 'Wind_speed_gust_surface':
        cb.set_label(r'$m/s$', size='large')

In [ ]: x = interactive(plot, varname=widgets.DropDown(options={'Temperature': 'Temperature_surface',
'Relative Humidity': 'Relative_humidity_entire_atmosphere_single_layer', 'Wind Speed': 'Wind_speed_gust_surface'},
description='Variable', alignment='center'), time=widgets.SelectionSlider(description='Time', options=time_strings,
width='40%'), colormap=widgets.RadioButtons(description='Colormap', options=['viridis', 'coolwarm', 'YlGnBu', 'RdPu']
display(x)
```

Dashboard Layout

jupyter HRRR Plotter Last Checkpoint: 09/14/2016 (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Python 3

Code CellToolbar Dashboard View: </>

```
In [ ]: cat_url = 'http://thredds.ucar.edu/thredds/catalog/grib/NCEP/HRRR/CONUS_2p5km/catalog.xml'
ncss_url = get_latest_access_url(cat_url, 'NetcdfSubset')
ncss = NCSS(ncss_url)
```

```
In [ ]: def handler(change):
        make_plot(change['new'])

def get_data(varname):
    start = datetime.utcnow()
    query = ncss.query()
    query.lonlat_point(lon=-105, lat=40)
    query.time_range(start, start + timedelta(days=5))
    query.variables(varname)
    data = ncss.get_data(query)
    return np.array(data['date']), np.array(data[varname])

def make_plot(var):
    global line
    status.value = 'Getting Data...'
    x, y = get_data(var)
    # ax.lines.clear()
    line.set_data(x, y)
    ax.ignore_existing_data_limits = True
    ax.update_line_limits(line)
    status.value = 'Refreshing...'
    # line, = ax.plot(*get_data(var))
    ax.autoscale_view()
    fig.canvas.draw()
    status.value = 'Done.'
```

```
In [ ]: field_chooser = widgets.Dropdown(options=sorted(v for v in ncss.variables
                                                    if v.endswith('surface') or v.endswith('ground')))
field_chooser.observe(handler, names='value')
```

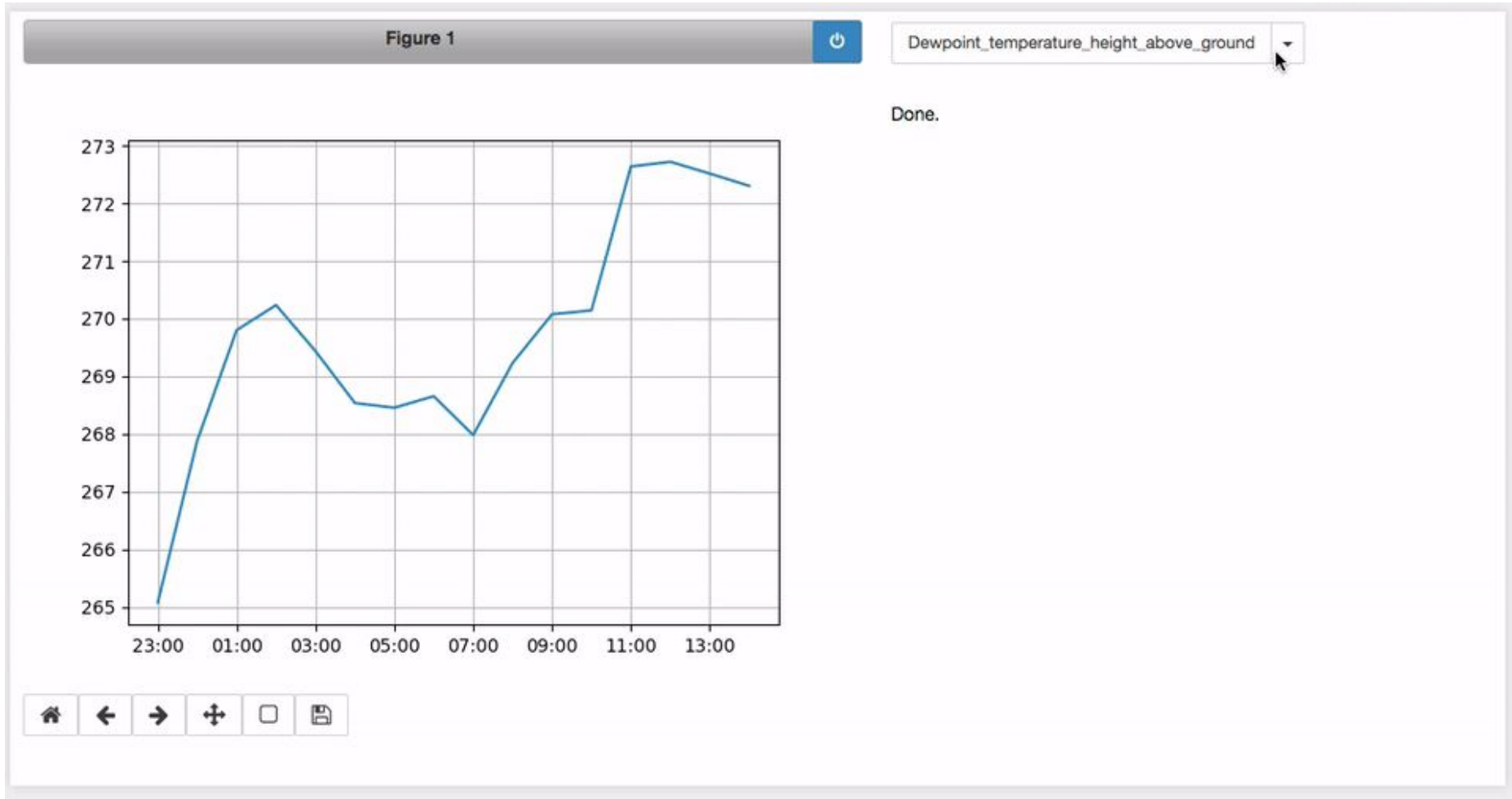
```
In [ ]: field_chooser
```

