Python's Data Science Stack

Jake VanderPlas @jakevdp
JSM, July 31, 2016

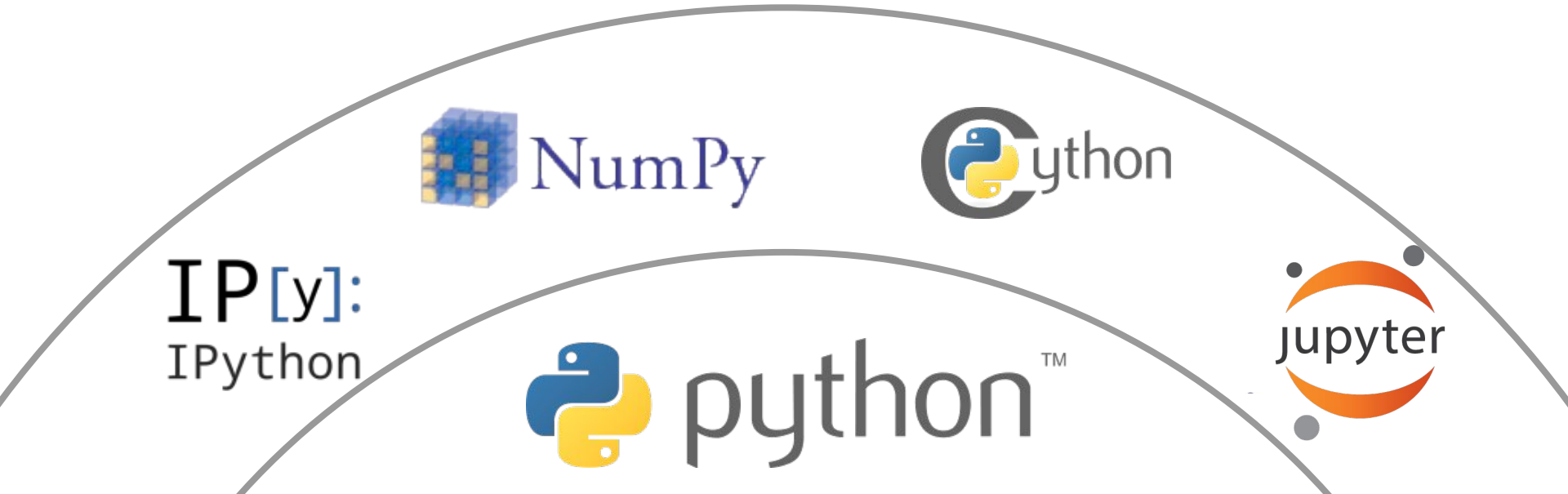
Python is *not* a statistical computing language!

Python is *not* a statistical computing language!

. . . and this may be its greatest strength as a language for statistical computing.

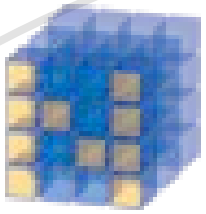
A Quick Tour of Python's Data Science Stack

Python's Data Science Stack


NumPy
Cython
IP[y]:
IPython
python™
jupyter

NumPy = Numerical Python

Efficient array storage, manipulation, and computation



NumPy

ython

IP[y]:
IPython

python™

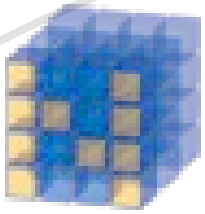
jupyter

NumPy = Numerical Python

Efficient array storage, manipulation, and computation

In [1]:
```
import numpy as np

# Create a 5x5 uniform random matrix
M = np.random.rand(5, 5)

# Compute the SVD
U, S, VT = np.linalg.svd(M)
print(S)
```

```
[ 2.46102945  0.94542853  0.53550015  0.20705388  0.13071452]
```



NumPy

Cython

IP[y]:
IPython

python™

jupyter

NumPy

ython


IP[y]:
IPython

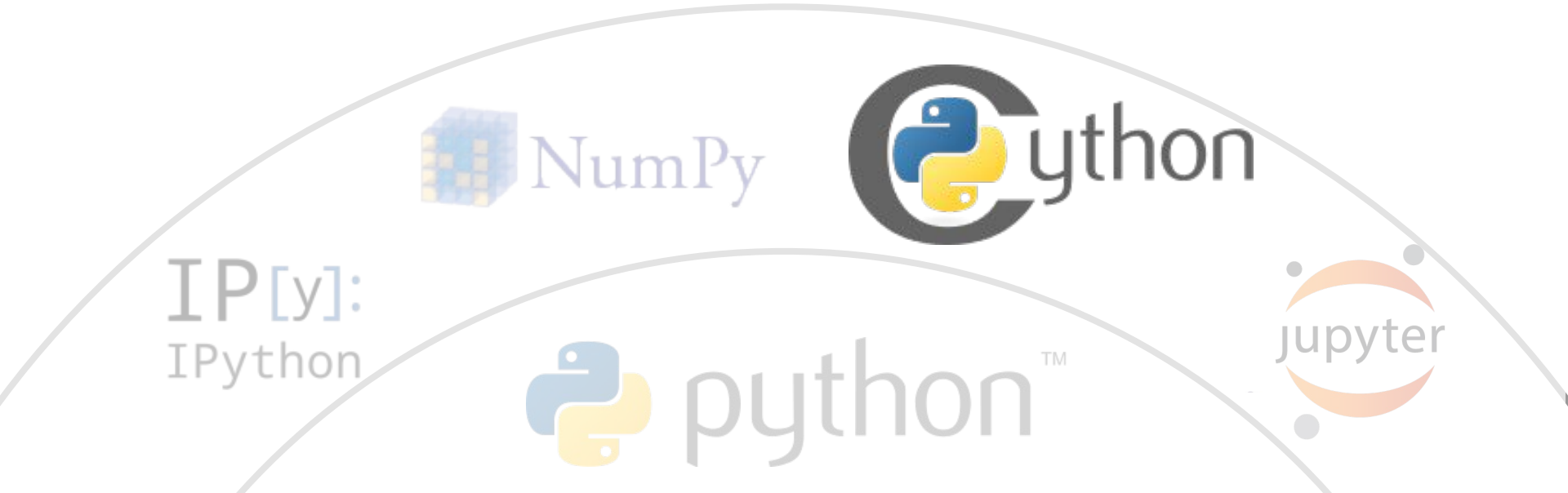
python™


jupyter

Cython = C + Python

Super-set of the Python language that allows easy interfacing with C & Fortran libraries (e.g. BLAS, LAPACK, etc.) and also fast Python code.

