Advances in Scientific Python

- Python 2 vs. 3
- Exploratory Analysis - Jupyter
- Data Visualization - Holoviews
- Data Storage - h5py, memmap
- Speed - Cython, numba
- Parallelization - Dask

Why Python 3?

- Support
- Community
- Features

Unnoticed Features

- Can't compare objects without a natural ordering

In [3]:
```
1 < '2'
0 > None
None < None
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-b8d865814961> in <module>()
----> 1 1 < '2'
      2 0 > None
      3 None < None

TypeError: '<' not supported between instances of 'int' and 'str'
```

- Comprehensions don't leak

In [ ]:

```
my_list = [i for i in range(10)]
print(i)
# Python 2: 9
# Python 3: not defined

number = 20
# More code ...
my_list = [number for number in range(5)]
print(number)
# Python 2: 4
# Python 3: 20
```

- Longer `ints`
- Always imports accelerated code - e.g. `pickle` vs. `cPickle`
- `dicts` use 25-30% less memory
- `dicts` are ordered by default
- Faster I/O - rewritten in C, 2-20x
- **Fast** integer lookup in `range` - 60,000x
- Faster `sorted()`, less memory

- The following modules have substantial improvements in Python 3: `abc`, `aifc`, `argparse`, `array`, `audioop`, `base64`, `binascii`, `bz2`, `cgi`, `cmath`, `code`, `codecs`, `collections`, `colorsys`, `compileall`, `contextlib`, `crypt`, `curses`, `datetime`, `dbm`, `difflib`, `dis`, `distutils`, `doctest`, `email`, `faulthandler`, `filecmp`, `ftplib`, `functools`, `gc`, `glob`, `hashlib`, `hmac`, `html`, `http`, `idlelib` and `IDLE`, `imaplib`, `imghdr`, `importlib`, `inspect`, `io`, `ipaddress`, `itertools`, `json`, `logging`, `math`, `mmap`, `multiprocessing`, `nntplib`, `operator`, `os`, `os.path`, `pdb`, `pickle`, `plistlib`, `poplib`, `pprint`, `pty`, `pydoc`, `re`, `resource`, `sched`, `select`, `shelve`, `shlex`, `shutil`, `signal`, `smtpd`, `smtplib`, `sndhdr`, `socket`, `socketserver`, `sqlite3`, `ssl`, `stat`, `struct`, `subprocess`, `sunau`, `sys`, `sysconfig`, `tarfile`, `tempfile`, `textwrap`, `threading`, `time`, `tkinter`, `traceback`, `types`, `unicodedata`, `unittest`, `urllib`, `wave`, `weakref`, `webbrowser`, `wsgiref`, `xml.etree`, `xml.sax`, `xmlrpc`, `zipfile`, `zlib`

New Features

Unpacking

In [4]:

```
a, *b, c = [1, 2, 3, 4, 5]
print("a:",a ,"\nb:", b, "\nc:", c)
```

```
a: 1
b: [2, 3, 4]
c: 5
```

yield from

In [5]:
```
def duplicate(n):
    for i in range(n):
        yield from [i, i]

print(list(duplicate(5)))
```

```
[0, 0, 1, 1, 2, 2, 3, 3, 4, 4]
```

A Curious Course on Coroutines and Concurrency (http://www.dabeaz.com/coroutines/)

f-strings

In [6]:

```
a = 5
b = 10
print(f'{a} + {b} is {a + b} and not {2 * (a + b)}.')
c = 10.123754
print(f'{c:.2f}')
```

```
5 + 10 is 15 and not 30.
10.12
```

pathlib

In [7]:

```
import os
in_file = os.path.join(os.path.join(os.getcwd(), "in"), "input.txt")
out_file = os.path.join(os.path.join(os.getcwd(), "out"), "output.txt")
```

In [8]:

```
from pathlib import Path
in_file_1 = Path.cwd() / "in" / "input.txt"
out_file_1 = Path.cwd() / "out" / "output.txt"
```

Native type hinting

- More readable code
- Easier to debug
- Type checking by IDEs (like PyCharm)

