

# Python for Data Analysis

Ben Zaitlen  
Travis E. Oliphant

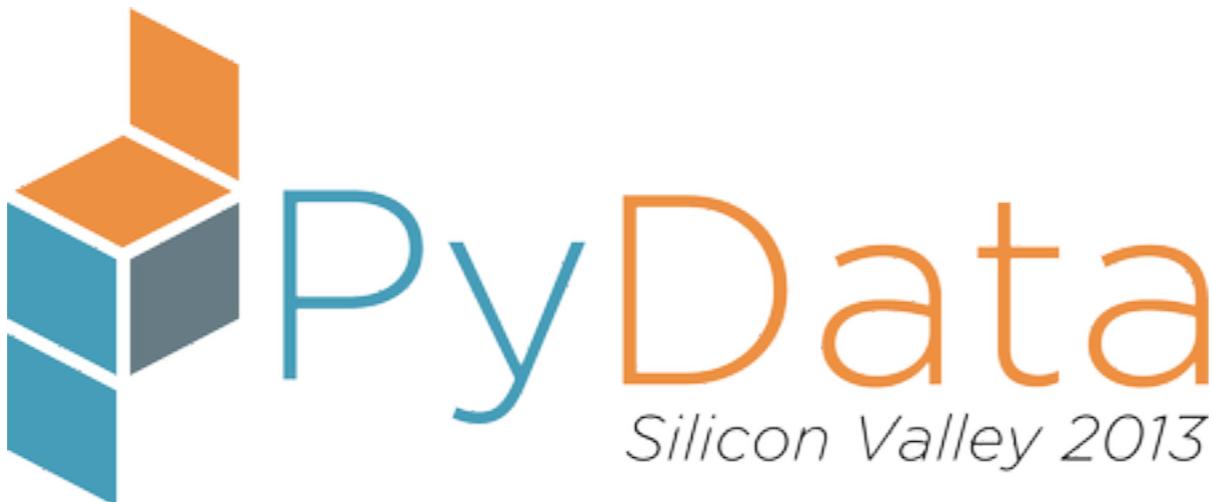
PyCon USA  
March 14, 2013



# Announcements

- Lunch is moved to Exhibit Hall D
- Schedule:
  - 9:00 - 10:50am
  - 20-min break
  - 11:10am - 12:20pm
- Please fill out survey
  - [https://www.surveymonkey.com/s/pycon2013\\_tutorials](https://www.surveymonkey.com/s/pycon2013_tutorials)
  - <https://goo.gl/PvHDc>
- <https://github.com/ContinuumIO/tutorials/>
- Volunteering Opportunities Available





<http://www.pydata.org/>

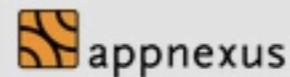
*Changing the way scientists, engineers, and analysts perceive big data*

**TUTORIALS:** March 18 | **MAIN CONFERENCE:** March 19-20

**Santa Clara Convention Center**  
Santa Clara, CA

*Sponsorship available. [Learn more.](#)*

## Sponsors



J.P.Morgan



Python Weekly

Software Developer's JOURNAL

Inside Analysis



# PyCon 2013 - Volunteering

A great way to meet people

# Low-commitment opportunities:

- SWAG bagging: Thur 4-8PM
  - Bag 10! (~1/2 hr)
- Registration Desk
  - 1-2 hrs helps
  - Anytime, but Fri => meet *everyone!*

# Current needs

- <http://bit.ly/pycon-volunteering-status>
- More info:
  - <http://bit.ly/pycon2013-volunteer>

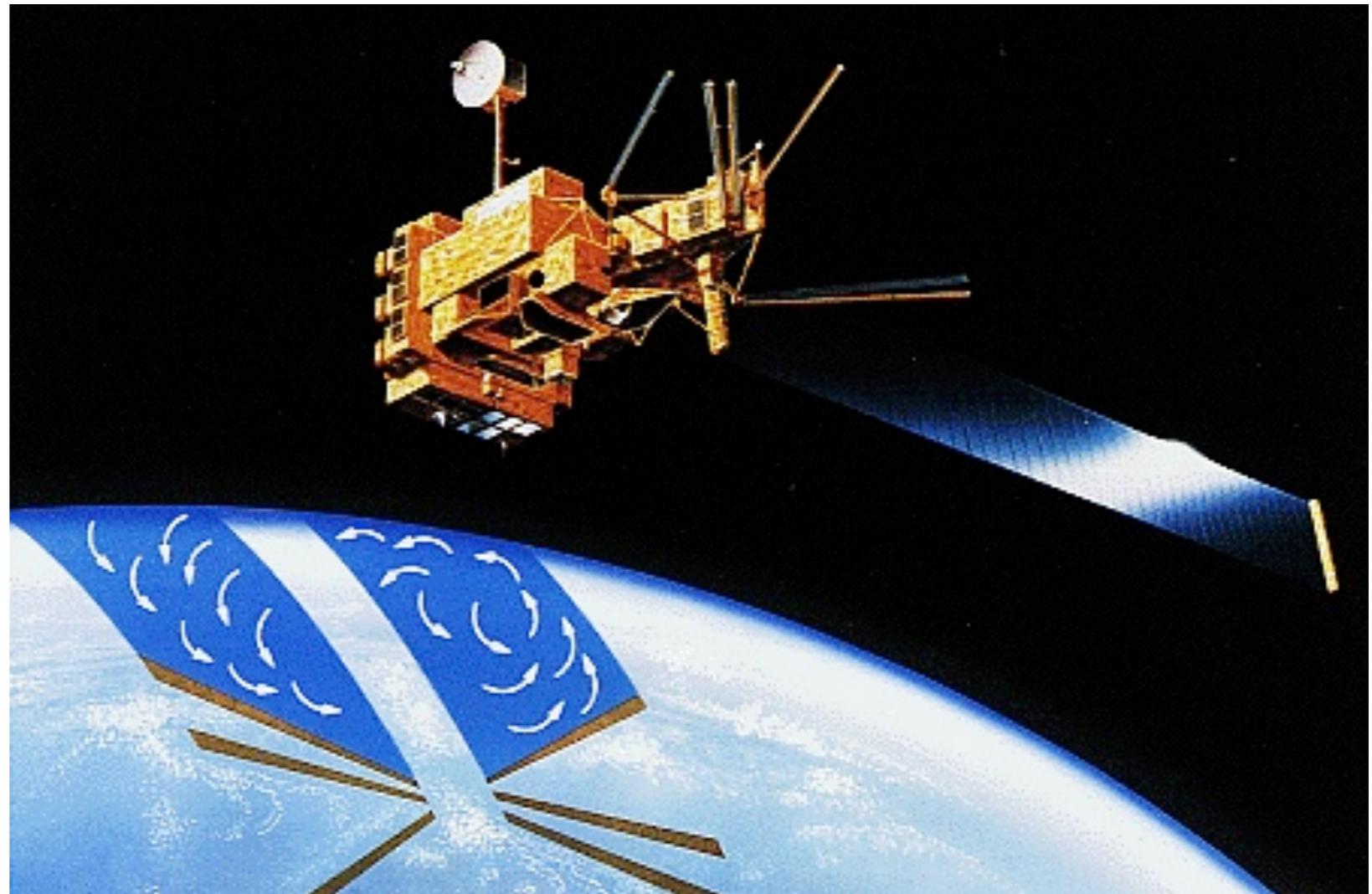
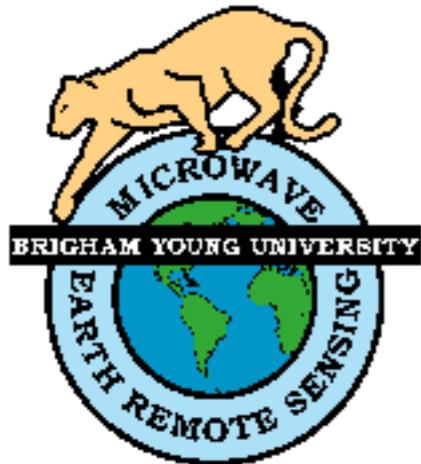
# Python for Data Analysis

