Argonne NATIONAL LABORATORY

ARM CLIMATE RESEARCH FACILITY

The Language, Libraries and Culture of Python in Meteorology

Jonathan Helmus

Argonne National Laboratory, USA

python™

NumPy

Cython

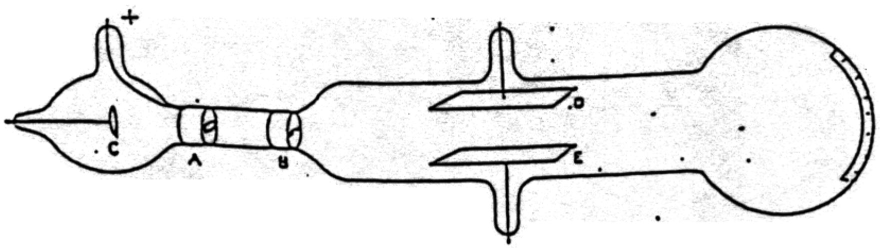
SciPy

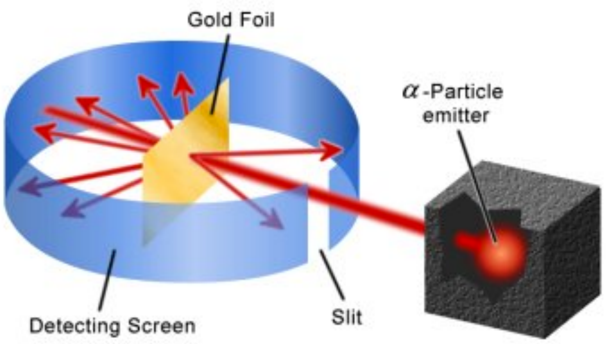
This presentation has been created by UChicago Argonne, LLC, Operator of Argonne National Laboratory ("Argonne"). Argonne, a U.S. Department of Energy Office of Science laboratory, is operated under Contract No. DE-AC02-06CH11357. This research was supported by the Office of Biological and Environmental Research of the U.S. Department of Energy as part of the Atmospheric Radiation Measurement Climate Research Facility.

U.S. DEPARTMENT OF ENERGY

Technology Powers Scientific Discovery

- Scientific advancements are driven by state-of-the-art technologies.
- New technologies lead the way to scientific breakthroughs and a more complete understanding of the world around us.

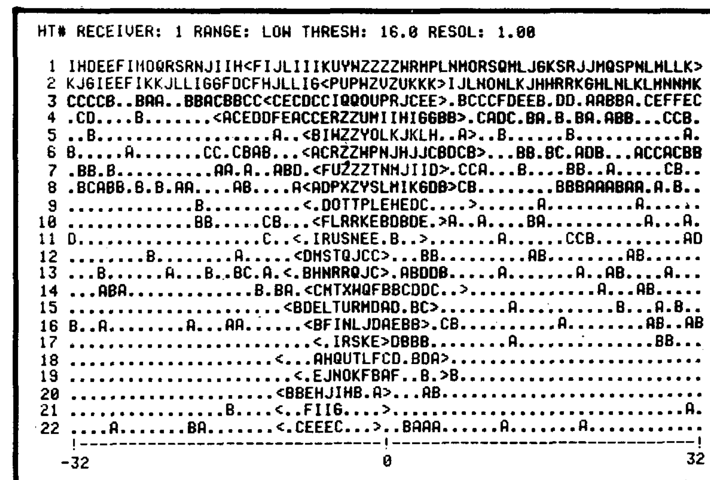




Computers Prime Technology for Science

- Computers are the most significant technology available to scientists today.
- The ability to perform fast computations has vastly changed meteorology and will continue to change the field in the future.