Drives many of the packages in the data science stack.



NumPy

ython


IP[y]:
IPython

python™


jupyter

IPython / Jupyter

Terminal, development environment, Notebooks, and more for efficient use of Python in day-to-day work

IPython / Jupyter

localhost:8888/notebooks/Jaynes-Cumming-model.ipynb

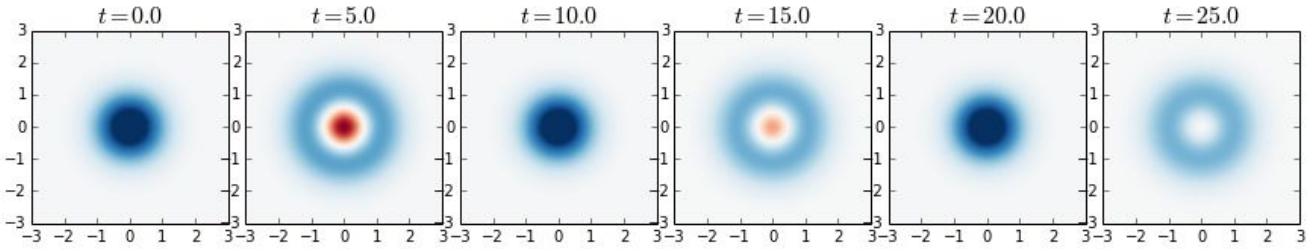
jupyter Jaynes-Cumming-model (unsaved changes)