Ben Zaitlen  
Travis E. Oliphant

PyCon USA  
March 14, 2012

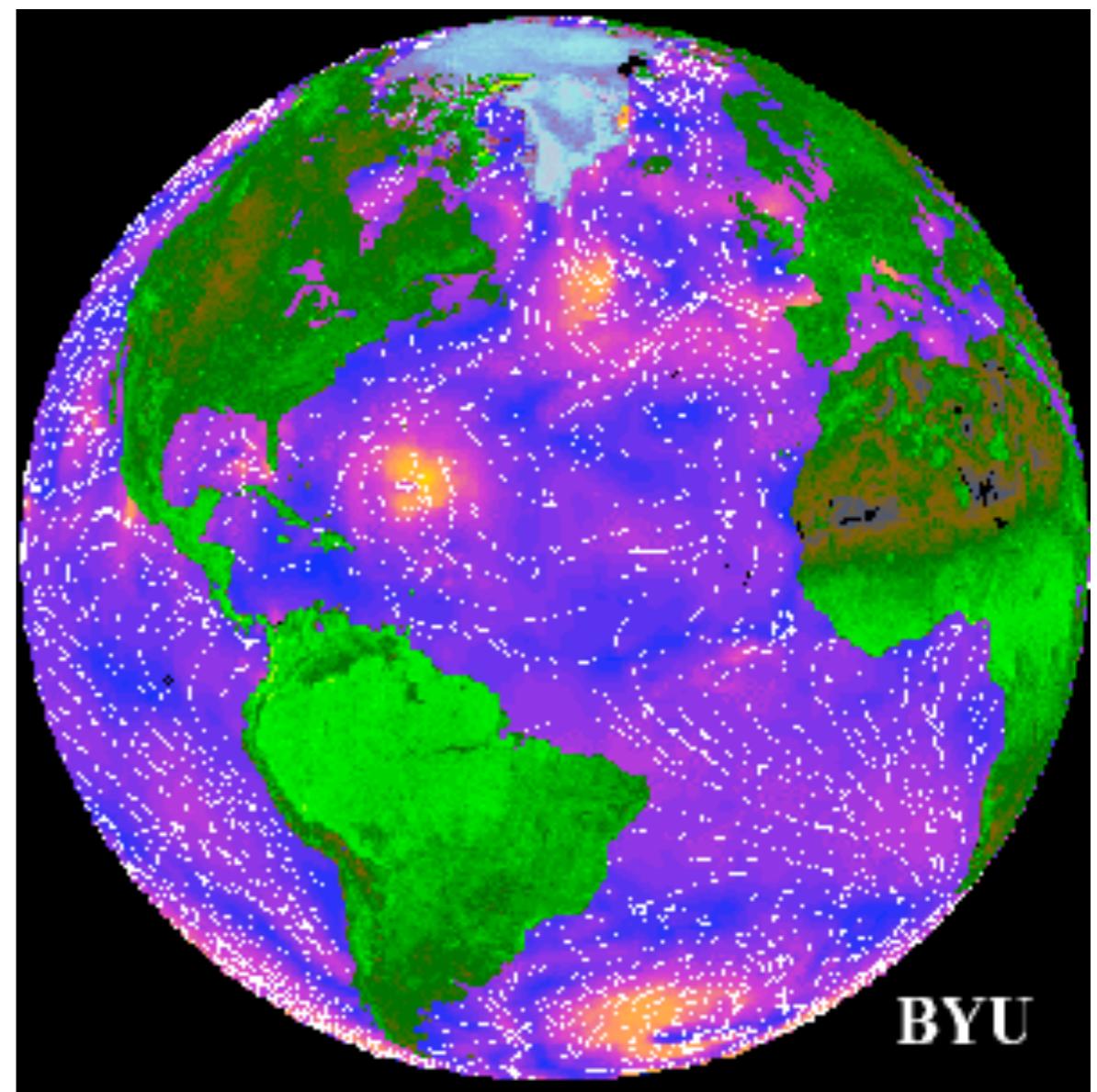
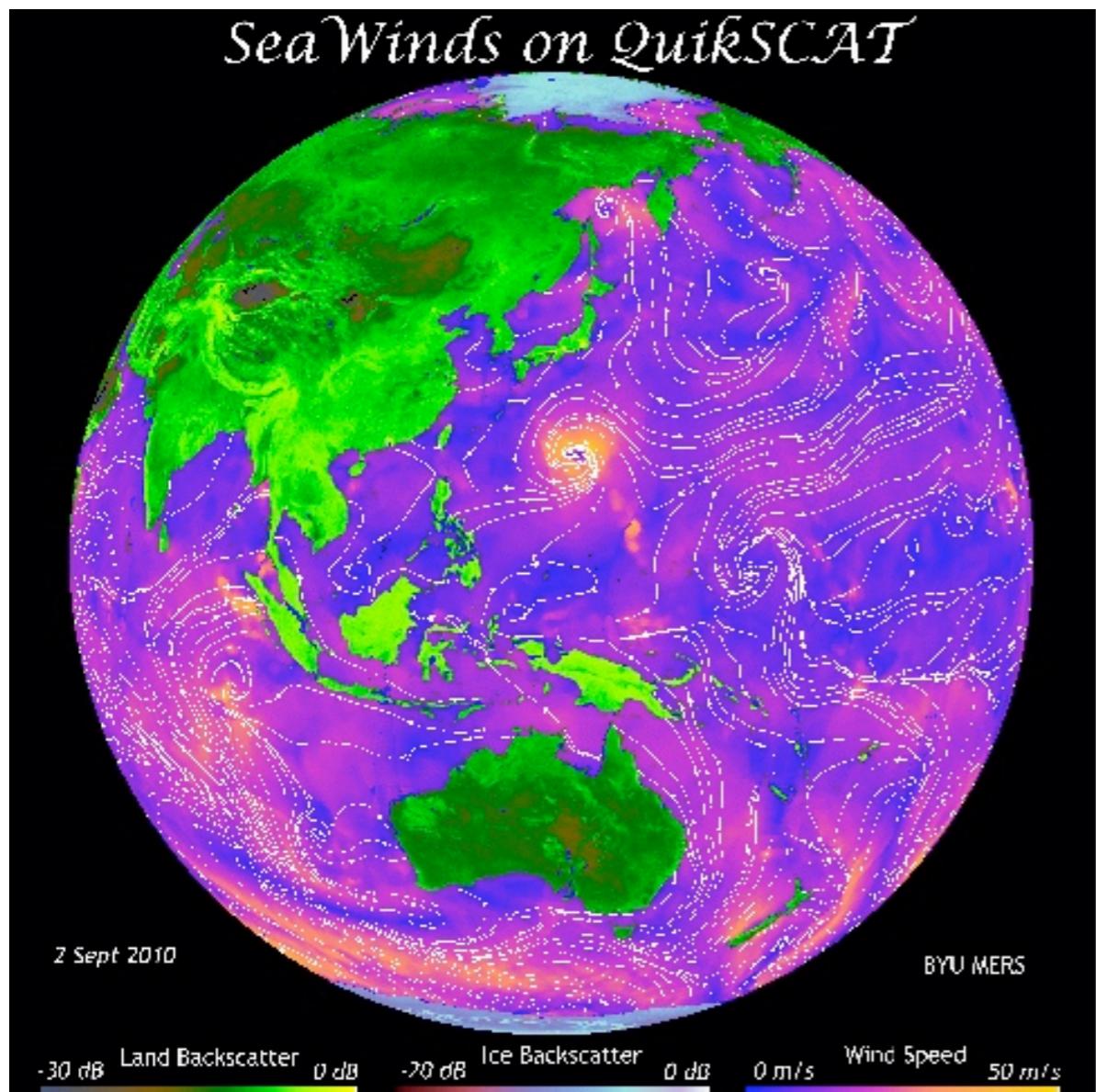