ARM CSAPR Radar
http://www.arm.gov/instruments/csapr



```
HT# RECEIVER: 1 RANGE: LOW THRESH: 16.0 RESOL: 1.00

 1 IHDEEFIMOQRSRNJIIH<FIJLIIIKUYWZZZZWRMPLNMORSQMLJGKSRJJMQSPNLMLLK>
 2 KJGIEEFIKKJLLIGGFDCFHJLLIG<PUPWZVZUKKK>IJLNONLKJHHRRKGHLNLKLHNNMK
 3 CCCCB..BAA..BBACBBCC<CECDCCIQQOUPRJCEE>.BCCCFDEEB.DD.AABBA.CEFFEC
 4 .CD....B.......<ACEDDFEACCERZZUMIIHIGGBB>.CADC.BA.B.BA.ABB...CCB.
 5 ..B..................A..<BIHZZYOLKJKLH..A>..B......B...........A.
 6 B.....A.......CC.CBAB...<ACRZZHPNJHJJCBDCB>...BB.BC.ADB...ACCACBB
 7 .BB.B..........AA.A..ABD.<FUZZZTNMJIID>.CCA...B....BB..A.....CB..
 8 .BCABB.B.B.AA....AB....A<ADPXZYSLMIKGDB>CB........BBBAAABAA.A.B..
 9 ..............B..........<.DOTTPLEHEDC....>......A..........A.....
10 ..............BB.....CB...<FLRRKEBDBDE.>A..A....BA...........A...A.
11 D...................C..<.IRUSNEE.B..>.......A......CCB.........AD
12 .........B.........A.....<DMSTQJCC>...BB..........AB.........AB......
13 ...B......A...B..BC.A.<.BHNRRQJC>.ABDDB.....A........A..AB....A...
14 ...ABA..............B.BA.<CMTXWQFBBCDDC..>..............A...AB.....
15 .....................<BDELTURMDAD.BC>.......A...........B...A.B..
16 B..A.........A...AA.......<BFINLJDAEBB>.CB...........A.........AB..AB
17 .........................<.IRSKE>DBBB.........A..................BB...
18 ......................<...AHQUTLFCD.BDA>............................
19 .......................<.EJNOKFBAF..B.>B.........................
20 ......................<BBEHJIHB.A>...AB..........................
21 ................B....<..FIIG....>...............................A.
22 ....A........BA.......<.CEEEC...>..BAAA......A........A............
   !------------------------------!------------------------------!
  -32                             0                              32
```

FIG. 7. Sample of real-time Doppler spectra available to a radar operator by telephone.

Source: R. G. Strauch *el al*, JTech, **1984**, 1, 37

Programming: Controlling Computers for Science

- The daily operation of science involves computers.
- Computers require specific directions in the form of software to realize their potential to power scientific research and operations.
- The ability to create software customized to a specific scientific task is often needed.
- Writing software can be a time consuming and burdensome task.
  - Scientists estimate that 35% of their research time is spent developing software [1].
  - Most scientists learn to develop software informally from their peers [2].
  - The choice of programming language affects the time required for development.

[1] P. Prabhu, et al, A Survey of the Practice of Computational Science, **2011**, 10.1145/2063348.2063374
[2] J. E. Hannay, *et al*, How Do Scientists Develop and Use Scientific Software, **2009**, 10.1109/SECSE.2009.5069155.

Python is Ideal for Writing Scientific Software

- Python is the ideal programming language for writing software to meet the computational challenges in meteorology and atmospheric science.

- Developing software in Python is simple and fast due to:
  - The design and philosophy of the **language**.
  - The availability of a number of high-quality, efficient scientific Python **libraries.**
  - The welcoming, vibrant, and growing Python **community**.

Python: the Language

Python, the programming language is:

- A **high-level** language.
  - Allows one to focus on the the task concepts rather than the details of the computer.
  - Makes development simple and quick.

- **Interpreted** and interactive.
  - No compilation step, speeds up development cycle.
  - Interactive shell (REPL) for testing and debugging.

- **General purpose**.
  - Can be used to write software across a wide range of applications, not domain specific.
  - Larger user and developer base.

Python: the Language

Python is an ideal language for scientific programming because it is:

- **Concise** and **readable**

Python

```python
colors = ['red', 'blue', 'green']
for color in colors:
    print(color)
```

Perl

```perl
@colors = ("red", "blue", "green");
foreach (@colors) {
    print "$_\n”;
}
```

- **Open Source** and **Free**
  - No company controls the software, no licensing costs.

- Has a large number of third-party **libraries**.
  - PyPI contains more than 72,000 packages.
  - Many science-focused libraries exist covering a variety of fields.

- Easy to **interface** with existing C, C++ and Fortran code.
  - Tools like Cython, cffi, ctypes, and F2PY make interfacing with existing code simple.

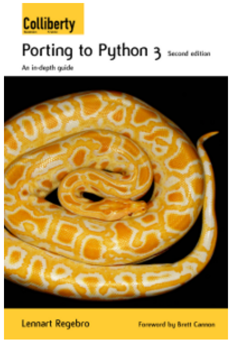
Python: the Challenges

Python does face some challenges:

- Execution speed is **slow** compared to some programming languages.
  - Many tasks are not limited by execution speed.
  - Options exist to speed up Python code.