File Edit View Insert Cell Kernel Help Python 3

Markdown CellToolbar

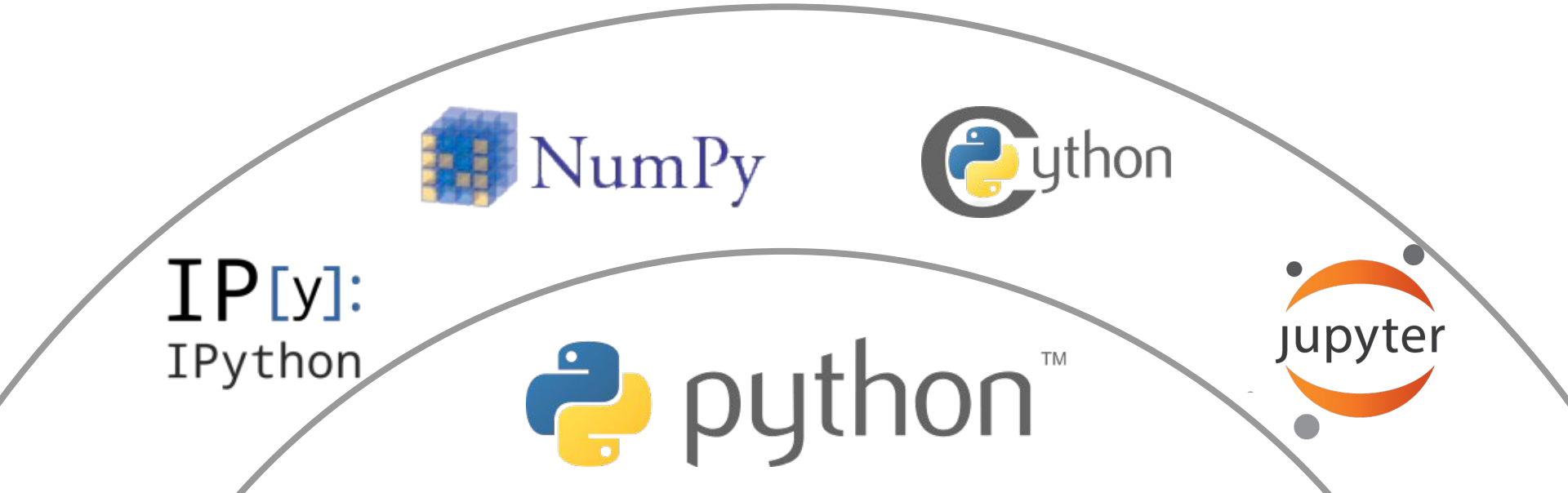
Alternative view of the model

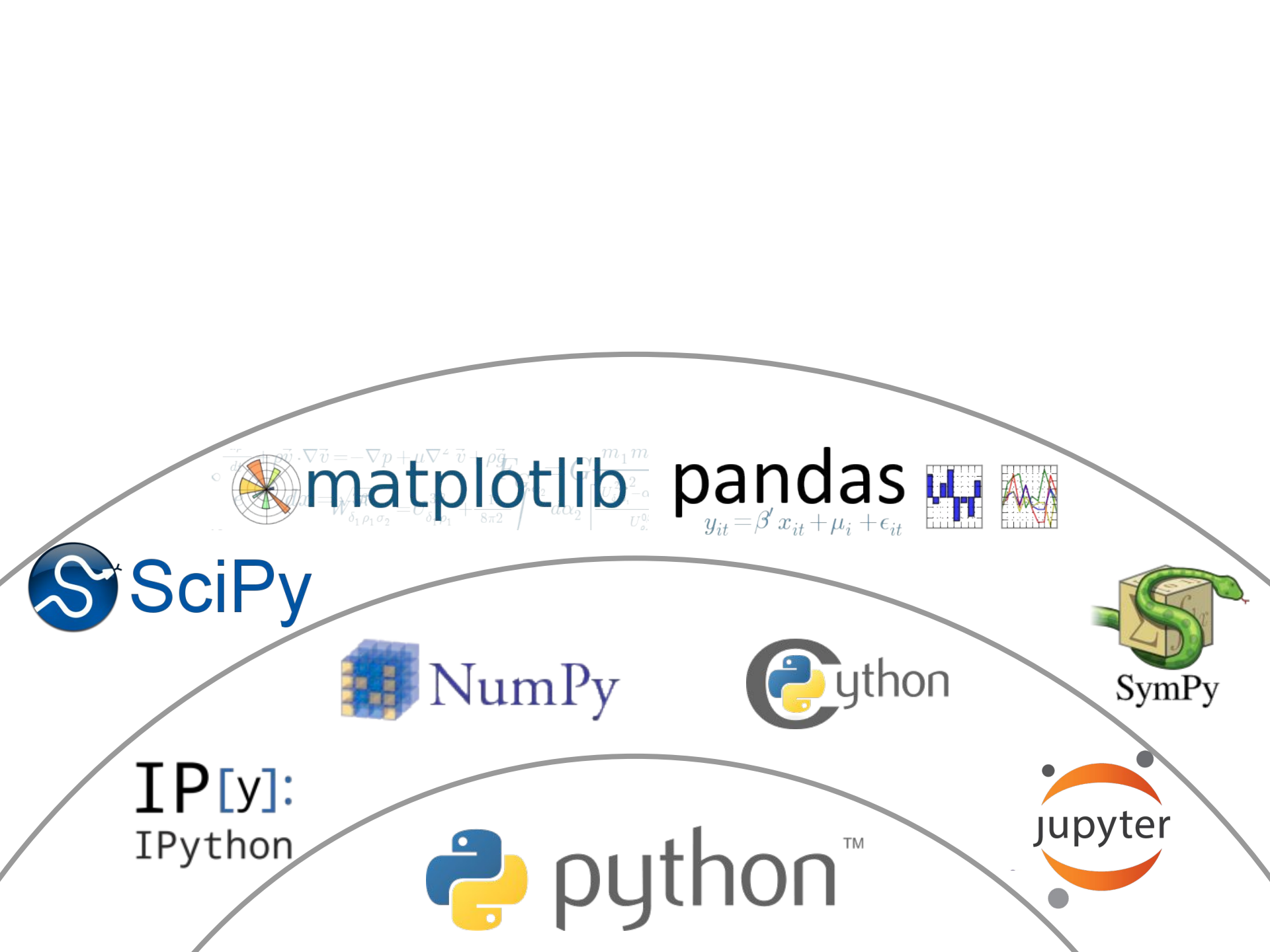
In [22]:
```
t_idx = where([tlist == t for t in [0.0, 5.0, 10, 15, 20, 25]])[1]
rho_list = array(output.states)[t_idx]

fig_grid = (2, len(rho_list)*2)
fig = plt.figure(figsize=(2.5*len(rho_list),5))

for idx, rho in enumerate(rho_list):
    rho_cavity = ptrace(rho, 0)
    W = wigner(rho_cavity, xvec, xvec)
    ax = plt.subplot2grid(fig_grid, (0, 2*idx), colspan=2)
    ax.contourf(xvec, xvec, W, 100, norm=mpl.colors.Normalize(-.25,.25), cmap=plt.get_cmap('RdBu'))
    ax.set_title(r"$t = %.1f$" % tlist[t_idx][idx], fontsize=16)

# plot the cavity occupation probability in the ground state
ax = plt.subplot2grid(fig_grid, (1, 1), colspan=(fig_grid[1]-2))
ax.plot(tlist, n_c, label="Cavity")
ax.plot(tlist, n_a, label="Atom excited state")
ax.legend()
ax.set_xlabel('Time')
ax.set_ylabel('Occupation probability');
```



IPython

python™


NumPy
Cython
IP[y]:
IPython
python™
jupyter


matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

SciPy

Provides an interface to common scientific computing Tasks, including wrappers of many NetLib packages.

SciPy

Provi
Tasks

List from http://docs.scipy.org/doc/scipy/reference/

- Special functions (**scipy.special**)
- Integration (**scipy.integrate**)
- Optimization (**scipy.optimize**)
- Interpolation (**scipy.interpolate**)
- Fourier Transforms (**scipy.fftpack**)
- Signal Processing (**scipy.signal**)
- Linear Algebra (**scipy.linalg**)
- Sparse Eigenvalue Problems with ARPACK
- Compressed Sparse Graph Routines (**scipy.sparse.csgraph**)
- Spatial data structures and algorithms (**scipy.spatial**)
- Statistics (**scipy.stats**)
- Multidimensional image processing (**scipy.ndimage**)
- File IO (**scipy.io**)

IP[y]:
IPython

python™

jupyter


matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

Sympy

Library for symbolic computation: algebraic operations, differentiation & integration, optimization, etc.

Sympy

Library for symbolic computation: algebraic operations, differentiation & integration, optimization, etc.

Polynomials and rational functions

SymPy does not expand brackets automatically. The function expand is used for this.

In [6]:
```
a=(x+y-z)**6
a
```

Out[6]: $(x+y-z)^6$

In [7]:
```
a=expand(a)
a
```

Out[7]: $$x^6+6x^5y-6x^5z+15x^4y^2-30x^4yz+15x^4z^2+20x^3y^3-60x^3y^2z+60x^3yz^2-20x^3z^3$$
$$+15x^2y^4-60x^2y^3z+90x^2y^2z^2-60x^2yz^3+15x^2z^4+6xy^5-30xy^4z+60xy^3z^2$$
$$-60xy^2z^3+30xyz^4-6xz^5+y^6-6y^5z+15y^4z^2-20y^3z^3+15y^2z^4-6yz^5+z^6$$


matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

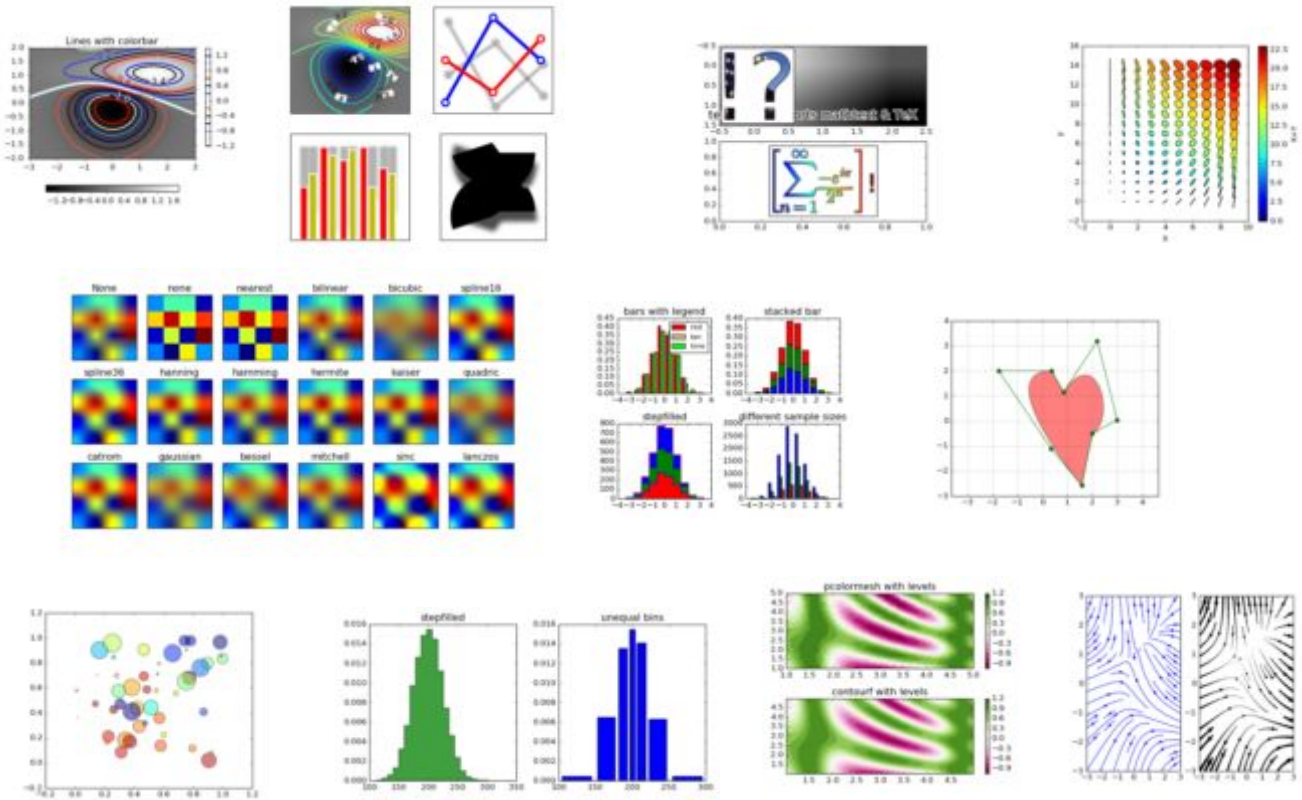
matplotlib

Matlab-inspired plotting and visualization

matplotlib

Matlab-inspired plotting and visualization

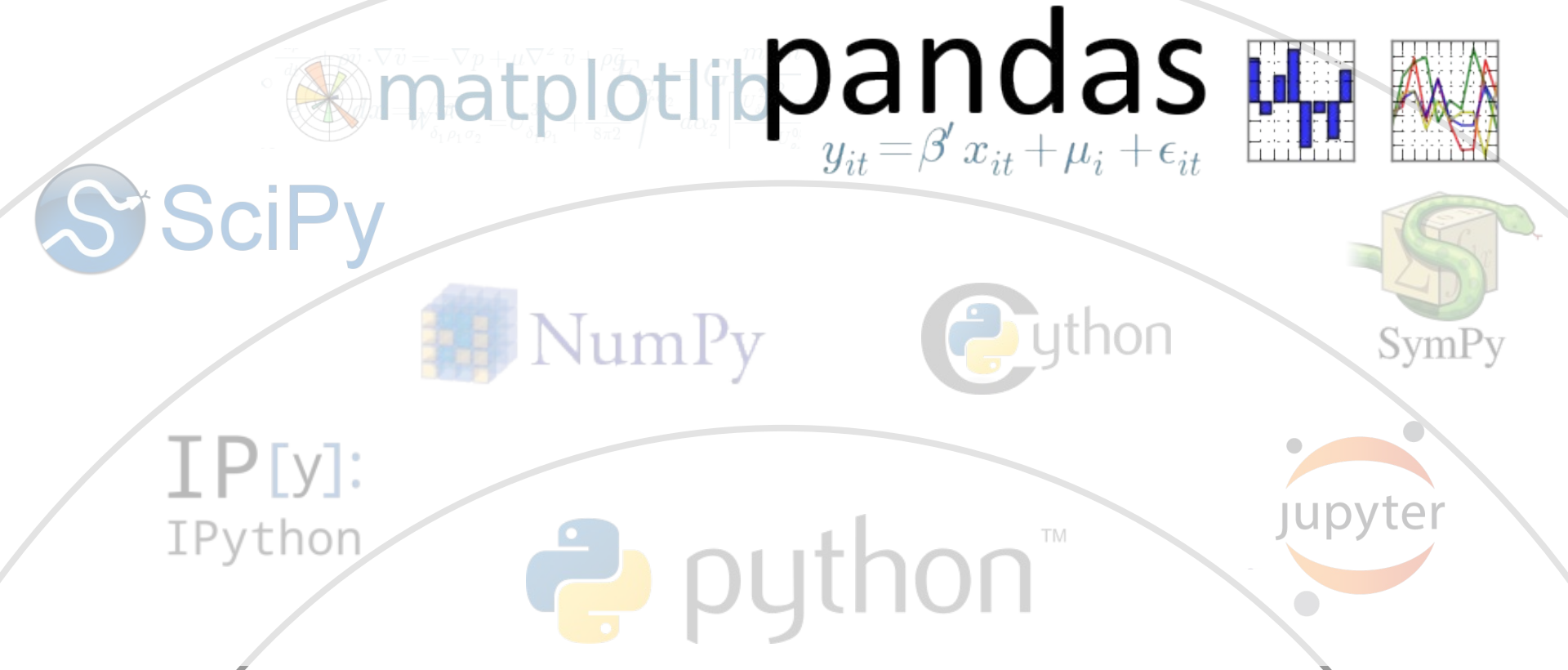
From http://matplotlib.org/gallery.html




matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

Pandas

R-inspired DataFrames & associated functionality (data munging & cleaning, group-by & transformations, and much more)