# My Roots



# My Roots

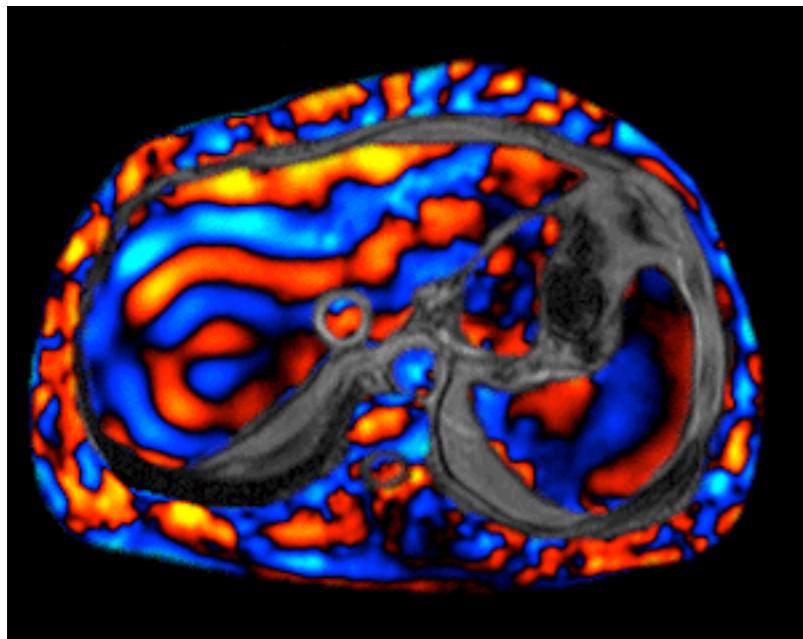
Images from BYU Mers Lab



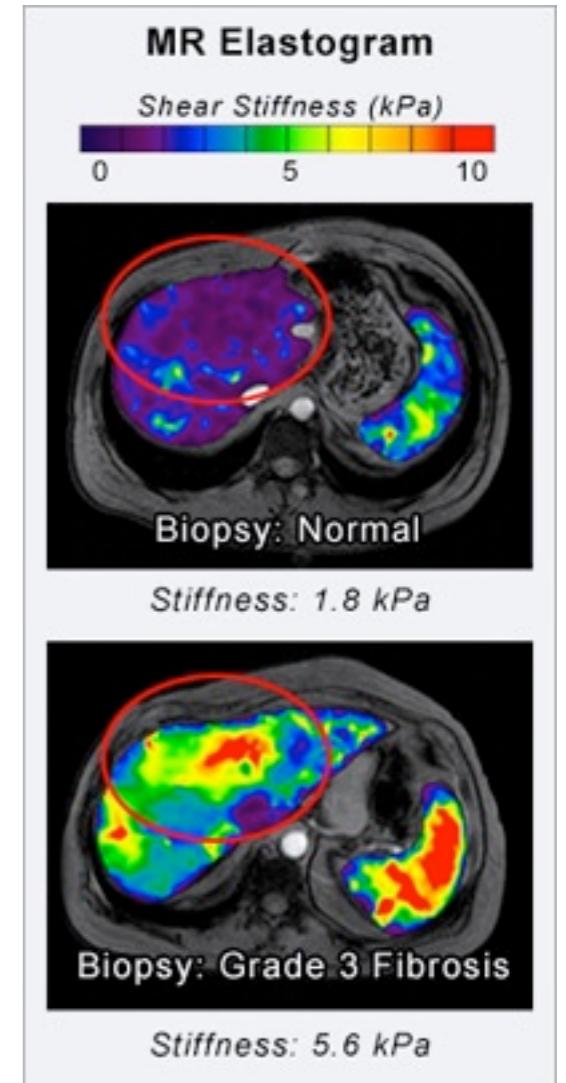
# Science led to Python

$$\rho_0 (2\pi f)^2 U_i (\mathbf{a}, f) = [C_{ijkl} (\mathbf{a}, f) U_{k,l} (\mathbf{a}, f)]_{,j}$$

Raja Muthupillai



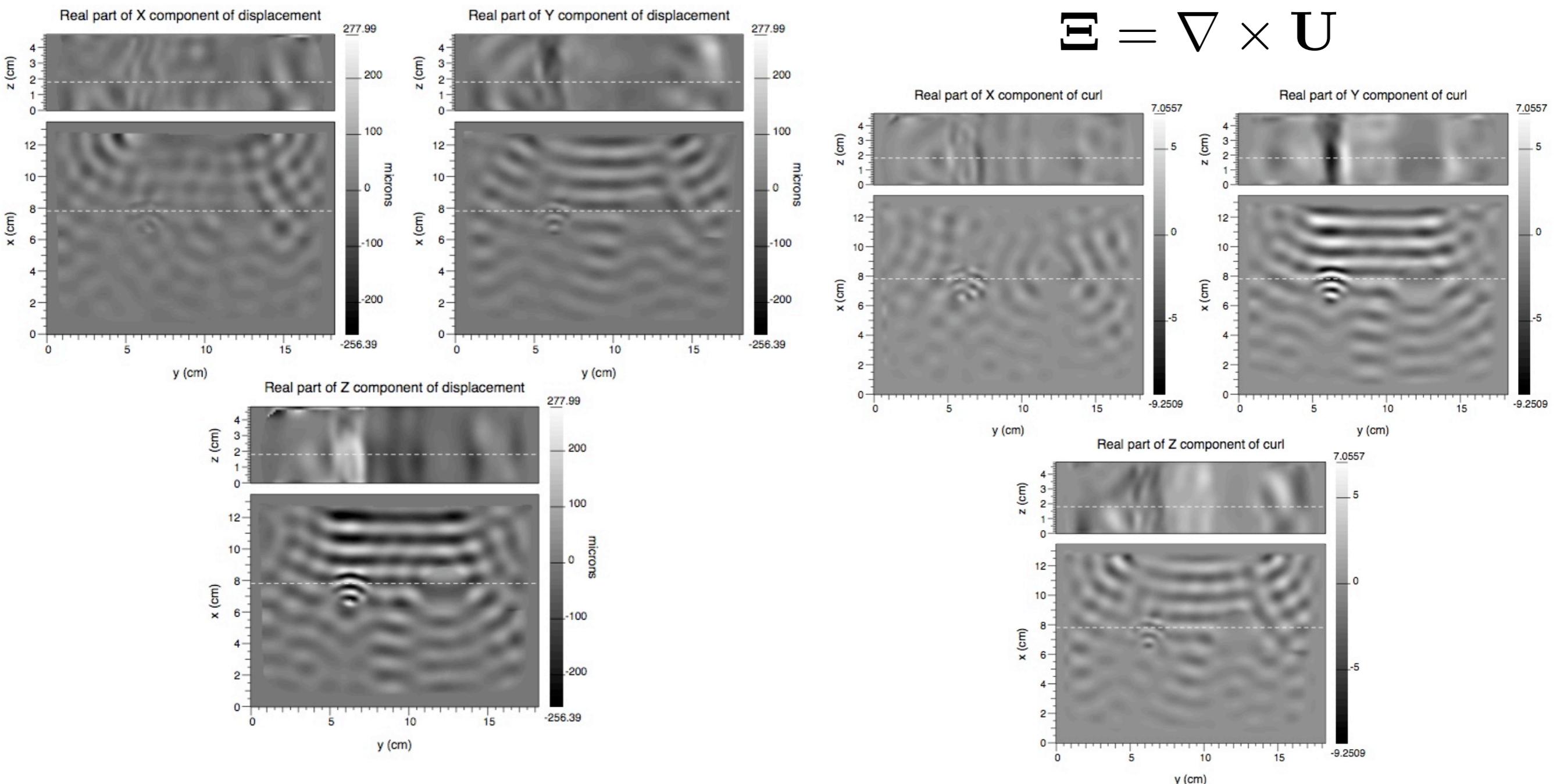
Richard Ehman  
1997



Armando Manduca



# Finding derivatives of 5-d data



# Scientist at heart



# Python origins

<http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html>

Version	Date
0.9.0	Feb. 1991
0.9.4	Dec. 1991
0.9.6	Apr. 1992
0.9.8	Jan. 1993
1.0.0	Jan. 1994
1.2	Apr. 1995
1.4	Oct. 1996
1.5.2	Apr. 1999



# Python origins

<http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html>

Version	Date
0.9.0	Feb. 1991
0.9.4	Dec. 1991
0.9.6	Apr. 1992
0.9.8	Jan. 1993
1.0.0	Jan. 1994
1.2	Apr. 1995
1.4	Oct. 1996
1.5.2	Apr. 1999



# Brief History

Person	Package	Year
	Jim Fulton Matrix Object in Python	1994
	Jim Hugunin Numeric	1995
 	Perry Greenfield, Rick White, Todd Miller Numarray	2001
	Travis Oliphant NumPy	2005

# 1999 : Early SciPy emerges

Discussions on the matrix-sig from 1997 to 1999 wanting a complete data analysis environment: Paul Barrett, Joe Harrington, Perry Greenfield, Paul Dubois, Konrad Hinsen, and others. Activity in 1998, led to increased interest in 1999.

In response on 15 Jan, 1999, I posted to matrix-sig a list of routines I felt needed to be present and began wrapping / writing in earnest. On **6 April 1999**, I announced I would be creating this uber-package which eventually became SciPy

Gaussian quadrature	5 Jan 1999
cephes 1.0	30 Jan 1999
sigtools 0.40	23 Feb 1999
Numeric docs	March 1999
cephes 1.1	9 Mar 1999
multipack 0.3	13 Apr 1999
Helper routines	14 Apr 1999
multipack 0.6 (leastsq, ode, fsolve, quad)	29 Apr 1999
sparse plan described	30 May 1999
multipack 0.7	14 Jun 1999
SparsePy 0.1	5 Nov 1999
cephes 1.2 (vectorize)	29 Dec 1999



Plotting??