In [24]:

```
def add(a: int, b: float) -> float:
    return a+b
```

`asyncio` library

- asynchronous I/O with coroutines
- For web-dev and database queries

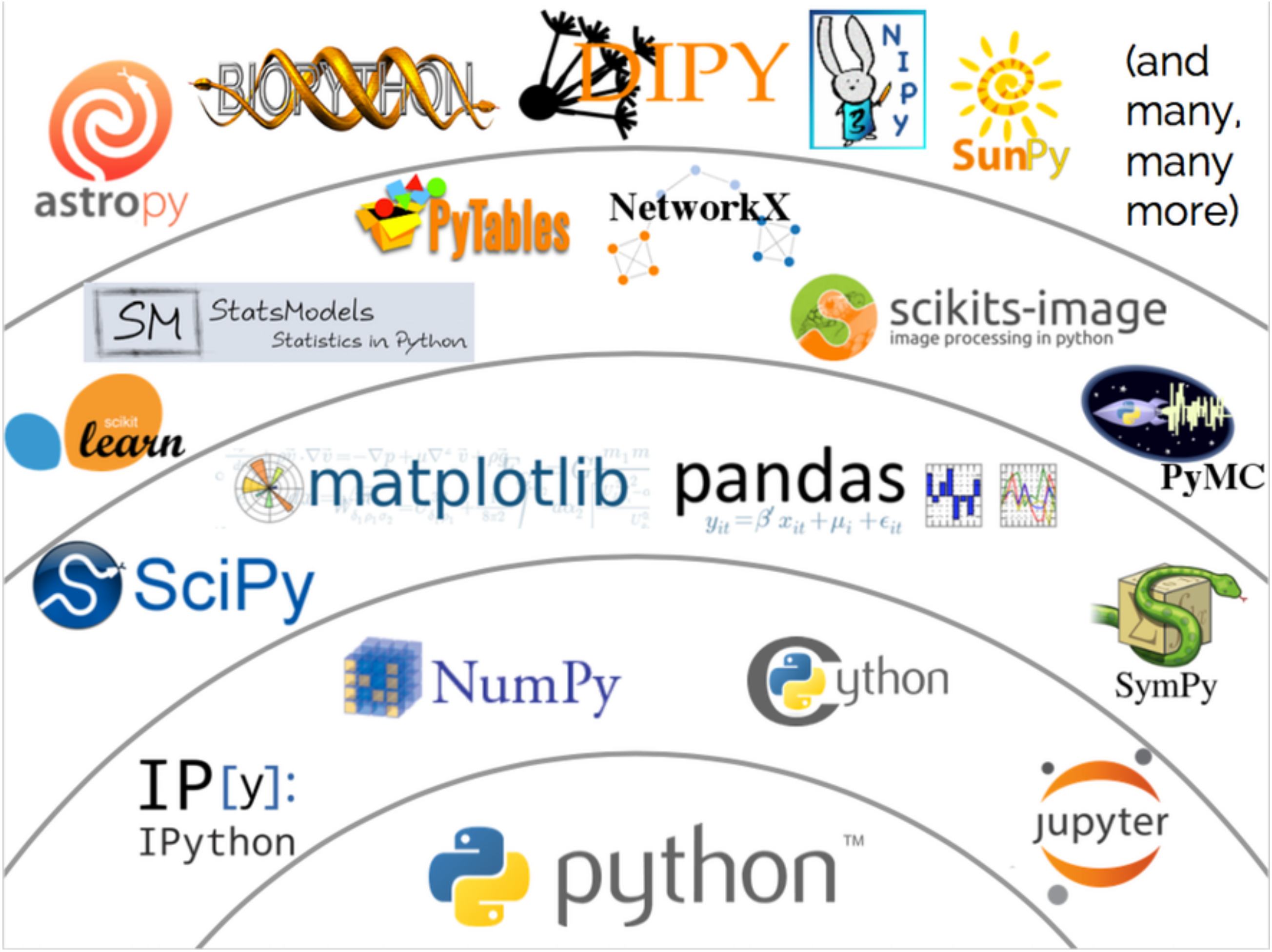
dataclass

In [10]:
```
class StarWarsMovie:
    def __init__(self,
                 title: str,
                 episode_id: int,
                 opening_crawl: str,
                 director: str,
                 producer: str,
                 release_date: str,
                 characters: List[str],
                 planets: List[str],
                 starships: List[str],
                 vehicles: List[str],
                 species: List[str],
                 created: str,
                 edited: str,
                 url: str
                 ):
        self.title = title
        self.episode_id = episode_id
        self.opening_crawl= opening_crawl
        self.director = director
        self.producer = producer
        self.release_date = release_date
        self.characters = characters
        self.planets = planets
        self.starships = starships
        self.vehicles = vehicles
        self.species = species
        self.created = created
        self.edited = edited
        self.url = url
```