Pa

```
In [1]: import pandas as pd
        data = pd.read_csv('iris.csv')
        data.head()
```

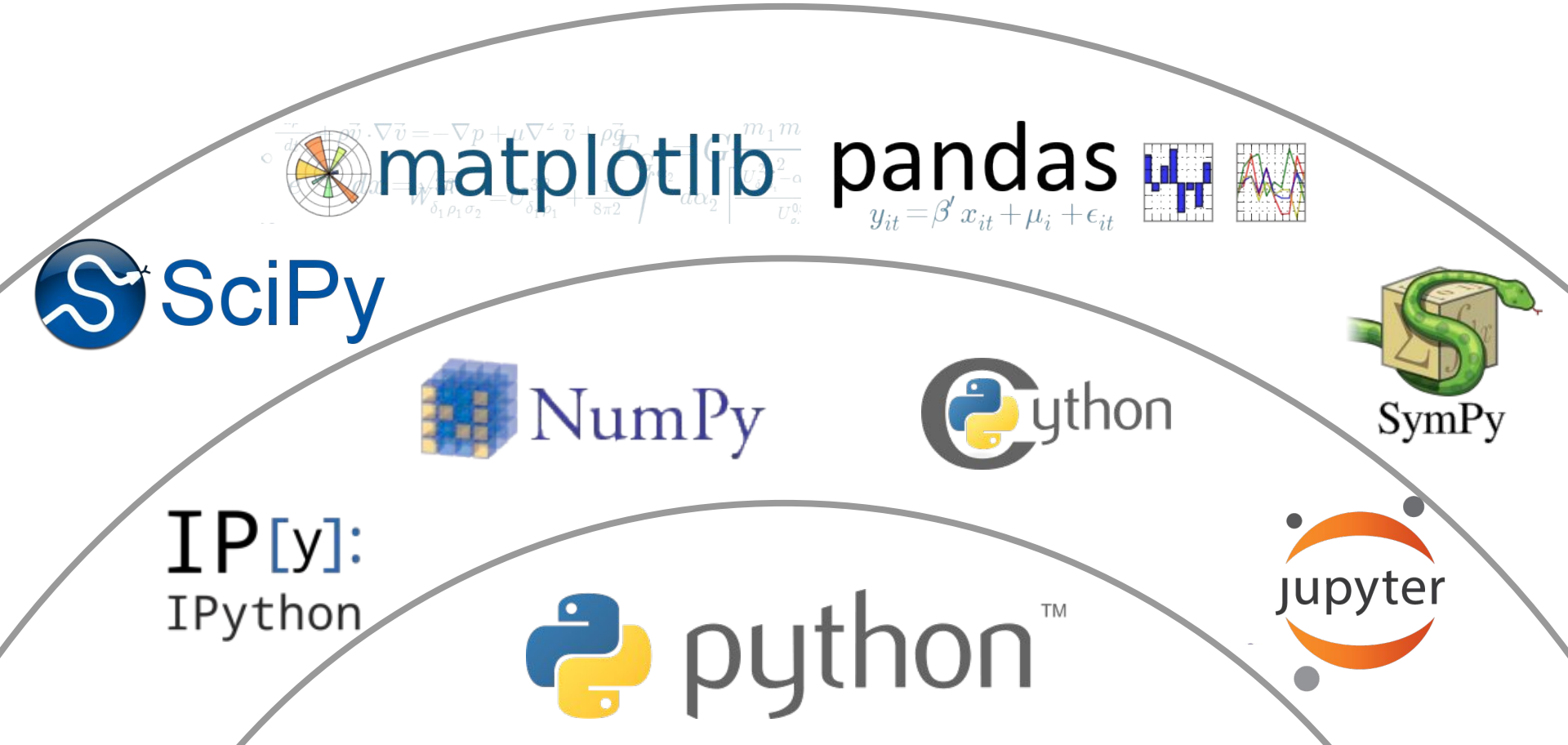
Out[1]:

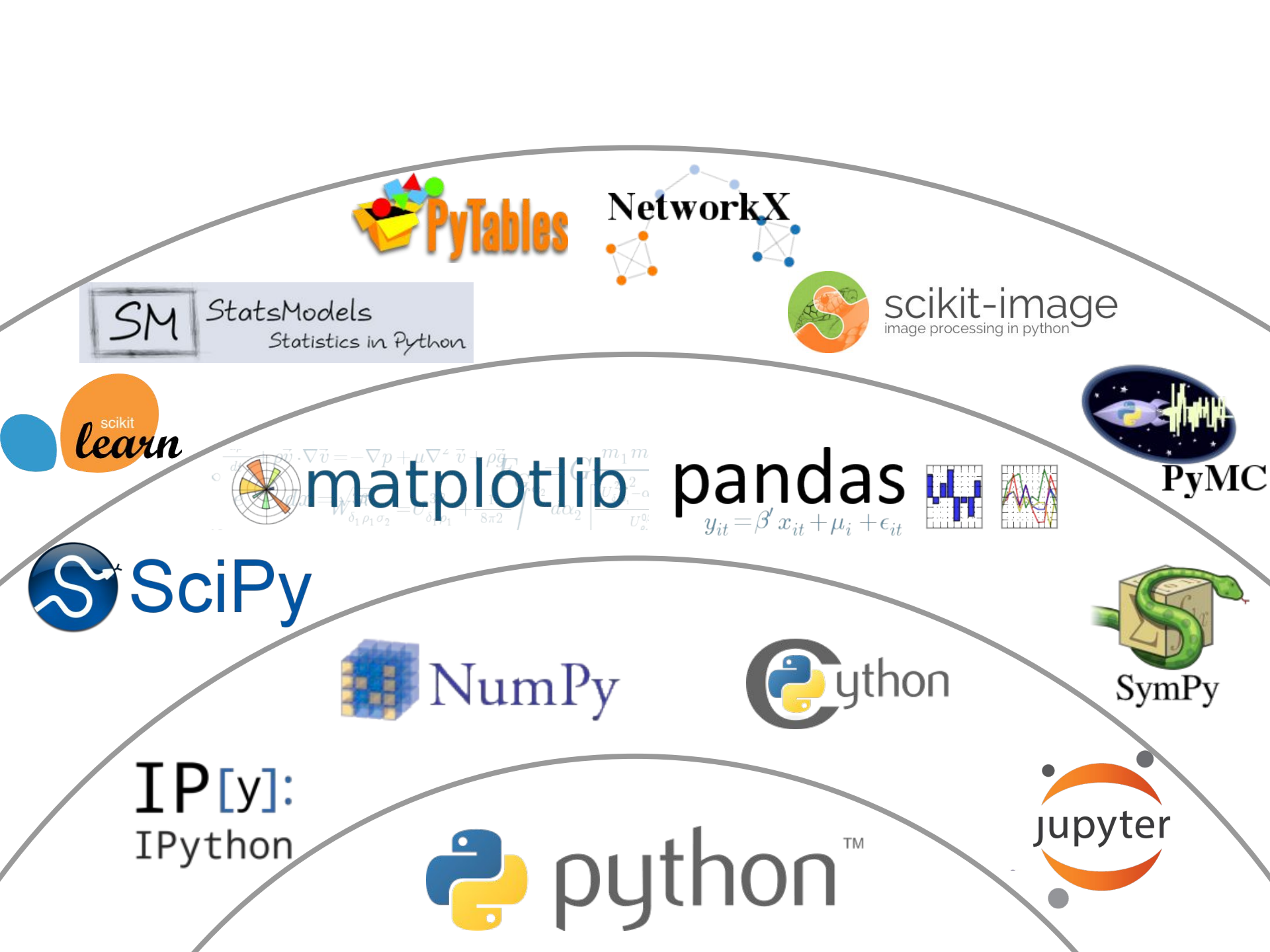
	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
In [2]: data.groupby('Species').mean()
```

Out[2]:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Species				
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026


matplotlib
pandas
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter


PyTables
NetworkX
SM
StatsModels
Statistics in Python
scikit-image
image processing in python
scikit
learn
matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
PyMC
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

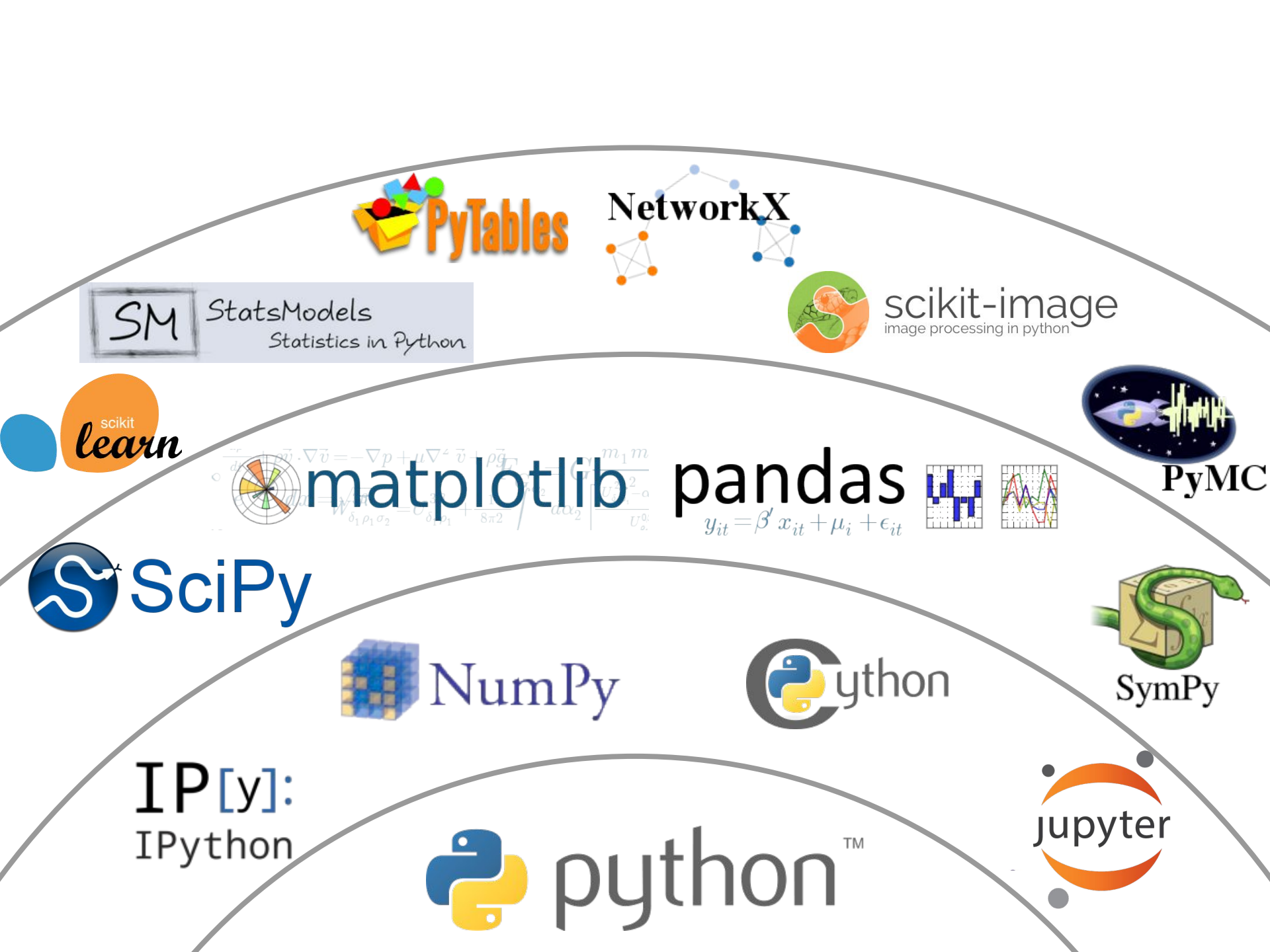
Scikit-Learn

Machine Learning in Python, built on NumPy and SciPy

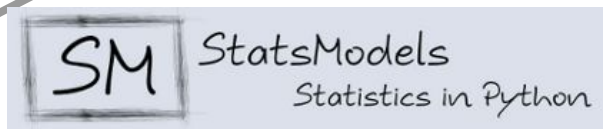
Scikit-Learn

Machine Learning in Python, built on NumPy and SciPy