Gist  
XPLOT  
DISLIN  
Gnuplot

→ Helping with f2py

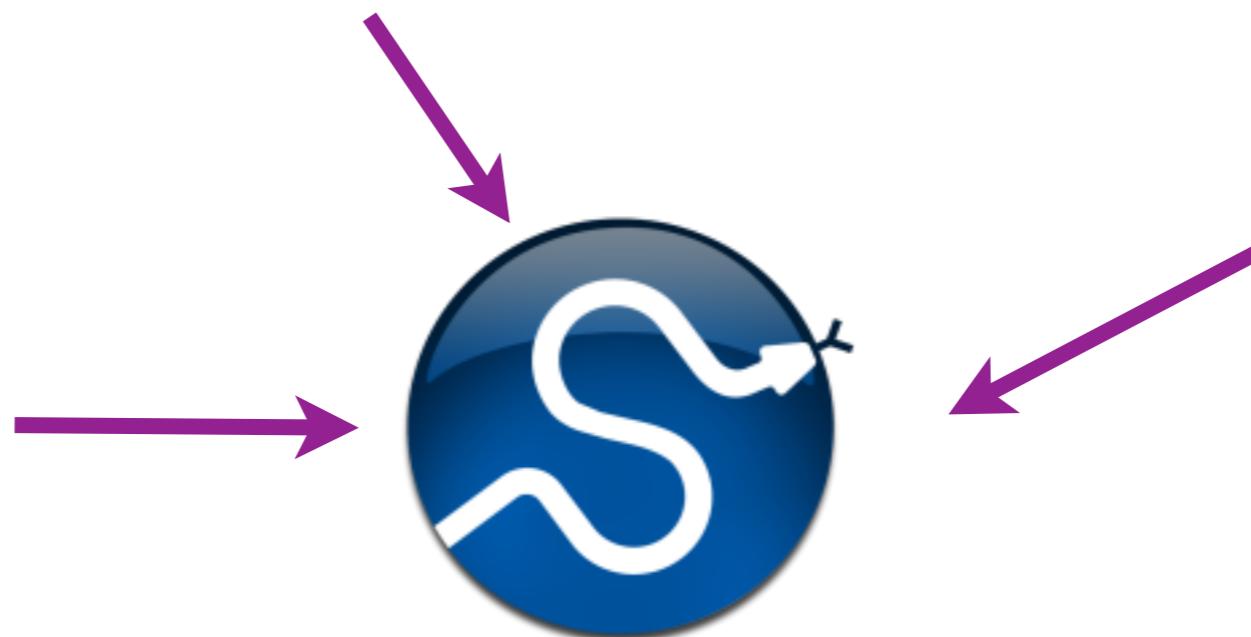
# SciPy 2001



Travis Oliphant  
optimize  
sparse  
interpolate  
integrate  
special  
signal  
stats  
fftpack  
misc



Pearu Peterson  
linalg  
interpolate  
f2py



Founded in 2001 with Travis Vaught



Eric Jones  
weave  
cluster  
GA\*



# Community effort

- Chuck Harris
- Pauli Virtanen
- David Cournapeau
- Stefan van der Walt
- Dag Sverre Seljebotn
- Robert Kern
- Warren Weckesser
- Ralf Gommers
- Mark Wiebe
- Nathaniel Smith



# Why Python for Data Analysis

- Syntax (it gets out of your way)
- Over-loadable operators
- Complex numbers built-in early
- Just enough language support for arrays
- “Occasional” programmers can grok it
- Supports multiple programming styles
- Expert programmers can also use it effectively
- Has a simple, extensible implementation
- General-purpose language --- can build a system
- Critical mass + Pandas!



# What is wrong with Python?

- Packaging is still not solved well (distribute, pip, and distutils2 don't cut it --- but look at conda!)
- Missing anonymous blocks
- The CPython run-time is aged and needs an overhaul (GIL, global variables, lack of dynamic compilation support)
- No approach to language extension (macros) except for “import hooks” (lightweight DSL need) and now IPython cell magics in notebook...
- The distraction of multiple run-times (for most data analysis you should ignore PyPy, IronPython, Jython --- they are cool projects but they don't have the ecosystem)
- Array-oriented and NumPy not used by many Python devs.



# Putting Science back in Comp Sci

- Much of the software stack is for systems programming --- C++, Java, .NET, ObjC, web
  - Complex numbers?
  - Vectorized primitives?
- Array-oriented programming has been too often replaced by Object-oriented programming --- and now Hadoop is how people are learning about “vectorization”
- Software stack for scientists is not as helpful as it should be
- Fortran is still where many scientists end up



# Array-Oriented Computing

## Example I: Fibonacci Numbers

$$\begin{aligned}f_n &= f_{n-1} + f_{n-2} \\f_0 &= 0 \\f_1 &= 1\end{aligned}$$

$$f = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \dots$$



# Common Python approaches

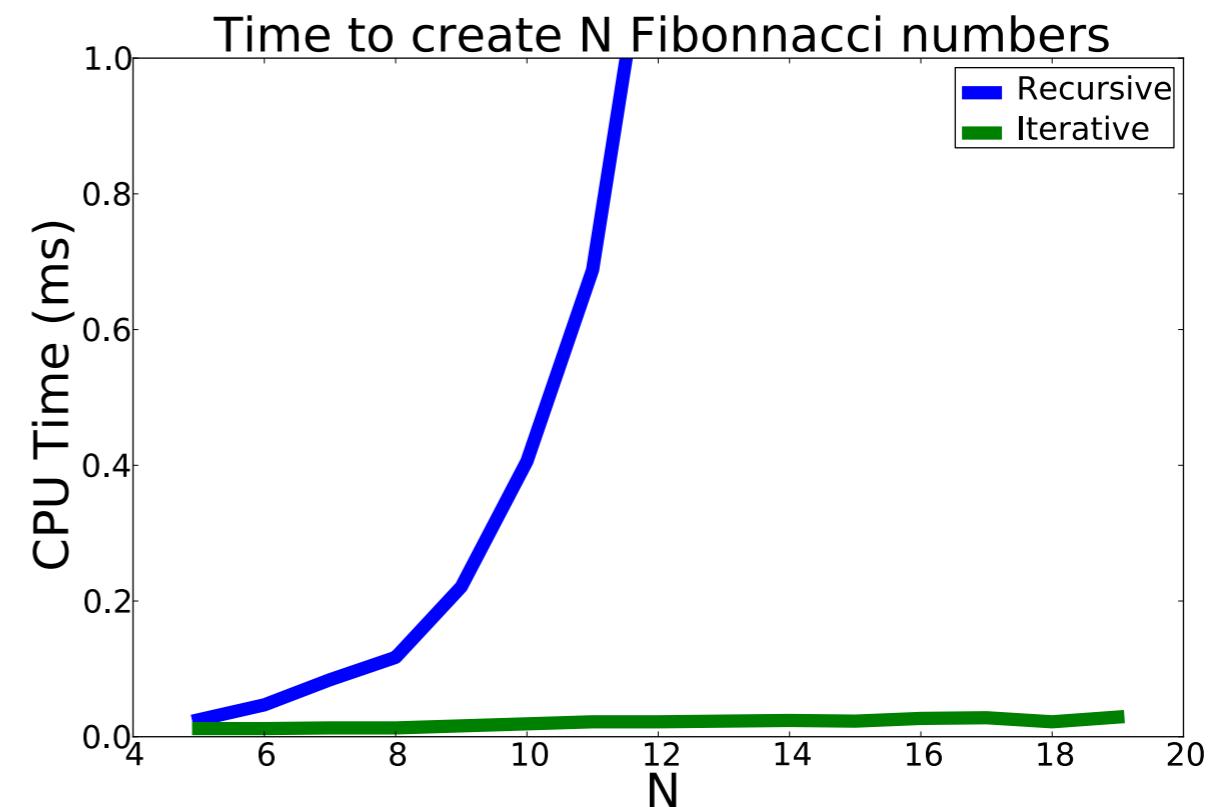
## Recursive

```
def _fib(n):
    if n <= 0:
        return 0
    if n == 1:
        return 1
    return fib(n-1) + fib(n-2)
```

```
def fib(N):
    return [_fib(n) for n in xrange(N)]
```

## Iterative

```
def fib1(N):
    result = [0,1]
    for k in range(2,N):
        result.append(result[k-1] + result[k-2])
    return result
```



Algorithm matters!!