```
In [ ]: status = widgets.Label('Welcome!')
status
```

```
In [ ]: fig, ax = plt.subplots(1, 1)
ax.autoscale(True)
ax.grid(True)
line, = ax.plot(*get_data(field_chooser.value))
```

```
In [ ]:
```

Dashboard Running



JupyterLab

About

Welcome to the JupyterLab alpha preview

This demo gives an alpha-level preview of the JupyterLab environment. Here is a brief description of some of the things you'll find in this demo.

File Browser

Clicking the "Files" tab, located on the left, will toggle the file browser. Navigate into directories by double-clicking, and use the breadcrumbs at the top to navigate out. Create a new file/directory by clicking the plus icon at the top. Click the middle icon to upload files, and click the last icon to reload the file listing. Drag and drop files to move them to subdirectories. Click on a selected file to rename it. Sort the list by clicking on a column header. Open a file by double-clicking it or dragging it into the main area. Opening an image displays the image. Opening a code file opens a code editor. Opening a notebook opens a very preliminary proof-of-concept **non-executable** view of the notebook.

Command Palette

Clicking the "Commands" tab, located on the left, will toggle the command palette. Execute a command by clicking, or navigating with your arrow keys and pressing Enter. Filter commands by typing in the text box at the top of the palette. The palette is organized into categories, and you can filter on a single category by clicking on the category header or by typing the header surrounded by colons in the search input (e.g., `:file:`).

You can try these things out from the command palette:

- Open a new terminal (requires OS X or Linux)
- Open a new file
- Save a file
- Open up a help panel on the right

Main area