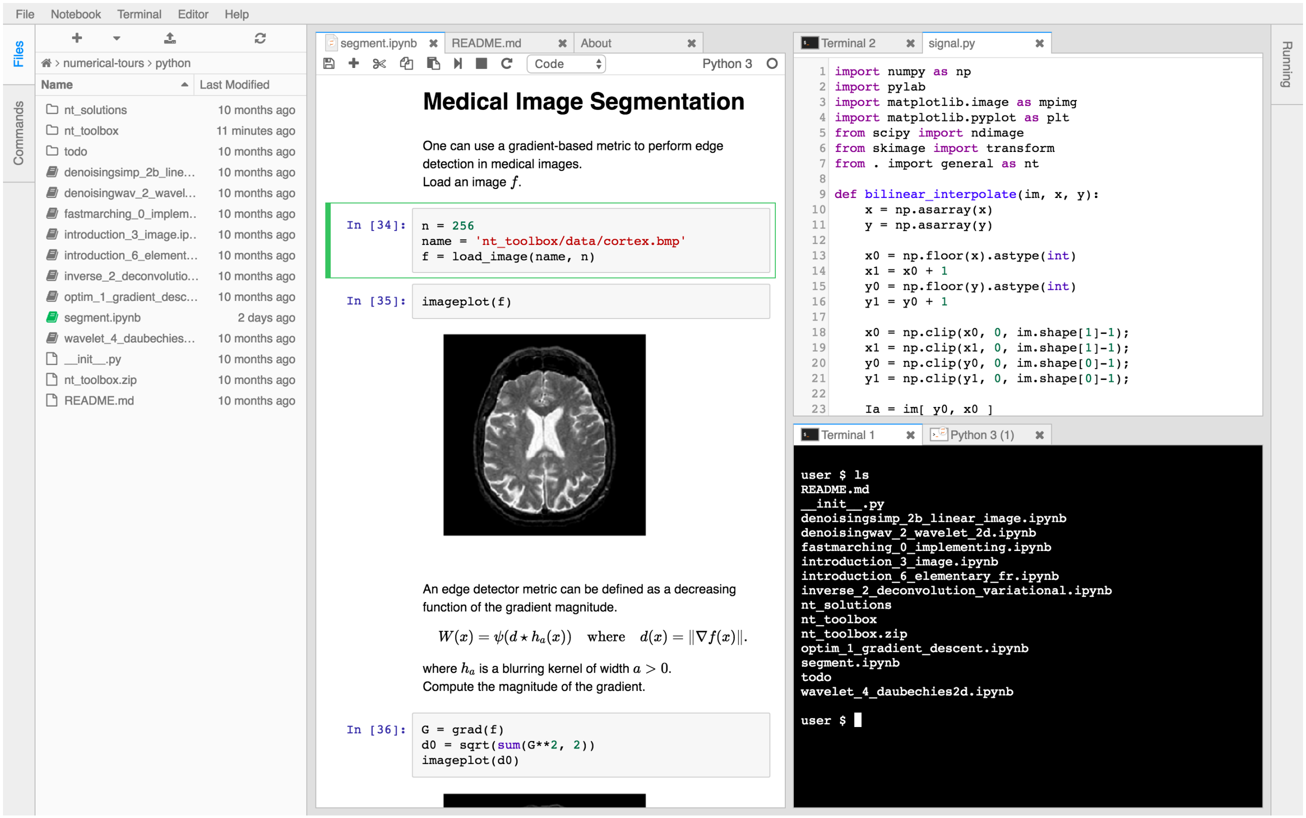
In [ ]:

```
@dataclass
class StarWarsMovie:
    title: str
    episode_id: int = 1
    opening_crawl: str
    director: str
    producer: str
    release_date: datetime
    characters: List[str]
    planets: List[str]
    starships: List[str]
    vehicles: List[str]
    species: List[str]
    created: datetime
    edited: datetime
    url: str
```

- `__post_init__` instead of `__init__`


astropy
BIOPYTHON
DIPY
NIPY
SunPy
(and many, many more)
PyTables
NetworkX
StatsModels
Statistics in Python
scikits-image
image processing in python
scikit learn
matplotlib
pandas
PyMC
SciPy
NumPy
Cython
SymPy
IP[y]:
IPython
python™
jupyter