# Array-oriented approaches

## Using LFilter

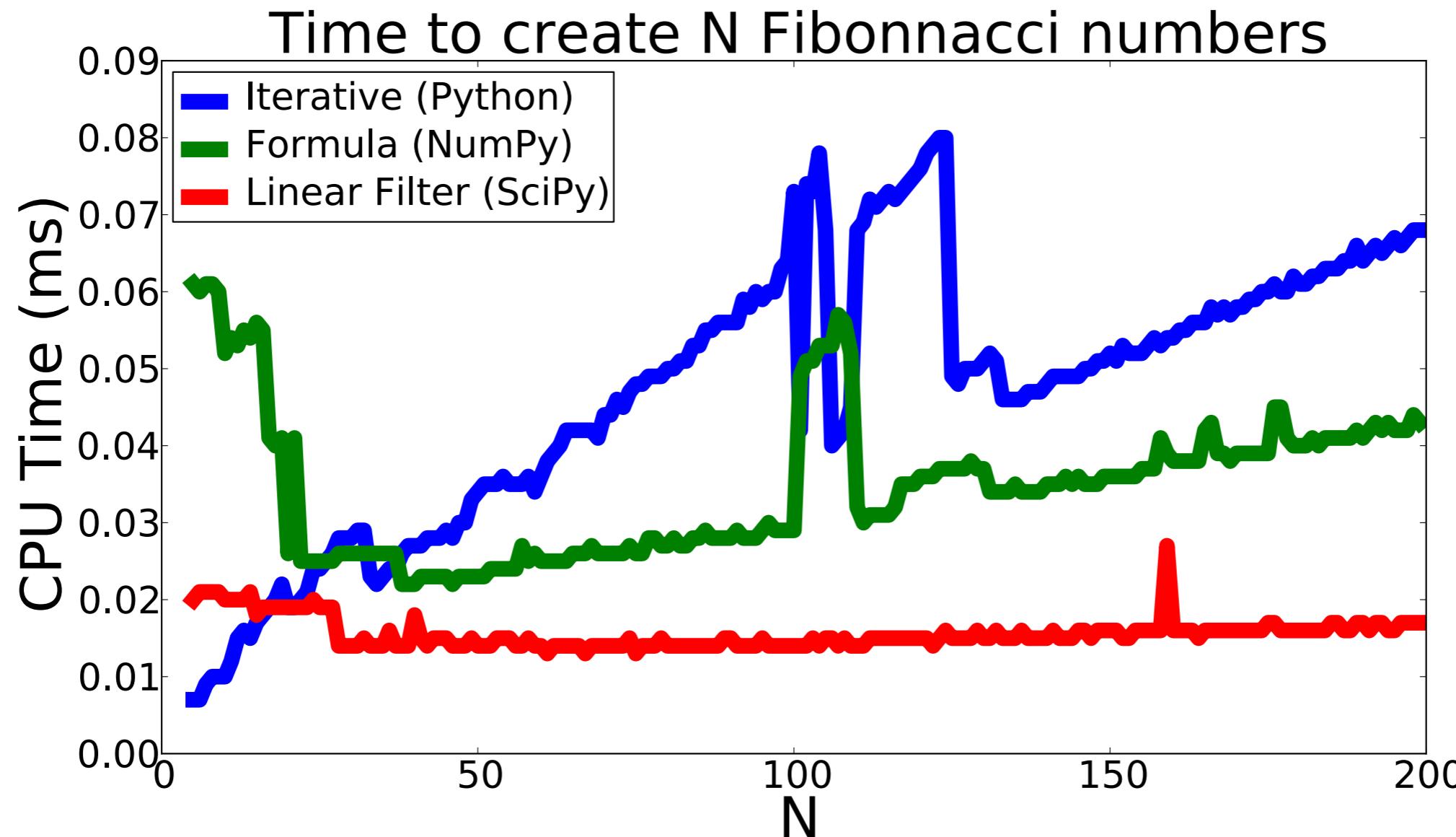
```
from scipy.signal import lfilter
from numpy import zeros
b = array([1.0])
a = array([1., -1, -1])
zi = array([0, 1.0])
def fib3a(N):
    y, zf = lfilter(b, a,
                     zeros(N,dtype=float),zi=zi)
    return y
```

## Using Formula

```
from numpy import roots, arange
r1, r2 = roots([1,-1,-1])
C = 1.0/(r1-r2)
def fib2a(N):
    n = arange(N,dtype=float)
    return C*(r1**n - r2**n)
```



