Easily reduce runtimes with Cython

Michal Mucha

@jeremimucha

Lightning talk
PyData London Meetup
10/3/2017

Did you know..

how easy you can get

up to 3 orders of magnitude

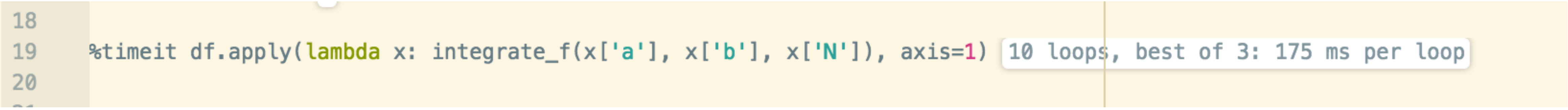
of runtime reduction?

Applications

- speed up that API request
- make your algorithm run faster
- less waiting for your simulation results
- $ave dat money on your cloud bill

example #1

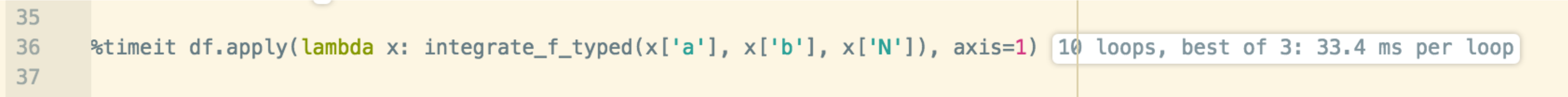
apply a function to observations



```
%timeit df.apply(lambda x: integrate_f(x['a'], x['b'], x['N']), axis=1)  10 loops, best of 3: 175 ms per loop
```

175ms

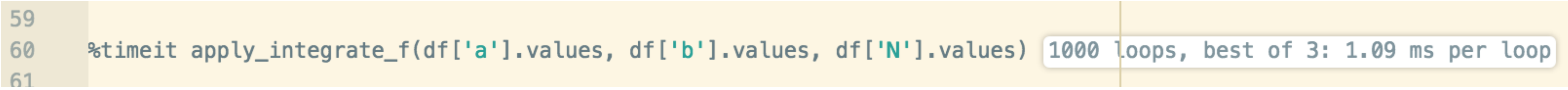
typed cython function



```
%timeit df.apply(lambda x: integrate_f_typed(x['a'], x['b'], x['N']), axis=1)  10 loops, best of 3: 33.4 ms per loop
```

33ms

typed function applied to underlying arrays



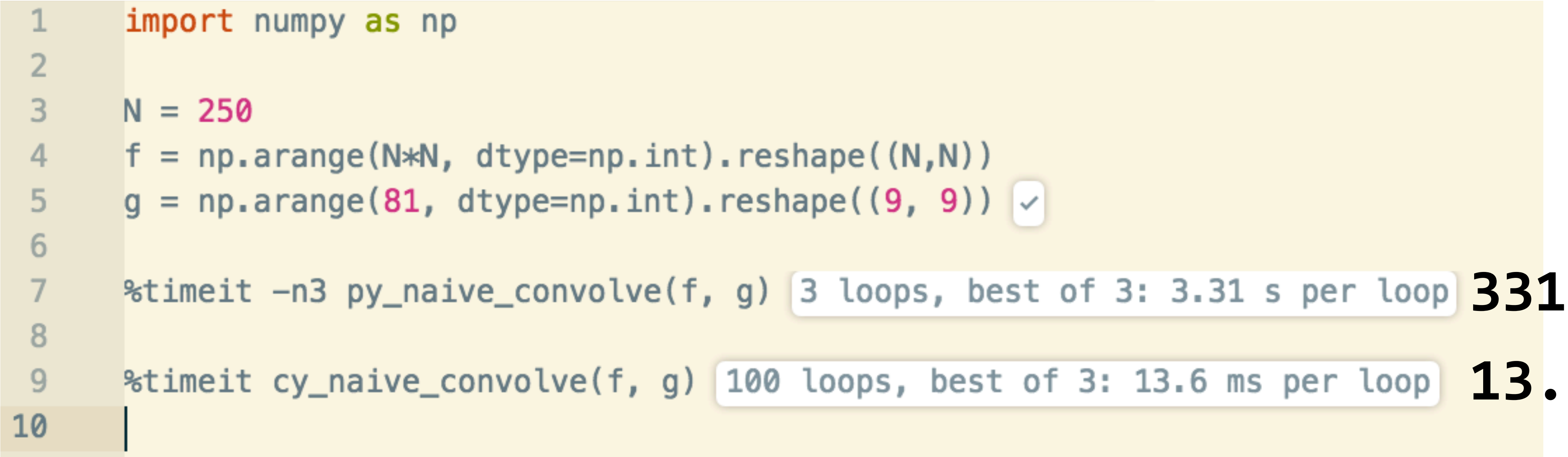
```
%timeit apply_integrate_f(df['a'].values, df['b'].values, df['N'].values)  1000 loops, best of 3: 1.09 ms per loop
```