- The language is not the best at some types of **parallel** computations.
  - The Global Interpreter Lock (GIL) poses limits of multi-threaded performance.
  - Handling concurrency is not as robust as some languages (Go, Erlang, Clojure).
  - Support for computation on GPUs and other highly parallel architectures is limited and requires additional libraries (Numba, PyCUDA)

- The language is currently **in flux**.
  - Python 3 is backwards incompatible with Python 2.
  - Some libraries do not support Python 3.



Scientific Python Libraries: Introduction

The Scientific Python Ecosystem or **SciPy stack** is a collection of open source software for scientific computing in Python. Some of the core packages are:

- **NumPy** : Provides powerful, efficient multi-dimensional arrays in Python
- **SciPy** : Fundamental numerical algorithms for common tasks in science.
- **matplotlib** : comprehensive publication-quality 2D plotting

- **Jupyter/IPython** : Rich, interactive interfaces for processing data and testing ideas.
- **pandas** : High performance, easy to use data structures.
- **SymPy** : Symbolic mathematics and computer algebra

The Python ARM Radar Toolkit: Py-ART

- Py-ART is a module for visualizing, correcting and analyzing **weather radar data** using packages from the scientific Python stack.
- Development began to address the needs of the **ARM** program with the acquisition of **multiple scanning cloud and precipitation radars**.
- The project has since been expanded to work with a **variety of weather radars**, including NEXRAD, and has a wide user base including radar researchers, weather enthusiasts and climate modelers.
- Available on GitHub as **open source software**, arm-doe.github.io/pyart/.
- Conda packages are available at anaconda.org/jjhelmus

ANACONDA CLOUD

Scientific Python Libraries: NumPy

- NumPy is the fundamental package for scientific computation in Python.

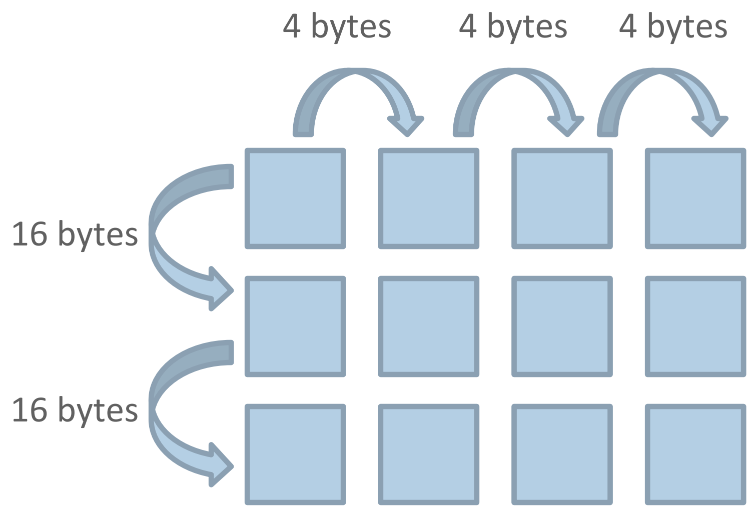
- NumPy provides:
  - Powerful, efficient (fast) multi-dimensional **array object**, the ndarray class.
  - Robust methods for **manipulating** these arrays.
  - Masked and record array objects.
  - Routines for linear algebra, Fourier transform, and random numbers.
  - Comprehensive, well-written documentation, http://docs.scipy.org/doc/.

- Other scientific Python packages build on NumPy to add additional features and abilities.

NumPy’s ndarray details

- NumPy’s core functionality is provided by its **ndarray** object.
- A ndarray is a **homogeneous**, **strided** view on a **contiguous** block of memory.

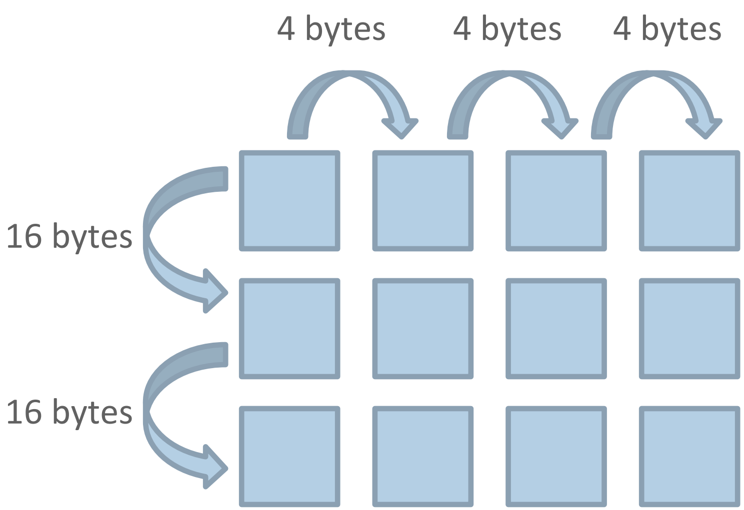
(3, 4) ndarray of float32
strides (16, 4)