JupyterLab



File Notebook Terminal Editor Help
Files
Commands
Running
numerical-tours › python
Name
Last Modified
nt_solutions 10 months ago
nt_toolbox 11 minutes ago
todo 10 months ago
denoisingsimp_2b_line… 10 months ago
denoisingwav_2_wavel… 10 months ago
fastmarching_0_implem.. 10 months ago
introduction_3_image.ip.. 10 months ago
introduction_6_element… 10 months ago
inverse_2_deconvolutio… 10 months ago
optim_1_gradient_desc… 10 months ago
segment.ipynb 2 days ago
wavelet_4_daubechies… 10 months ago
__init__.py 10 months ago
nt_toolbox.zip 10 months ago
README.md 10 months ago
segment.ipynb
README.md
About
Code
Python 3
Medical Image Segmentation
One can use a gradient-based metric to perform edge detection in medical images.
Load an image f.
In [34]: n = 256
name = 'nt_toolbox/data/cortex.bmp'
f = load_image(name, n)
In [35]: imageplot(f)
An edge detector metric can be defined as a decreasing function of the gradient magnitude.
W(x) = ψ(d ⋆ h_a(x)) where d(x) = ||∇f(x)||.
where h_a is a blurring kernel of width a > 0.
Compute the magnitude of the gradient.
In [36]: G = grad(f)
d0 = sqrt(sum(G**2, 2))
imageplot(d0)
Terminal 2
signal.py
import numpy as np
import pylab
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
from scipy import ndimage
from skimage import transform
from . import general as nt
def bilinear_interpolate(im, x, y):
x = np.asarray(x)
y = np.asarray(y)
x0 = np.floor(x).astype(int)
x1 = x0 + 1
y0 = np.floor(y).astype(int)
y1 = y0 + 1
x0 = np.clip(x0, 0, im.shape[1]-1);
x1 = np.clip(x1, 0, im.shape[1]-1);
y0 = np.clip(y0, 0, im.shape[0]-1);
y1 = np.clip(y1, 0, im.shape[0]-1);
Ia = im[y0, x0]
Terminal 1
Python 3 (1)
user $ ls
README.md
__init__.py
denoisingsimp_2b_linear_image.ipynb
denoisingwav_2_wavelet_2d.ipynb
fastmarching_0_implementing.ipynb
introduction_3_image.ipynb
introduction_6_elementary_fr.ipynb
inverse_2_deconvolution_variational.ipynb
nt_solutions
nt_toolbox
nt_toolbox.zip
optim_1_gradient_descent.ipynb
segment.ipynb
todo
wavelet_4_daubechies2d.ipynb
user $

Extensions

- variable_inspector (http://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/varInspector/README.html)
- execution_dependencies (http://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions/execution_dependencies/README.html) - Annotate cells with dependencies - run dependencies before running cell.
- RISE (https://github.com/damianavila/RISE)

Teaching

- exercise, nbgrader, nbtutor
- JupyterHub: Manage >1000 users on a single notebook server

Collaboration

With Google Drive - jupyterlab-google-drive (https://github.com/jupyterlab/jupyterlab-google-drive), jupyter-drive (https://github.com/jupyter/jupyter-drive)

Reproducible academic publications (https://github.com/jupyter/jupyter/wiki/a-gallery-of-interesting-jupyter-notebooks#reproducible-academic-publications)

HoloViews

- Interactive visualization
- `bokeh` - renderer
- `datashader` - rasterization of **large** amounts of data
- Uses `numpy` arrays or `pandas` dataframes

Reference Gallery (http://holoviews.org/reference/index.html)

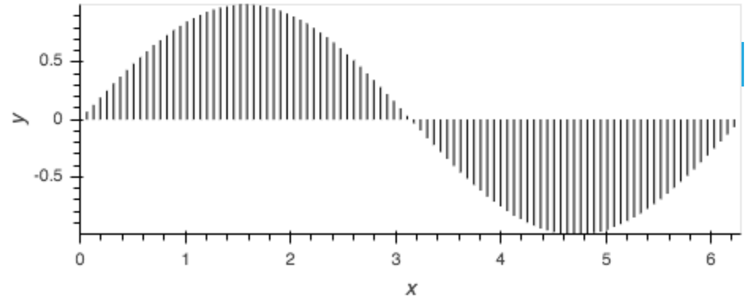
In [6]:

```
import holoviews as hv
hv.extension("bokeh")
```

Consistent API

In [59]:
```
xs = np.linspace(0,2*np.pi,100)
ys = np.sin(xs)

sin_plot = hv.Curve((xs,ys))
hv.Spikes(sin_plot)
```

Out[59]:



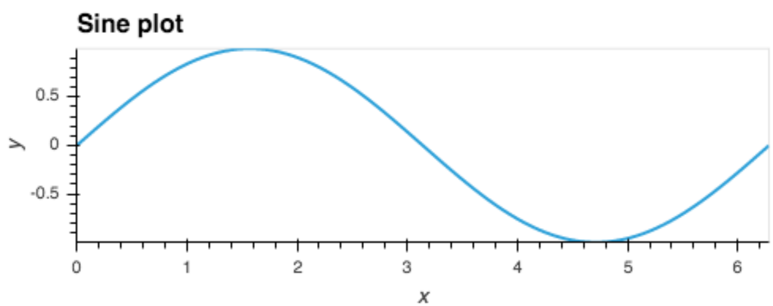
(https://bokeh.pyda

Composable Elements

In [69]:

```
cos_plot = sin_plot.clone((sin_plot.data.x, np.cos(sin_plot.data.x)))
sin_plot = sin_plot.relabel("Sine plot")
cos_plot = cos_plot.relabel("Cosine plot")

layout = sin_plot + hv.Table(sin_plot)
layout
```