```
In [3]: from sklearn.ensemble import RandomForestClassifier

        features = data.drop('Species', axis=1)
        labels = data['Species']

        model = RandomForestClassifier()
        model.fit(features, labels)

        predicted = model.predict(features.iloc[:5])
        print(predicted)

        ['setosa' 'setosa' 'setosa' 'setosa' 'setosa']
```


PyTables
NetworkX
SM
StatsModels
Statistics in Python
scikit-image
image processing in python
scikit
learn
matplotlib
pandas
$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$
PyMC
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

(and many, many more)



Recent-ish Developments

- **Dask**: Parallelization of Data & Computation
- **Numba**: LLVM compilation of Python code
- **Jupyter Lab**: interactive & extensible polyglot development environment
- **Altair**: Declarative Visualization based on Vega-Lite

Dask: Parallel Computation for Distributed Arrays & DataFrames

With minimal changes to your NumPy & Pandas expressions, parallelize your computations over distributed data!

http://dask.pydata.org/

Dask: Parallel Computation for Distributed Arrays & DataFrames

A straightforward NumPy computation:

In [3]:
```
import numpy as np

# create an array of normally-distributed random numbers
a = np.random.randn(1000)

# multiply this array by a factor
b = a * 4

# find the minimum value
b_min = b.min()
print(b_min)
```

```
-11.4051061336
```

http://dask.pydata.org/

Dask: Parallel Computation for Distributed Arrays & DataFrames

Dask uses the same expressions . . .

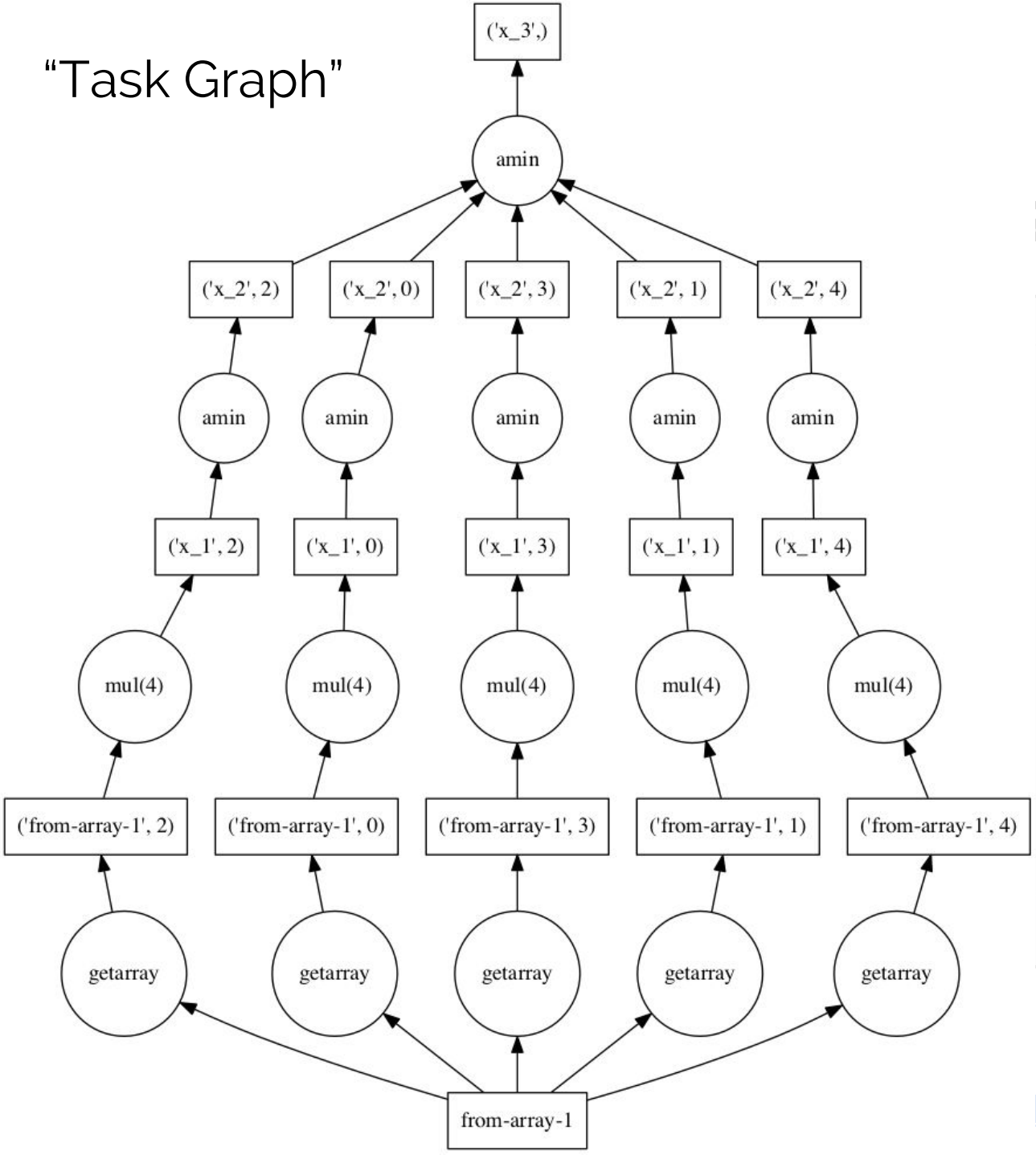
In [4]:
```
import dask.array as da