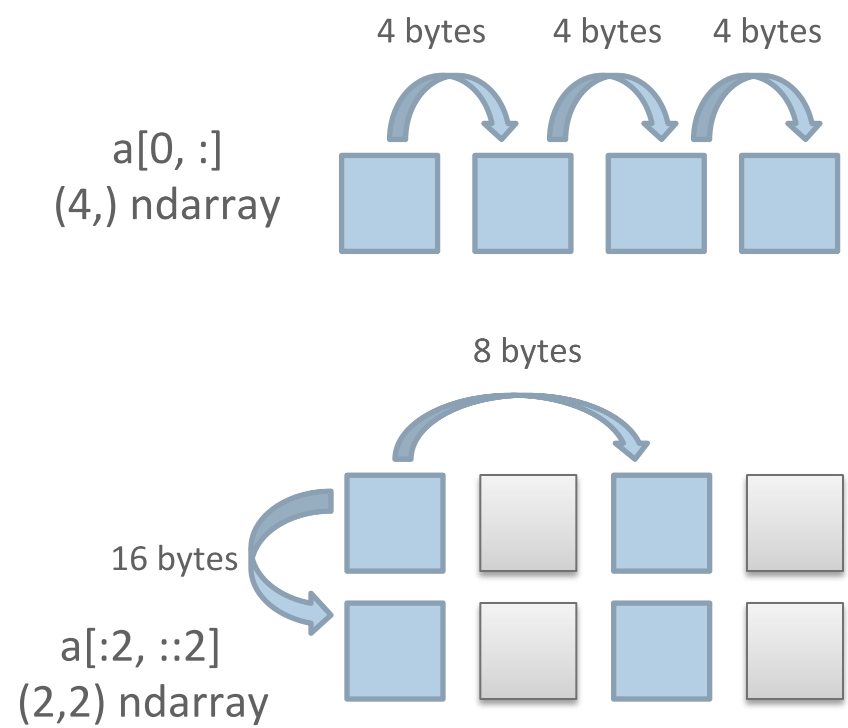
- Although simple, the ndarray is a **powerful construct** as the location of underlying memory can be passed to other languages (C, C++, Cython and Fortran) without the need to copy data.

ndarray views

- Slicing a NumPy ndarray almost always creates a “**view**” of the data.



a = np.zeros((3, 4), np.float32)
(3,4) ndarray



- No copying of data is needed when accessing or modifying views.

NumPy in Py-ART

NumPy is used extensively throughout Py-ART as ndarrays are the primary objects used to store and manipulate numerical data.

```
elif data_type_name == 'PHIDP2':
    out[:] = 360. * (data.view('uint16') - 1.) / 65534.
    mask[data.view('uint16') == 0] = True

elif data_type_name == 'HCLASS2':
    out[:] = data.view('uint16')

elif data_type_name == 'XHDR':
    # Here we return an array with the times in milliseconds.
    return data[..., :2].copy().view('i4')

# one byte data types
elif data_type_name[-1] != '2':
    # make a view of left half of the data as uint8,
    # this is the actual ray data collected, the right half is blank.
    nrays, nbin = data.shape
    ndata = data.view('(2,) uint8').reshape(nrays, -1)[:, :nbin]
```

NumPy in Py-ART

```
# decode run length encoding
rle_size = radial_header['nbytes'] * 2
rle = np.fromstring(buf2[pos:pos+rle_size], dtype='>u1')
colors = np.bitwise_and(rle, 0b00001111)
runs = np.bitwise_and(rle, 0b11110000) // 16
radial[:] = np.repeat(colors, runs)
```

Scientific Python Libraries: SciPy

SciPy is a collection of mathematical algorithms and functions which build upon NumPy to provide efficient solutions to common numerical tasks.