Out[69]:



#	x	y

Live scrolling

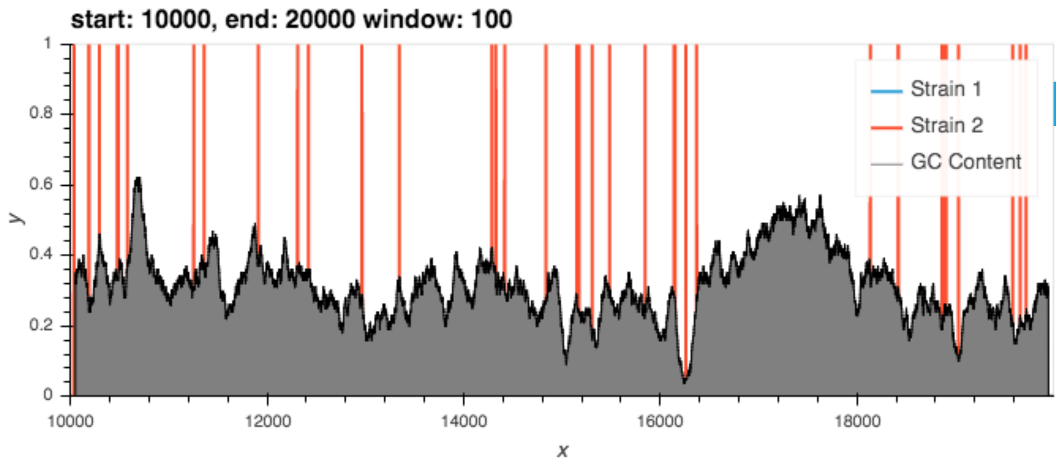
In [56]:

```
def find_gc_content(start=0, end=5000, window=100):
    G = ord("G");C = ord("C")
    seq = vcf.fasta_array[0][start:end].view("int8")
    res = find_gc_content_core(seq, start, end, window, G, C)
    return hv.Area((np.arange(start,end), res))

def extract_SNV(snv_id=0, start=0,end=5000):
    seq = vcf.vcf_array[snv_id][start:end].view("int8")
    return hv.Curve((np.arange(start,end), np.log1p(seq)//4))

def gc_and_SNV(start=0, end=5000, window=100):
    gc_content = find_gc_content(start, end, window).opts(style=dict(color="grey")
)
    snv_0 = extract_SNV(0, start, end).relabel("Strain 1")
    snv_1 = extract_SNV(1, start, end).relabel("Strain 2")
    layout = snv_0 * snv_1 * gc_content * hv.Curve(gc_content).relabel("GC Content
").opts(style=dict(color="grey",line_width=None))
    return layout
```

In [57]:

```
%opts Curve [width=700 height=300]
%opts Area [width=700 height=300]
dmap = hv.DynamicMap(gc_and_SNV,kdims=["start","end","window"])
dmap = dmap.redim.range(start=(10000,100000), end=(20000,200000), window=(100,500)
)
dmap
```

Out[57]:



(https://boke

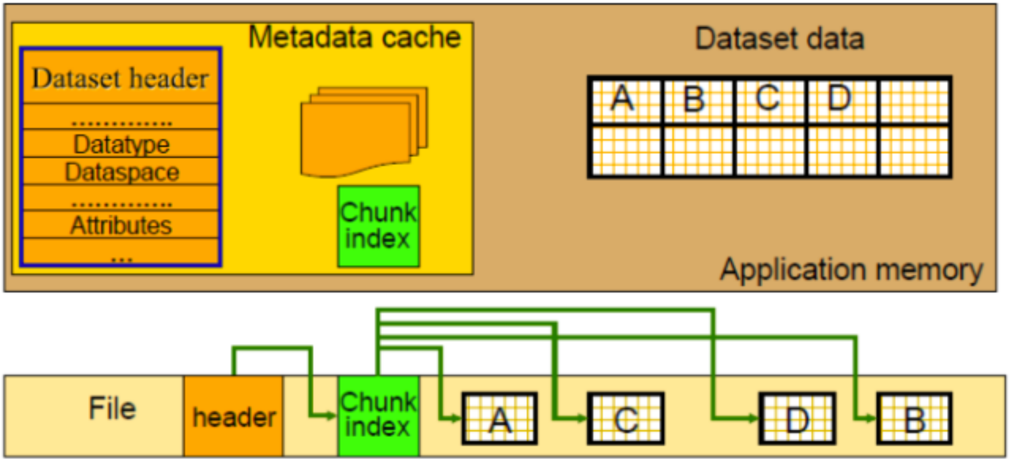
Data storage

h5py (HDF5 for python)