# create a dask array from the above array
a2 = da.from_array(a, chunks=200)

# multiply this array by a factor
b2 = a2 * 4

# find the minimum value
b2_min = b2.min()
print(b2_min)
```

```
dask.array<x_3, shape=(), chunks=(), dtype=float64>
```

http://dask.pydata.org/

Dask
Distr

"Task Graph"



In [4

k.pydata.org/

Dask: Parallel Computation for Distributed Arrays & DataFrames

```
In [6]: b2_min.compute()

Out[6]: -11.405106133564583
```

http://dask.pydata.org/

Numba: JIT-compilation of Python code

With a simple decorator, Python is compiled to LLVM and executes at near C/Fortran speed!

```
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

%timeit fib(50)
```

```
100000 loops, best of 3: 3.83 µs per loop
```

Still some features missing, but very promising (see my blog posts for some examples).

http://numba.pydata.org/

Numba: JIT-compilation of Python code

With a simple decorator, Python is compiled to LLVM and executes at near C/Fortran speed!

```
@numba.jit
def fib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = b, a + b
    return a

%timeit(fib(50))
```

```
1 loops, best of 3: 468 ns per loop
```

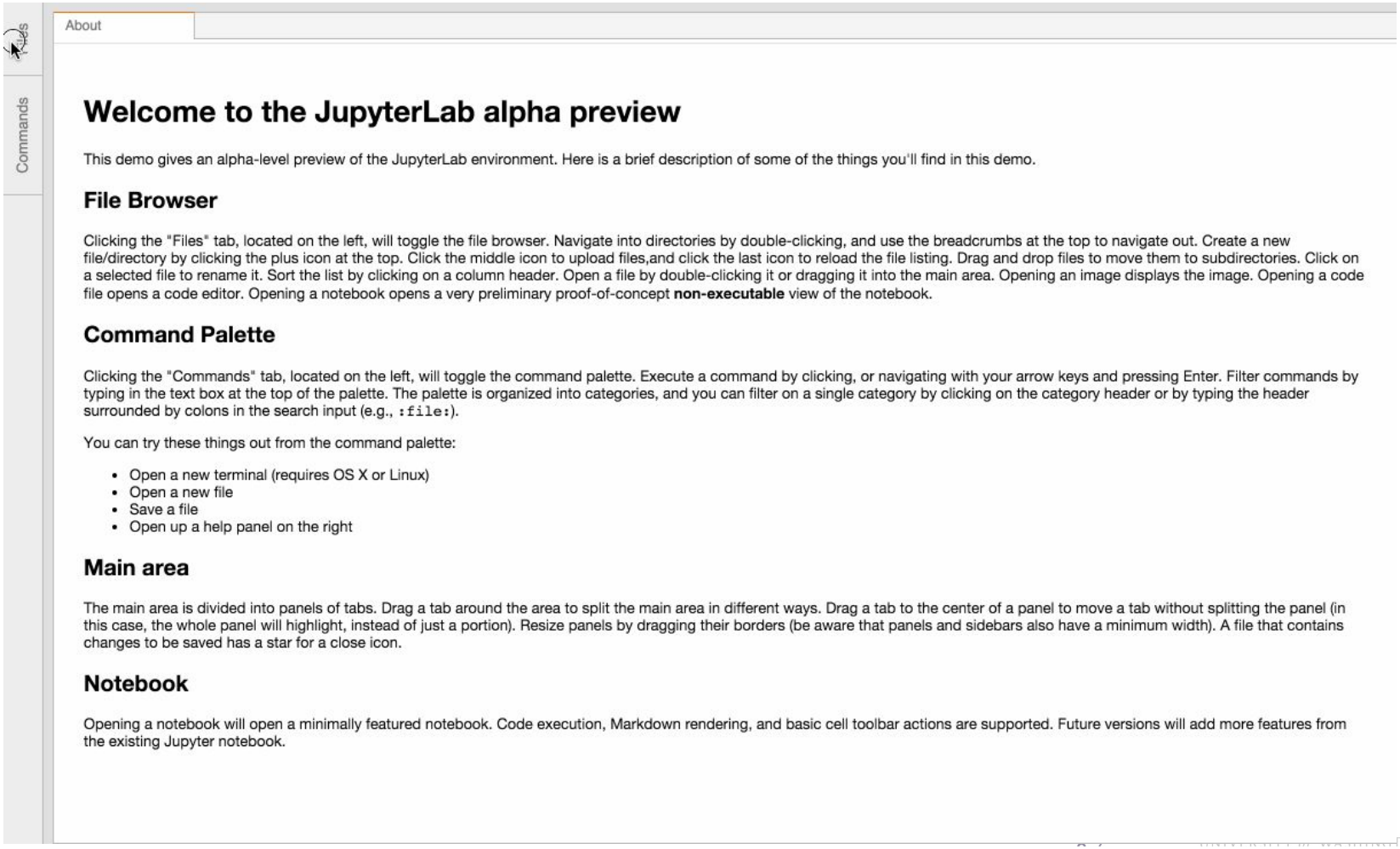
20x speedup!

Still some features missing, but very promising (see my blog posts for some examples).

http://numba.pydata.org/

Jupyter Lab

Jupyter beyond notebooks: extensible cross-platform interactive computing environment (release soon!)



Link to Animation

http://jupyter.org

Thank You!

Email: jakevdp@uw.edu

Twitter: @jakevdp

Github: jakevdp

Web: http://vanderplas.com

Blog: http://jakevdp.github.io