# Array-oriented approaches

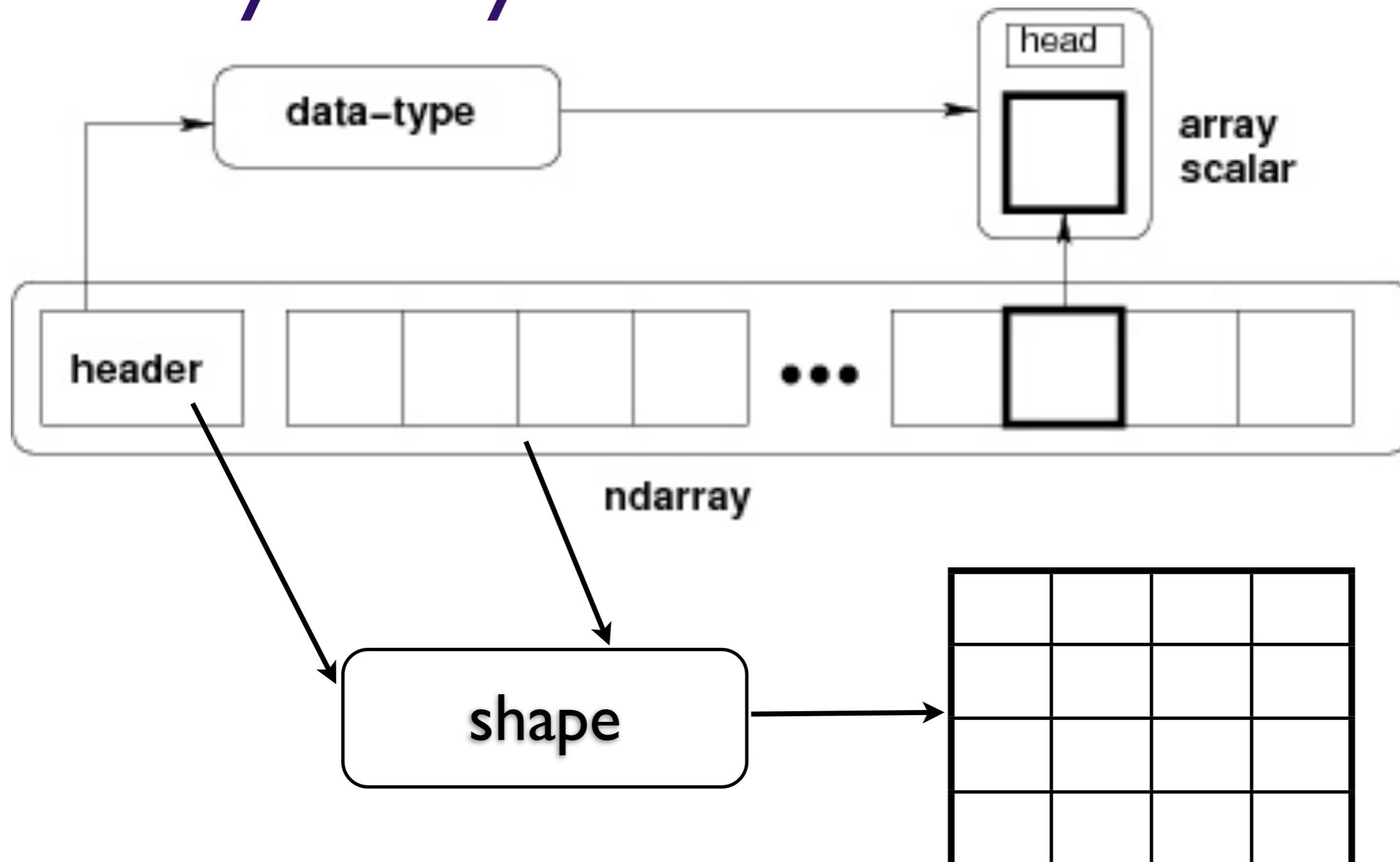


# NumPy: an Array-Oriented Extension

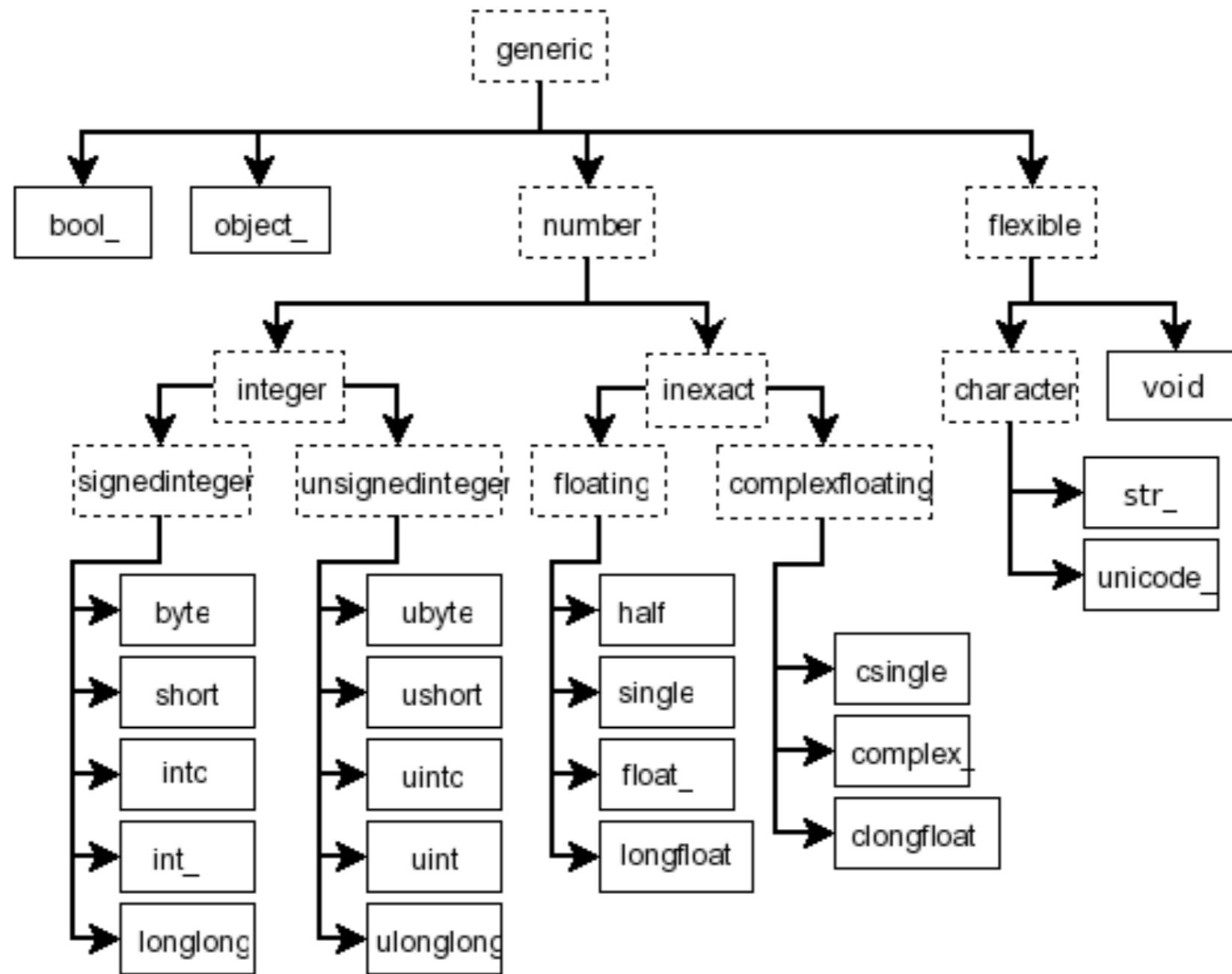
- Data: the array object
  - slicing and shaping
  - data-type map to Bytes
- Fast Math:
  - vectorization
  - broadcasting
  - aggregations



# NumPy Array



# Broad Type Support



# NumPy Slicing

```
>>> a[:,1]
```

```
>>> a[::2, ::3]
```

```
>>> a[1,2:5]
```

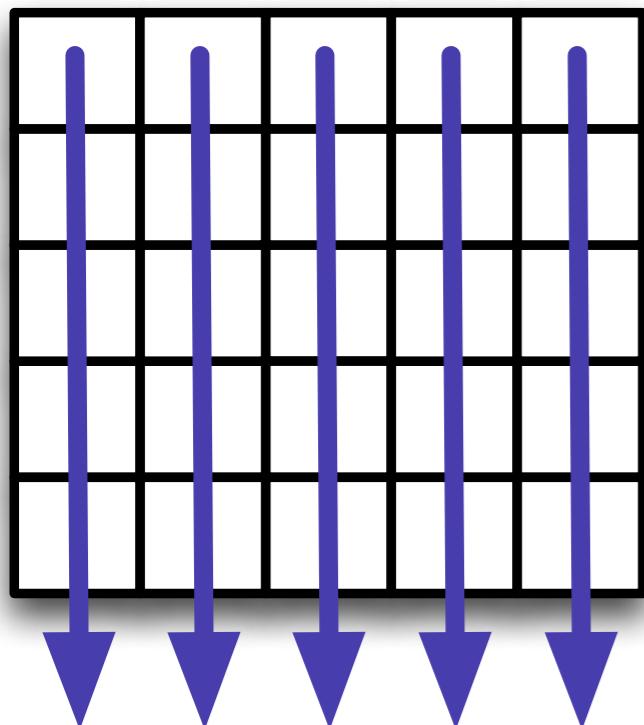
```
>>> a[3:5, 4:6]
```

1	2	3	4	5	6
11	12	13	14	15	16
31	32	33	34	35	36
41	42	43	44	45	46
51	52	53	54	55	56
61	62	63	64	65	66

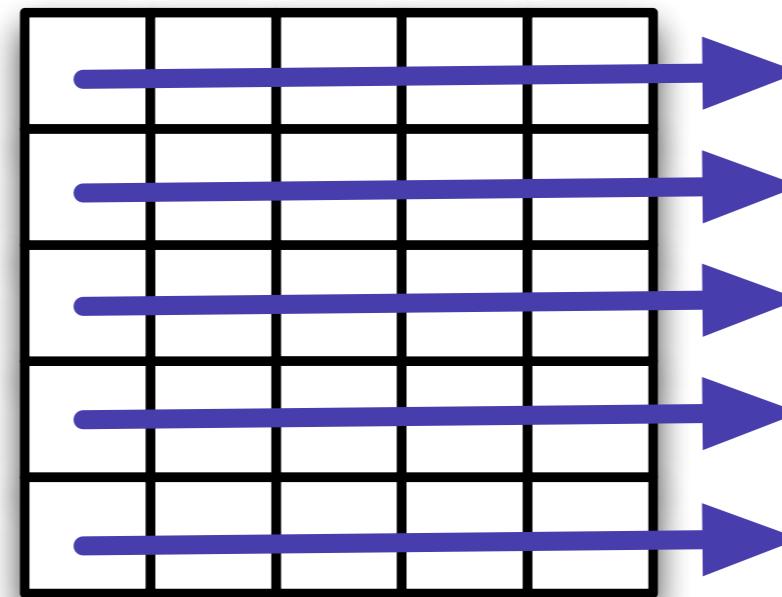
# NumPy axis argument (e.g. reduction)

```
a = rand(10,10)
a.std(axis=0)    --- reduce along 0th dimension
a.std(axis=1)    --- reduce along 1st dimension
a.std()          --- reduce along all dimensions
```

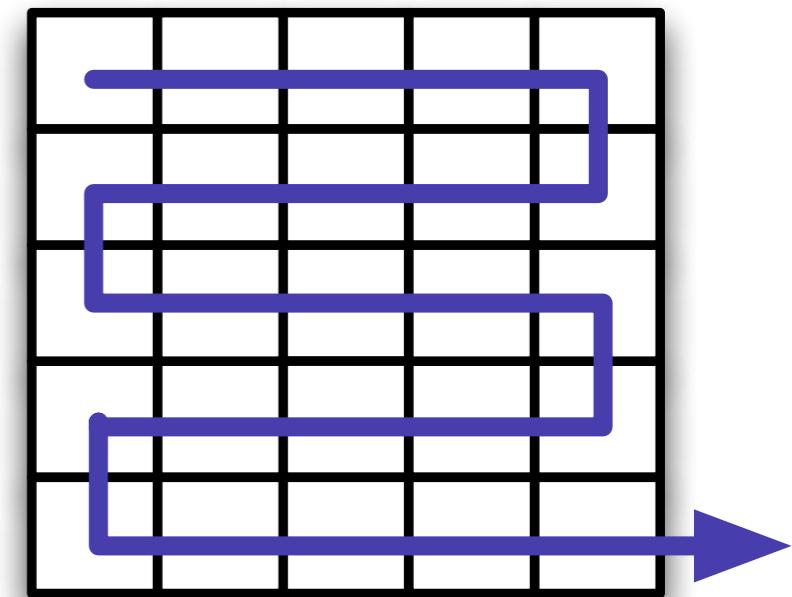
axis=0



axis=1



axis=None



# Broadcasting

```
>>> x  
array([10, 20])  
>>> x.shape  
(2,)
```



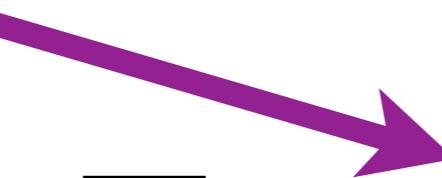
```
>>> y  
array([[1],  
       [2],  
       [3]])  
>>> y.shape  
(3, 1)
```



```
>>> x[np.newaxis,:]  
array([[10, 20]])  
>>> x[np.newaxis,:].shape  
(1, 2)
```



```
>>> x+y  
array([[11, 21],  
       [12, 22],  
       [13, 23]])  
>>> (x+y).shape  
(3, 2)
```