- HDF5 - a binary database format
- Some features are data chunking, data extension, parallel I/O, compression, complex selection, and in-core calculations

h5py - Data chunking

- Several reasons to use:
  - Data used in code may exceed RAM
  - Fast access to large amounts of stored data
  - Parallelizing updates in large numeric matrices/arrays



Speed

In [27]:

```
def pairwise_python(X):
    M = X.shape[0]
    N = X.shape[1]
    D = np.empty((M, M), dtype=np.float)
    for i in range(M):
        for j in range(M):
            d = 0.0
            for k in range(N):
                tmp = X[i, k] - X[j, k]
                d += tmp * tmp
            D[i, j] = np.sqrt(d)
    return D

X = np.random.random((1000, 3))
%timeit -n5 -r1 pairwise_python(X)
```

```
4.25 s ± 0 ns per loop (mean ± std. dev. of 1 run, 5 loops each)
```

Cython

- C/C++ speed
- Verbose code
- Compiled
- Hard to debug

In [51]:

```
%%cython
import numpy as np
cimport cython
from libc.math cimport sqrt

@cython.boundscheck(False)
@cython.wraparound(False)
def pairwise_cython(double[:, ::1] X):
    cdef int M = X.shape[0]
    cdef int N = X.shape[1]
    cdef double tmp, d
    cdef double[:, ::1] D = np.empty((M, M), dtype=np.float64)
    for i in range(M):
        for j in range(M):
            d = 0.0
            for k in range(N):
                tmp = X[i, k] - X[j, k]
                d += tmp * tmp
            D[i, j] = sqrt(d)
    return np.asarray(D)
```

In [30]:

```
%timeit pairwise_cython(X)
```

```
5.32 ms ± 316 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Numba

- C/Fortran speed
- One extra line of code
- JIT
- Debugging is just in normal Python
- Automatic parallelization

In [31]:
```
import numba
pairwise_numba = numba.jit(pairwise_python)
```

In [34]:
```
%timeit pairwise_numba(X)
```

```
5.35 ms ± 240 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

In [32]:
```
@numba.jit(parallel=True)
def pairwise_numba_parallel(X):
    M = X.shape[0]
    N = X.shape[1]
    D = np.empty((M, M), dtype=np.float)
    for i in numba.prange(M):
        for j in numba.prange(M):
            d = 0.0
            for k in numba.prange(N):
                tmp = X[i, k] - X[j, k]
                d += tmp * tmp
            D[i, j] = np.sqrt(d)
    return D
```

In [36]:
```
%timeit pairwise_numba_parallel(X)
```

```
3.86 ms ± 410 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

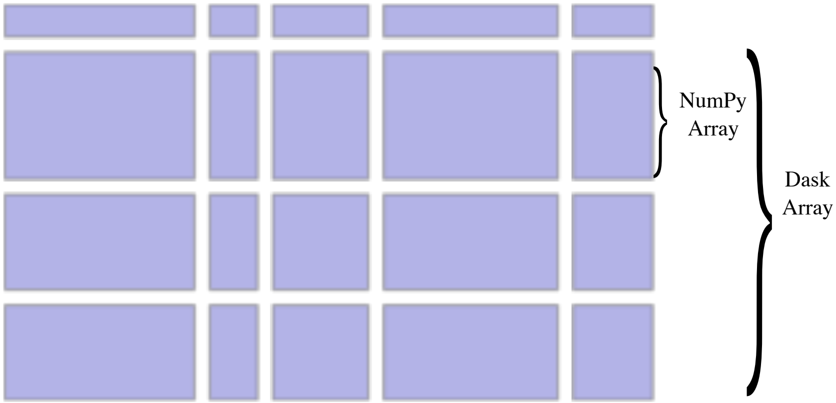
- e.g. sarkas - molecular dynamics library, slightly faster than optimized C code.
- May become industry standard soon

Parallelization

Dask

- Parallelizes NumPy, Pandas and SKLearn
- Arbitrary computations
- Can be used in cluster computing
- Dashboard and task graphs

NumPy



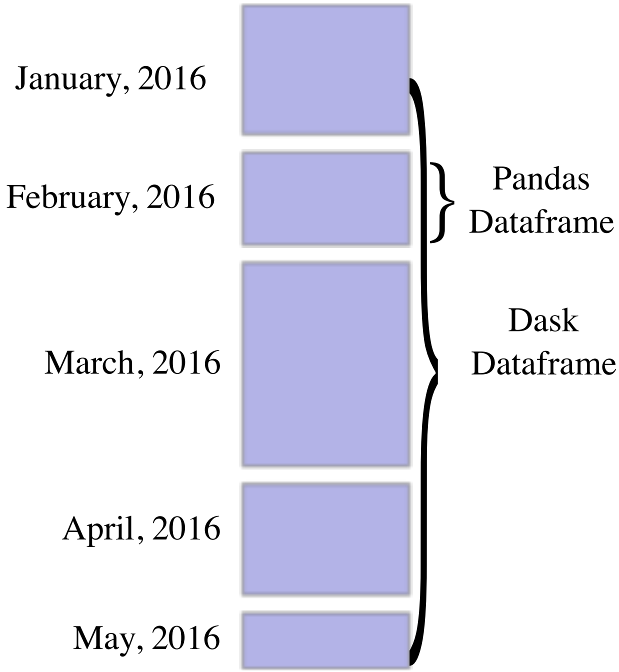
In [39]:
```
import numpy as np
x = np.random.random((10000,10000))
%timeit x + x.T - x.mean(axis=0)
```