SciPy is divided into subpackages which cover a number of scientific domains:

- Image processing
- Signal processing
- Interpolation
- Spatial data structures and algorithms
- Clustering
- Numerical integration
- Differential equations
- Statistics
- Sparse matrices

SciPy in Py-ART

SciPy functions for image processing, numerical integration, interpolation, spatial analysis and sparse matrix storage are all used in Py-ART.

```
import scipy.ndimage
…

def _find_regions(vel, gfilter, limits):
    """Find regions of similar velocity. """
    mask = ~gfilter
    label = np.zeros(vel.shape, dtype=np.int32)
    nfeatures = 0
    for lmin, lmax in zip(limits[:-1], limits[1:]):

        # find connected regions within the limits
        inp = (lmin <= vel) & (vel < lmax) & mask
        limit_label, limit_nfeatures = scipy.ndimage.label(inp)

        # add these regions to the global regions
        limit_label[np.nonzero(limit_label)] += nfeatures
        label += limit_label
        nfeatures += limit_nfeatures

    return label, nfeatures
```

Scientific Python Libraries: matplotlib

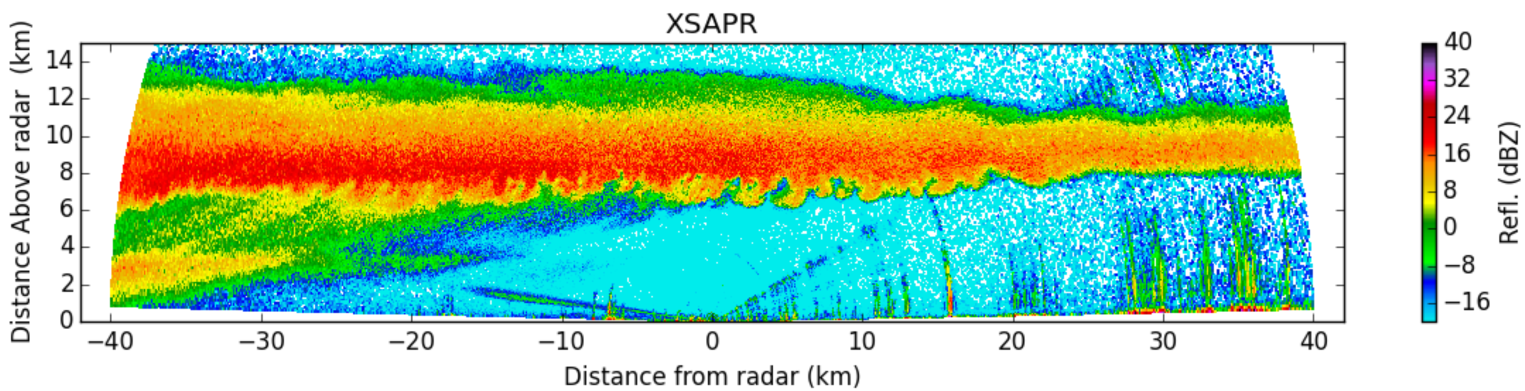
matplotlib is a plotting library which works with NumPy.

- Comprehensive 2D, publication quality plots.
  - Multiple plot types: line, scatter, image, contours, pseudocolor, …
  - Many output formats: png, jpg, svg, ps, pdf, …

- A limited set of 3D plots.
  - Line, scatter, wireframe, tri-surface, contour, polygon, …

- Plots can be examined interactively or embedded in applications.
  - Explore data in a GUI
  - ARTView : GUI viewer built on top of the Py-ART which embeds matplotlib plots