# Zen of NumPy

- strided is better than scattered
- contiguous is better than strided
- descriptive is better than imperative
- array-oriented is better than object-oriented
- broadcasting is a great idea
- vectorized is better than an explicit loop
- unless it's too complicated --- then use Numba
- think in higher dimensions



# Benefits of Array-oriented

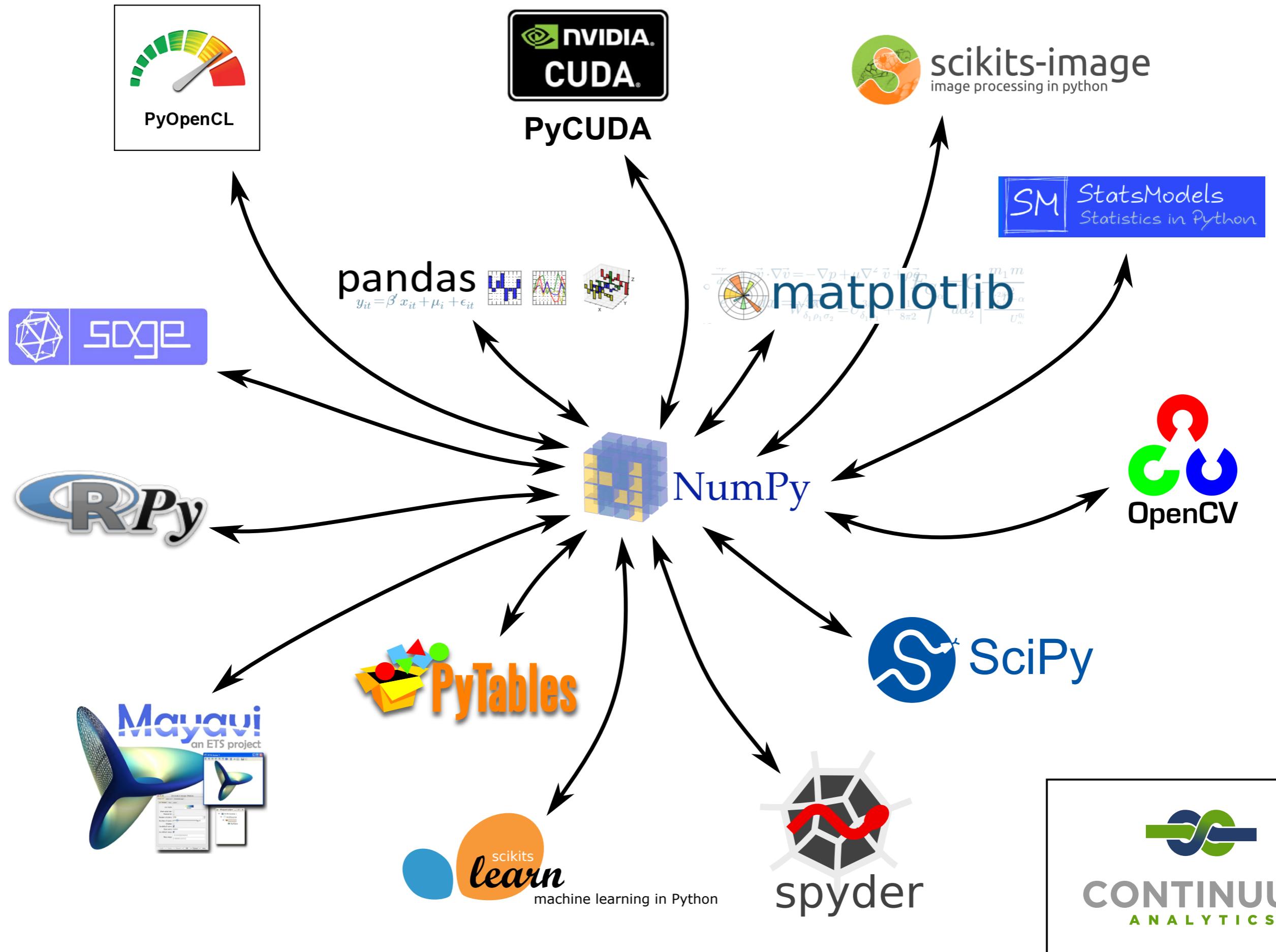
- Many technical problems are naturally array-oriented (easy to vectorize)
- Algorithms can be expressed at a high-level
- These algorithms can be parallelized more simply (quite often much information is lost in the translation to typical “compiled” languages)
- Array-oriented algorithms map to modern hard-ware caches and pipelines.

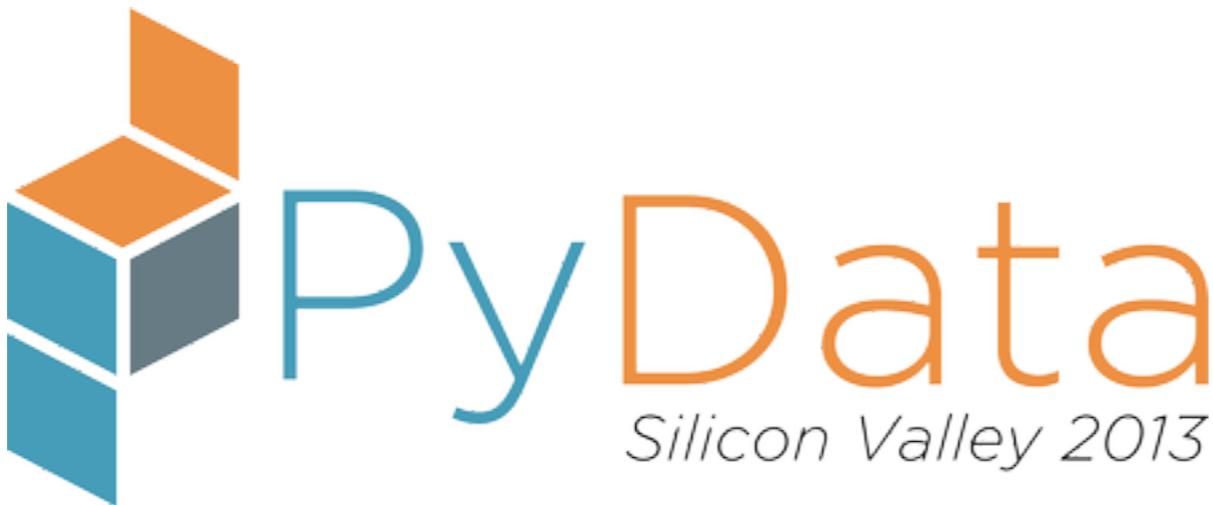


# What is good about NumPy?

- Array-oriented: slicing and broadcasting
- Extensive Dtype System (including structures)
- C-API
- Simple to understand data-structure
- Memory mapping
- Syntax support from Python
- Large community of users and packages that build on it
- Easy to interface C/C++/Fortran code