```
2.79 s ± 178 ms per loop (mean ± std. dev. of 7 runs, 1 loop each)
```

In [457]:
```
import dask.array as da
x = da.random.random((10000,10000), chunks=(1000, 1000))
%timeit x + x.T - x.mean(axis=0)
```

```
5.79 ms ± 585 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Pandas



January, 2016
February, 2016
March, 2016
April, 2016
May, 2016
Pandas
Dataframe
Dask
Dataframe

In [ ]:

```
import pandas as pd
df = pd.read_csv('myfile.csv', parse_dates=['timestamp'])
df.groupby(df.timestamp.dt.hour).value.mean()

import dask.dataframe as dd
df = dd.read_csv('hdfs://myfiles.*.csv', parse_dates=['timestamp'])
df.groupby(df.timestamp.dt.hour).value.mean()
```

Pure Python code

In [453]:

```
def inc(x):
    return x + 1

def double(x):
    return x + 2

def add(x, y):
    return x + y

data = [1, 2, 3, 4, 5]

output = []
for x in data:
    a = inc(x)
    b = double(x)
    c = add(a, b)
    output.append(c)

total = sum(output)
total
```

Out[453]:

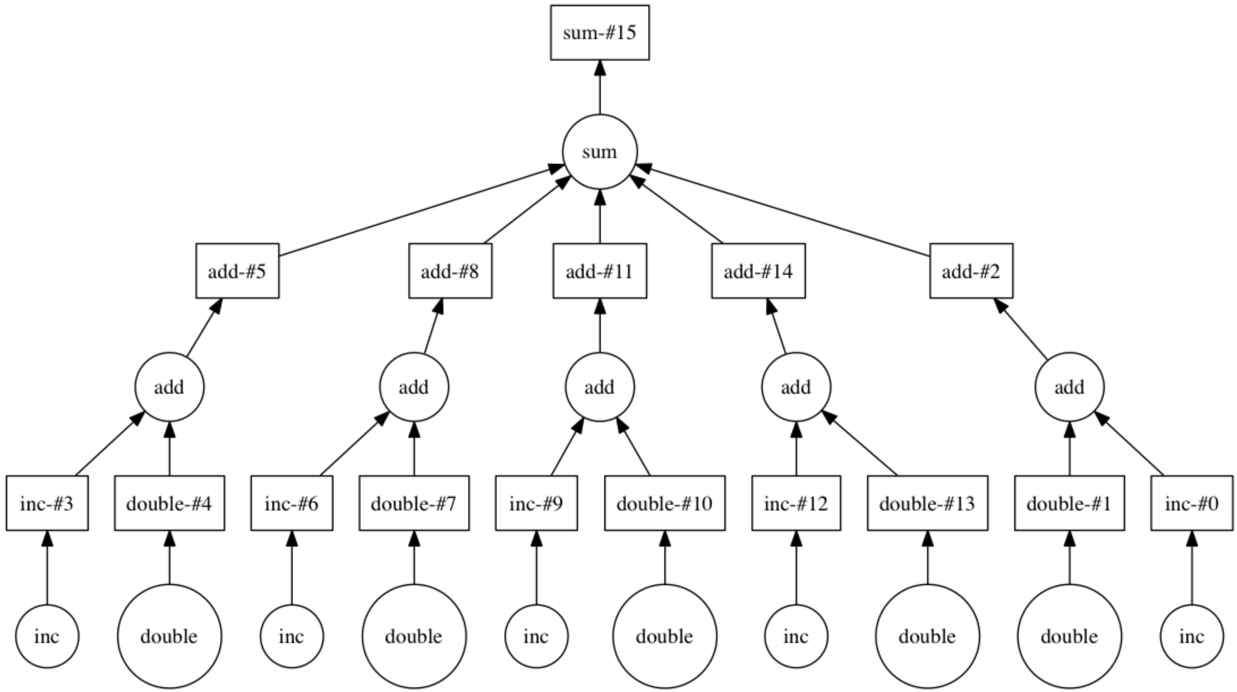
```
45
```

In [451]:
```
import dask

@dask.delayed
def inc(x):
    return x + 1

@dask.delayed
def double(x):
    return x + 2

@dask.delayed
def add(x, y):
    return x + y

data = [1, 2, 3, 4, 5]

output = []
for x in data:
    a = inc(x)
    b = double(x)
    c = add(a, b)
    output.append(c)

total = dask.delayed(sum)(output)
total.compute()
```

Out[451]:
```
45
```

In [452]:
```
total.visualize()
```

Out[452]:



Distributed Computing

In [65]:
```python
from dask.distributed import Client
client = Client()
client
```

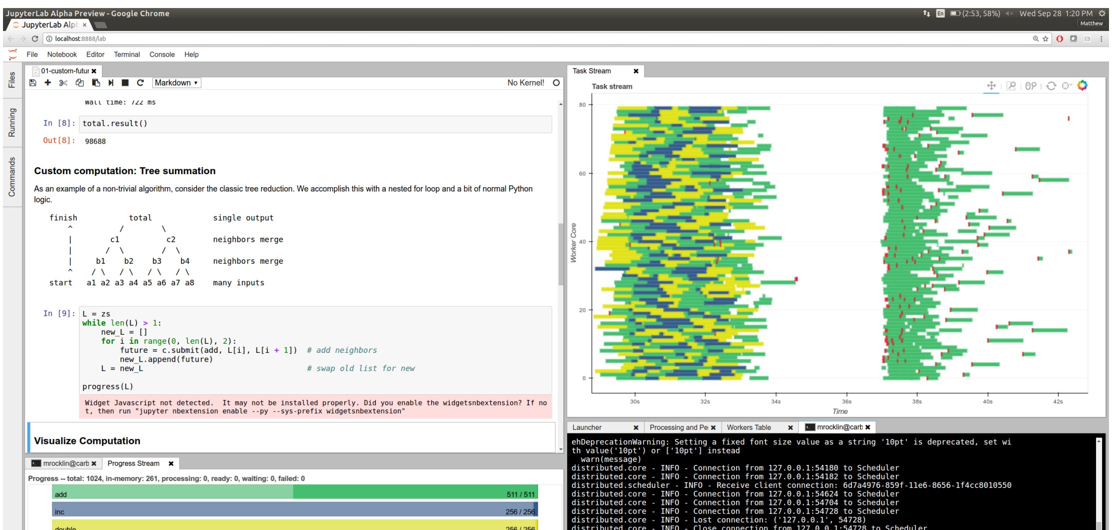
Out[65]:

Client

- **Scheduler:** tcp://127.0.0.1:50923
- **Dashboard:** [http://127.0.0.1:50924/status (http://127.0.0.1:50924/status)](http://127.0.0.1:50924/status)

Cluster

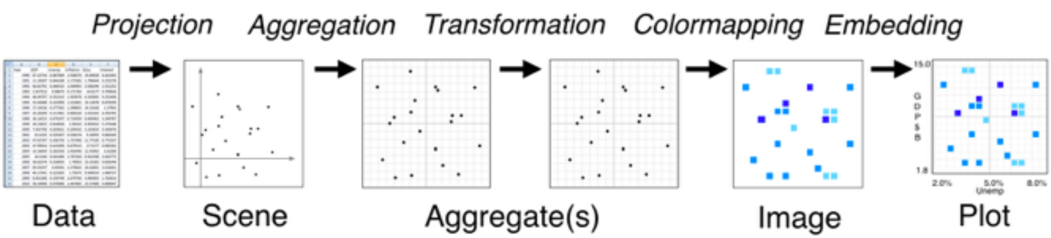
- **Workers:** 4
- **Cores:** 4
- **Memory:** 8.59 GB



Datashader

Numba DASK Datashader

How does datashader work?



SciPy 2018 (https://scipy2018.scipy.org/ehome/index.php?eventid=299527&)

EuroSciPy 2018 (https://www.euroscipy.org/2018/)

Questions?