The main area is divided into panels of tabs. Drag a tab around the area to split the main area in different ways. Drag a tab to the center of a panel to move a tab without splitting the panel (in this case, the whole panel will highlight, instead of just a portion). Resize panels by dragging their borders (be aware that panels and sidebars also have a minimum width). A file that contains changes to be saved has a star for a close icon.

Notebook

Opening a notebook will open a minimally featured notebook. Code execution, Markdown rendering, and basic cell toolbar actions are supported. Future versions will add more features from the existing Jupyter notebook.

Would you like to know more?

Takeaways

- Python continues to be a strong, community-driven tool for science
- Tools for meteorology are growing
 - MetPy wants your contributions!
- Leverages the efforts of a broad community
 - Better tools
 - Better skills for graduates in our field
 - We get out what we put in
- Jupyter notebooks and javascript are important tool for our increasing data volumes

Unidata is one of the University Corporation for Atmospheric Research (UCAR)'s Community Programs (UCP), and is funded primarily by the National Science Foundation (Grant NSF-1344155).



Resources

- PyAOS: <http://pyaos.johnny-lin.com/>
- SciPy Organization: <https://scipy.org/>
- Unidata Python Training Workshop:
<http://unidata.github.io/unidata-python-workshop>
- MetPy: <http://unidata.github.io/MetPy>
- JupyterLab: <https://github.com/jupyterlab>
- Dashboard: <https://github.com/jupyter/dashboards>
- Anaconda: <https://www.continuum.io/downloads>
- Miniconda: <http://conda.pydata.org/miniconda.html>

Links to Examples

- Scikit-learn:
http://scikit-learn.org/stable/auto_examples/index.html
- Scikit-image:
http://scikit-image.org/docs/dev/auto_examples/
- CartoPy:
<http://scitools.org.uk/cartopy/docs/latest/gallery.html>
- MetPy: <https://unidata.github.io/MetPy/examples/index.html>
- Matplotlib:
 - <http://matplotlib.org/gallery.html>
 - <http://matplotlib.org/examples/index.html>

Examples (cmocean)

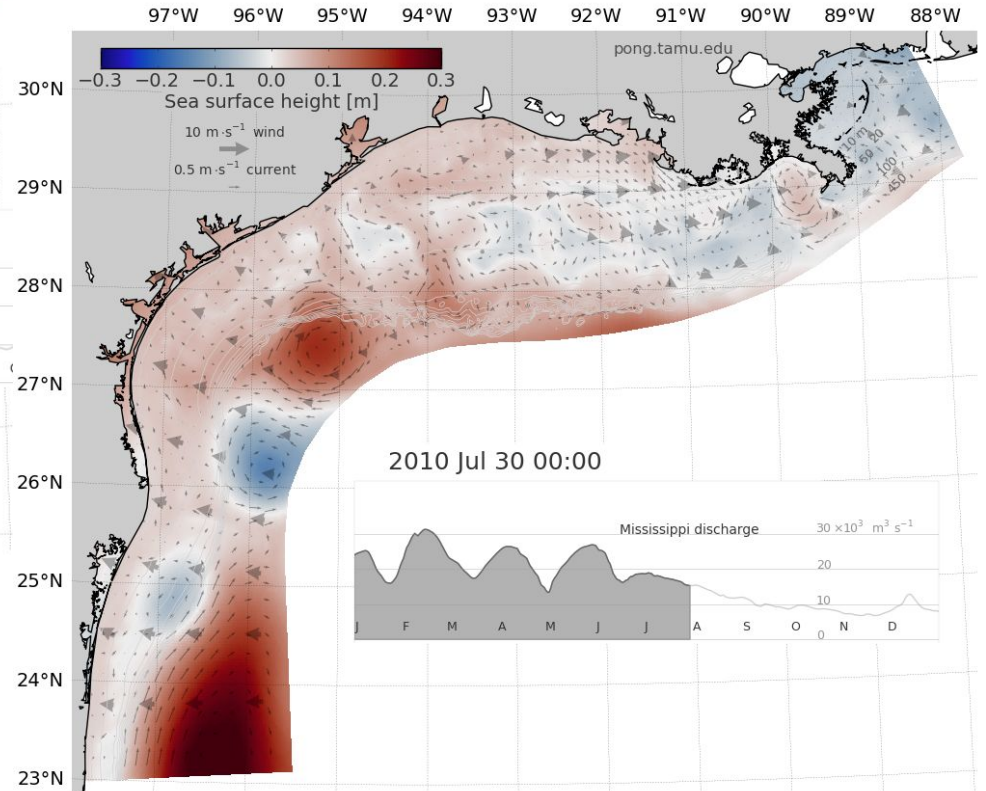
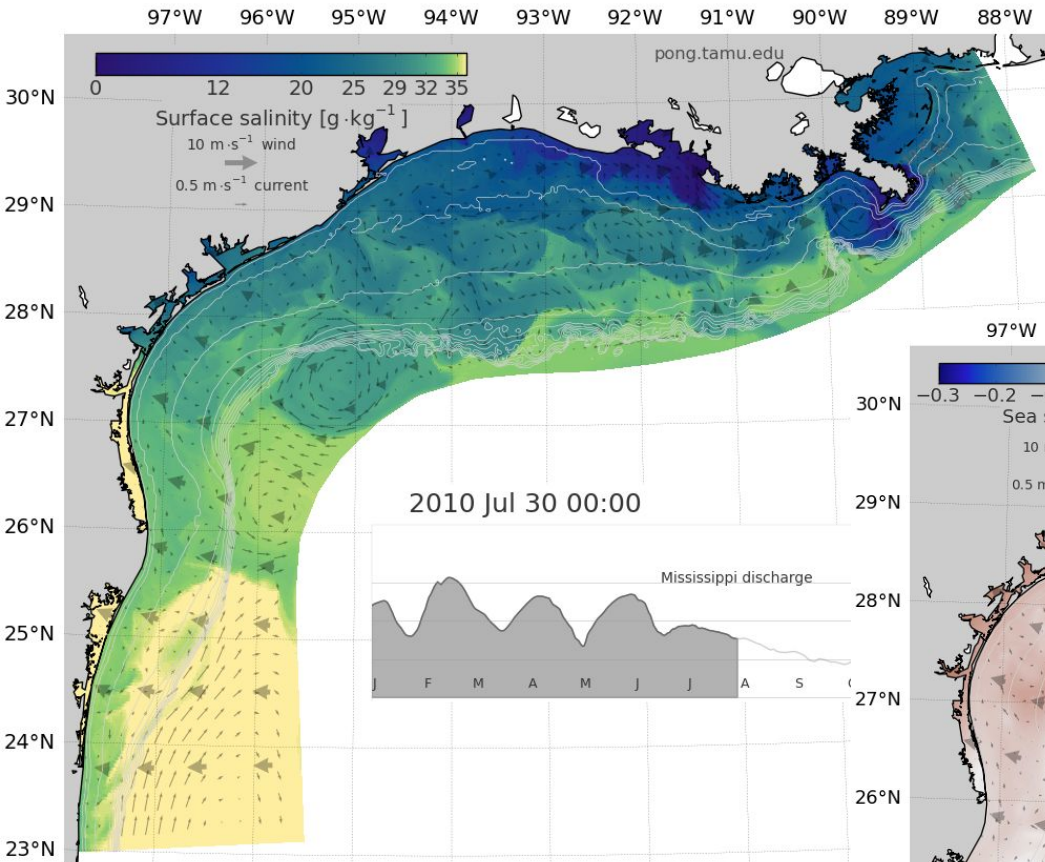


Image credit to Dr. Kristen Thyng, Texas A&M University