<http://www.pydata.org/>

*Changing the way scientists, engineers, and analysts perceive big data*

**TUTORIALS:** March 18 | **MAIN CONFERENCE:** March 19-20

**Santa Clara Convention Center**  
Santa Clara, CA

*Sponsorship available. [Learn more.](#)*

## Sponsors



J.P.Morgan



Python Weekly

Software Developer's JOURNAL

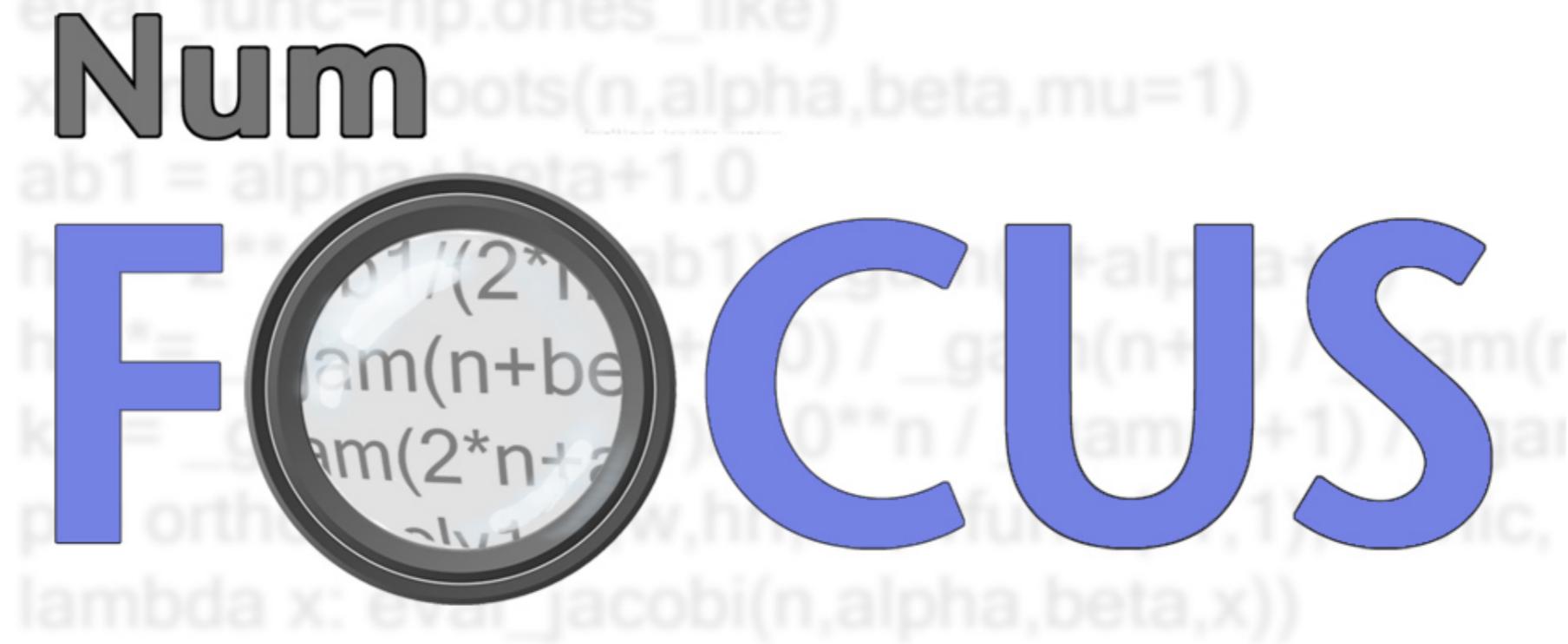
Inside Analysis



# NumFOCUS

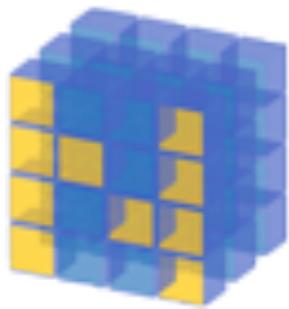
[www.numfocus.org](http://www.numfocus.org)

501(c)3 Public Charity



# NumFOCUS

## Core Projects

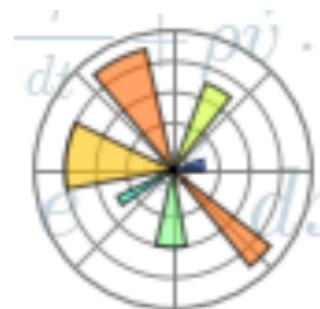


**NumPy**



**SciPy**

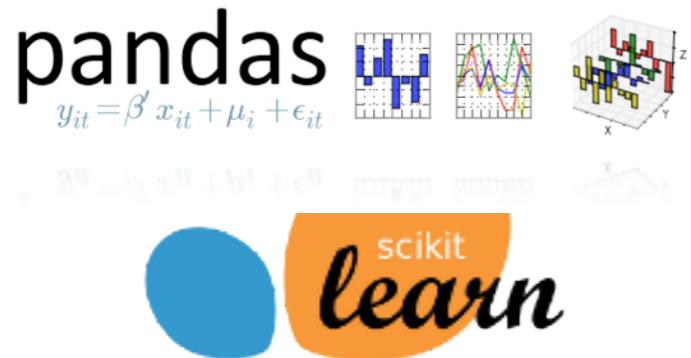
**IP[y]:**



**Matplotlib**



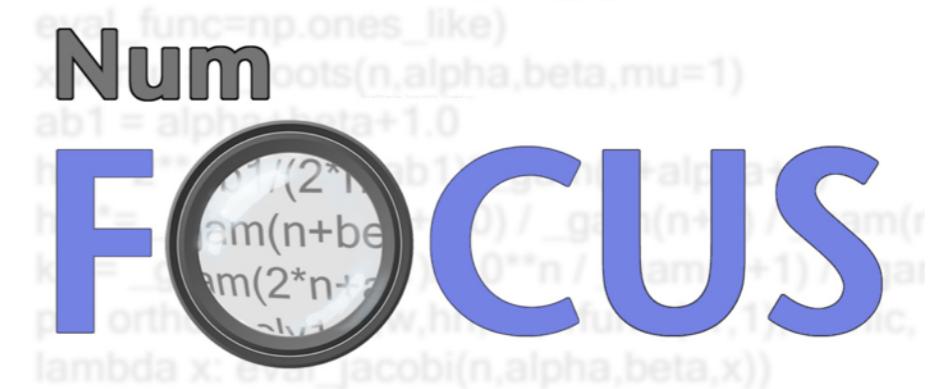
**SymPy**



**Scikits Image**



# NumFOCUS Mission

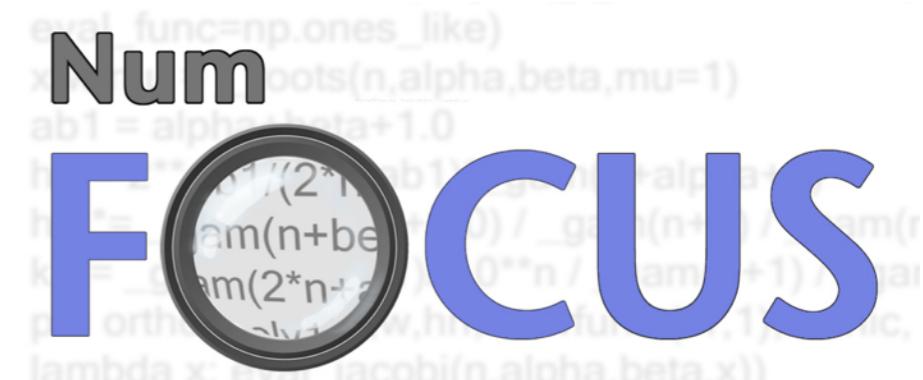


- Sponsor development of high-level languages and libraries for science
- Foster teaching of array-oriented and higher-order computational approaches and applied computational science
- Promote the use of open code in science and encourage reproducible and accessible research



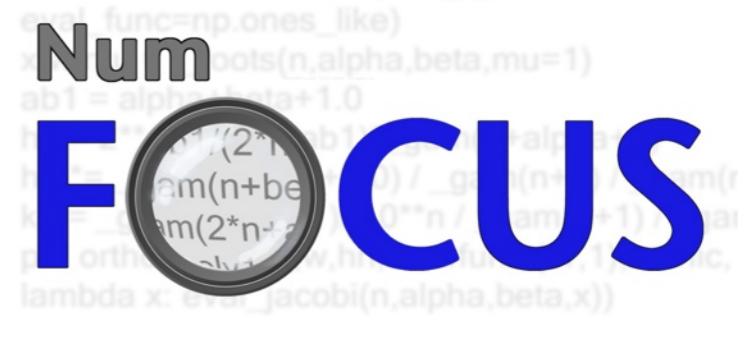
# NumFOCUS Activities

- Sponsor sprints and conferences
- Provide scholarships and grants
- Provide bounties and prizes for code development
- Pay for freely-available documentation and basic course development
- Equipment grants
- Sponsor BootCamps
- Raise funds from industries using open source high-level languages



# NumFOCUS

- Directors
  - Perry Greenfield
  - Jarrod Millman
  - Travis Oliphant
  - Fernando Perez
  - Andy Terrel
  - Didrik Pinte
  - Rolf Gommers
  - Emanuelle Gouillart



Apply now to become  
a member and support the  
Foundation!