matplotlib in Py-ART



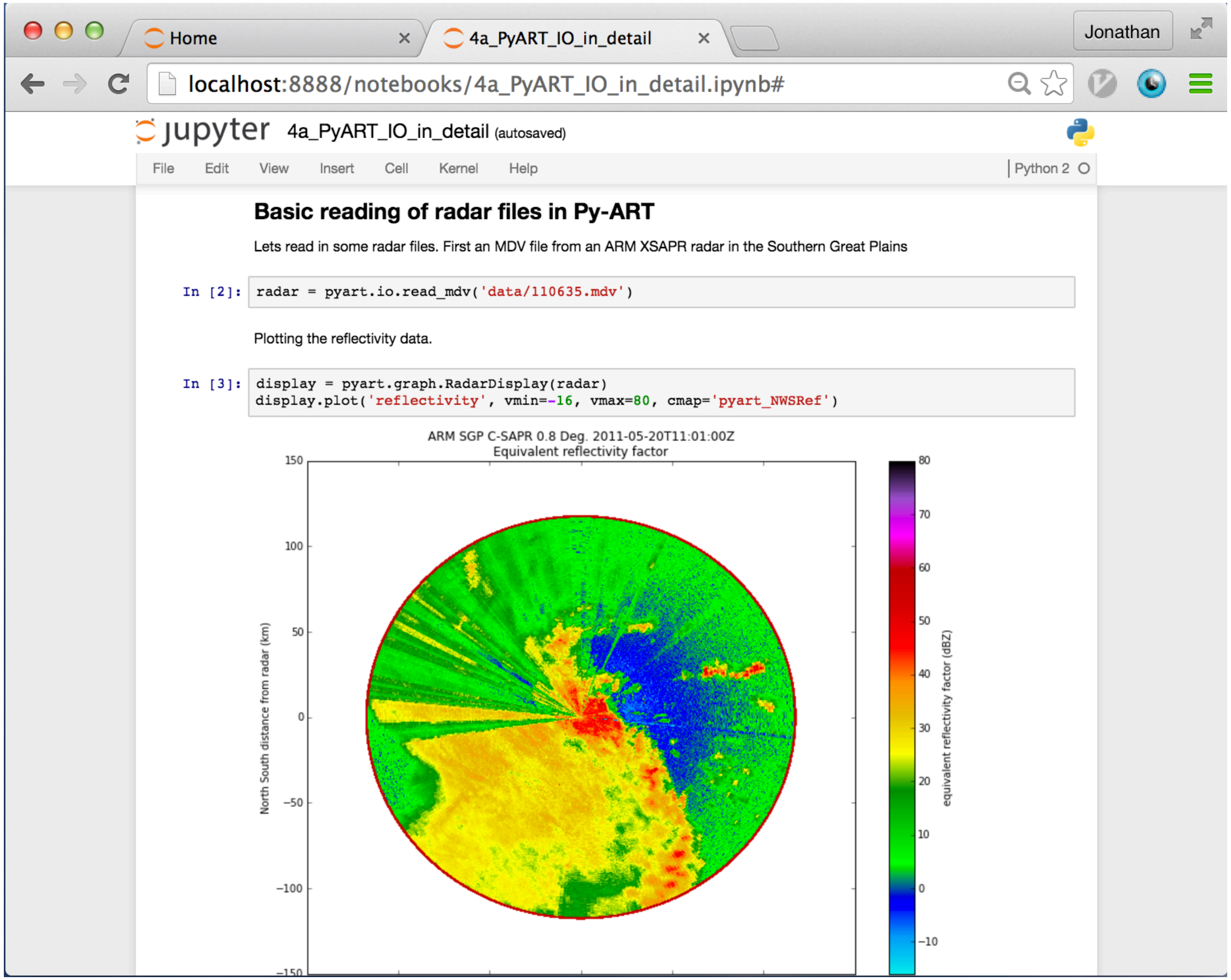

```
import matplotlib.pyplot as plt
import pyart

radar = pyart.io.read('sgpxsaprrhicmacI5.c0.20110524.015604_NC4.nc')
fig = plt.figure(figsize=(12, 3))

display = pyart.graph.RadarDisplay(radar)
display.plot('reflectivity_horizontal', vmin=-20, vmax=40,
             cmap='pyart_NWSRef', title='XSAPR', colorbar_label='Refl. (dBZ)')
display.set_limits(ylim=(0, 15), xlim=(-42, 42))

plt.tight_layout()
plt.savefig('figure.png')
```

Scientific Python Libraries: Jupyter/IPython

Project Jupyter (previously IPython) is a set of rich, interactive interfaces and tools for processing data and testing ideas in Python.

- User Interfaces
  - Juypter Console: Terminal based interactive Python environment
  - Juypter Notebook: Web based platform for authoring rich documents
  - Both have excellent integration with matplotlib
- Kernels
  - IPython: interactive computing in Python
  - ipyparallel: Lightweight parallel computing with notebook integration
  - IJulia, IRKernel, IRuby, IPerl, …
- Many other interesting tools: nbviewer, nbconvert, nbgrader, jupyterhub…

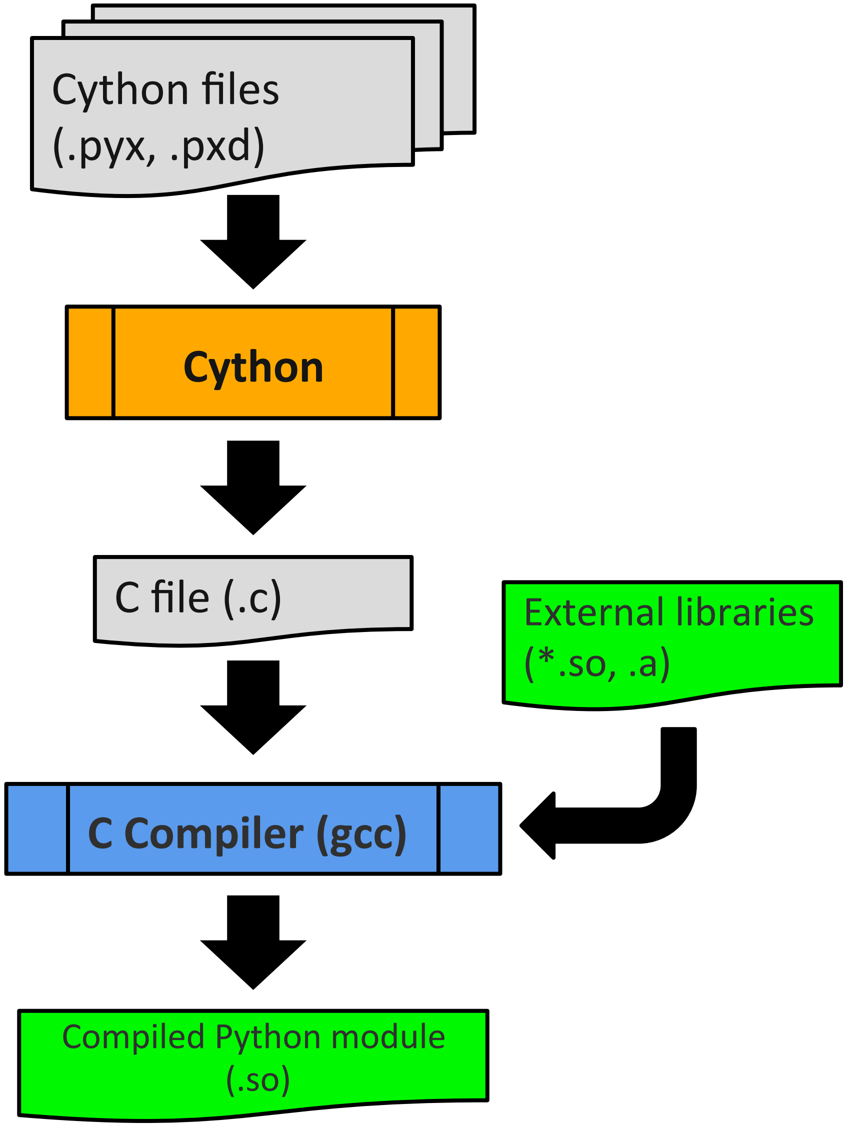
Jupyter/IPython in Py-ART



Scientific Python Libraries: Cython

Cython

- Python to C code translator.
- Generates a Python extension module.
- Can be used to speed up Python code by adding static type information.
- Also can be used to interact with C/C++ function and classes in external libraries.



Cython in Py-ART: wrapping libraries

_rsl_h.pxd

```
cdef extern from "rsl.h":

        ctypedef struct Radar:
            Radar_header h
            Volume **v

        ctypedef struct Radar_header:
            int month, day, year
            int hour, minute
            float sec
            ...

        ctypedef struct Volume:
            Volume_header h
            Sweep **sweep
            ...
    ...

    Radar * RSL_anyformat_to_radar(char *infile)
        ...
    void RSL_free_volume(Volume *v)
        void RSL_free_radar(Radar *r)
```

_rsl_interface.pyx

```
cimport _rsl_h

cdef class RslFile:
        cdef _rsl_h.Radar * _Radar
        cdef _rsl_h.Volume * _Volume

        def __cinit__(self, filename):
            self._Radar = _rsl_h.RSL_anyformat_to_radar(filename)
            if self._Radar is NULL:
                raise IOError('file cannot be read.')

        def __dealloc__(self):
            _rsl_h.RSL_free_radar(self._Radar)

        def get_volume(self, int volume_number):
            rslvolume = _RslVolume()
            rslvolume.load(self._Radar.v[volume_number])
            return rslvolume
        ...
        property month:
            def __get__(self):
                return self._Radar.h.month
            def __set__(self, int month):
                self._Radar.h.month = month
```

Cython in Py-ART: speeding up Python code

Cython

```
@cython.boundscheck(False)
@cython.wraparound(False)
def _fast_edge_finder(int[:, ::1] labels, float[:, ::1] data, int rays_wrap_around,
                      int max_gap_x, int max_gap_y, int total_nodes):
    """ Return the gate indices and velocities of all edges between regions. """
    cdef int x_index, y_index, right, bottom, y_check, x_check
    cdef int label, neighbor
    cdef float vel, nvel

    collector = _EdgeCollector(total_nodes)
    right = labels.shape[0] - 1
    bottom = labels.shape[1] - 1

    for x_index in range(labels.shape[0]):
        for y_index in range(labels.shape[1]):
            label = labels[x_index, y_index]
            if label == 0:
                continue
            vel = data[x_index, y_index]

            # left
            x_check = x_index - 1
            if x_check == -1 and rays_wrap_around:
                x_check = right     # wrap around
            if x_check != -1:
                neighbor = labels[x_check, y_index]
                nvel = data[x_check, y_index]
                ...
                # add the edge to the collection (if valid)
                collector.add_edge(label, neighbor, vel, nvel)
```

107 seconds vs. 0.234 seconds, x450 performance improvement.

More Scientific Python Libraries…

- pandas : data structures and analysis
- xray : labeled array and datasets
- Iris : meteorology/climate data model

- basemap : plot 2D data on maps
- pyproj : cartographic transformations
- Cartopy : cartographic tools for Python

- netCDF4-python : Read and write NetCDF files.
- h5py : Read and write HDF5 files

- scikit-learn : machine learning
- scikit-image : image processing

- And many, many more…

Python Community: Online

The Python **community** provides a welcoming, vibrant and helpful culture. Much of the community interactions occur online:

- **Websites**
  - Python.org: Documentation, tutorial, PyPI and a wiki.
  - Other websites, scipy.org, pyaos.johnny-lin.com
- **Social media**
  - Facebook, Twitter, Google+, IRC, YouTube
  - Blogs: planetpython.org and planet.scipy.org.
  - Podcasts: Talk Python to me and podcast.__init__
- **Mailing lists**
  - Nearly all the SciPy stack packages have their own mailing lists
  - PyAOS mailing list (pyaos.johnny-lin.com)

Python Community: in Person

The Python community also meets in person.

- Conferences
  - SciPy
  - PyCon
  - Local and specialized conferences (PyData, AMS)
  - Conference talks often available at pyvideo.org
- Local user groups
  - wiki.python.org/moin/LocalUserGroups
- Meetups and hackathons.
  - python.meetup.com

Python Community: for Developers

- Mailing lists
  - Many projects have a –dev mailing list
- Social coding sites
  - GitHub
  - Bitbucket
- NumFocus
- Scientific Python focused companies
  - Continuum Analytics
  - Enthought

My Own Path Through the Python Community

- Undergraduate - Chemistry at Michigan Technological University
- Ph.D. at The Ohio State University - Chemical Physics.
  - Learned to program in Python
  - Wrote nmrglue, a library for working with NMR data in Python
- Post-doc at UConn Heath Center
  - Continued to program in Python – signal processing for NMR
  - Attended first SciPy conference
- Advanced Algorithms Engineer at Argonne National Laboratory
  - Development lead of the widely used open source Py-ART project.
  - Contributing to other libraries in the SciPy stack.
  - Attend and at times present at SciPy, PyCon, ChiPy, …

Careers in Meteorology with Python Programming

Do you enjoy programming? Solving problems? Python?

- High Performance Computing (HPC)
- Information Processing Technologies
- Instrumentation and processing
- Data Assimilation
- Numerical Weather Prediction
- Climate and atmospheric modeling

Regardless of the career path you choose, learning Python will enhance your potential.

Questions?

Resources for Learning Python

- Websites
  - Python.org tutorial
  - Scipy Lecture Notes, www.scipy-lectures.org
- Books
  - "Think Python: How to Think Like a Computer Scientist" by Allen B. Downey
  - "Effective Computation in Physics: Field Guide to Research with Python" by Anthony Scopatz, Kathryn D. Huff
- In person
  - Tutorials at conference (PyCon, SciPy, AMS)
  - Software Carpentry, software-carpentry.org
- Videos
  - Many Python conferences makes their tutorials available on YouTube or pyvideo.org