1ms

https://pandas.pydata.org/pandas-docs/stable/enhancingperf.html

example #2

Convolution



```
import numpy as np

N = 250
f = np.arange(N*N, dtype=np.int).reshape((N,N))
g = np.arange(81, dtype=np.int).reshape((9, 9)) ✓

%timeit -n3 py_naive_convolve(f, g) 3 loops, best of 3: 3.31 s per loop

%timeit cy_naive_convolve(f, g) 100 loops, best of 3: 13.6 ms per loop

```

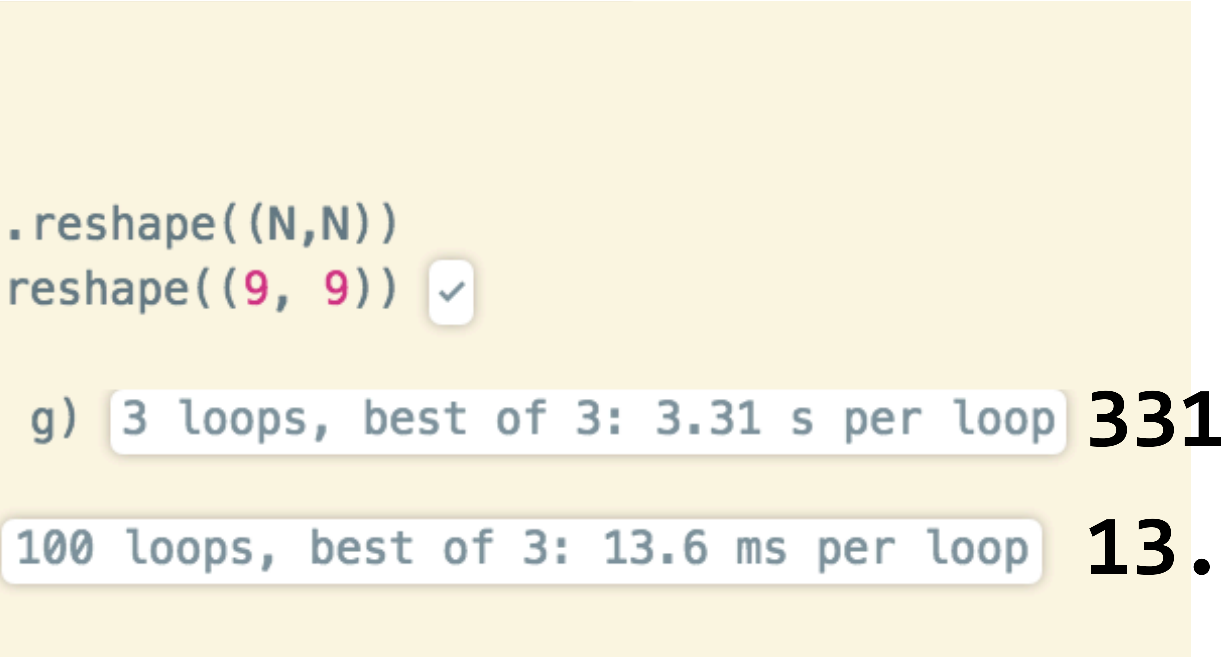
3310ms

13.6ms

http://docs.cython.org/en/latest/src/tutorial/numpy.html

example #2

Convolution



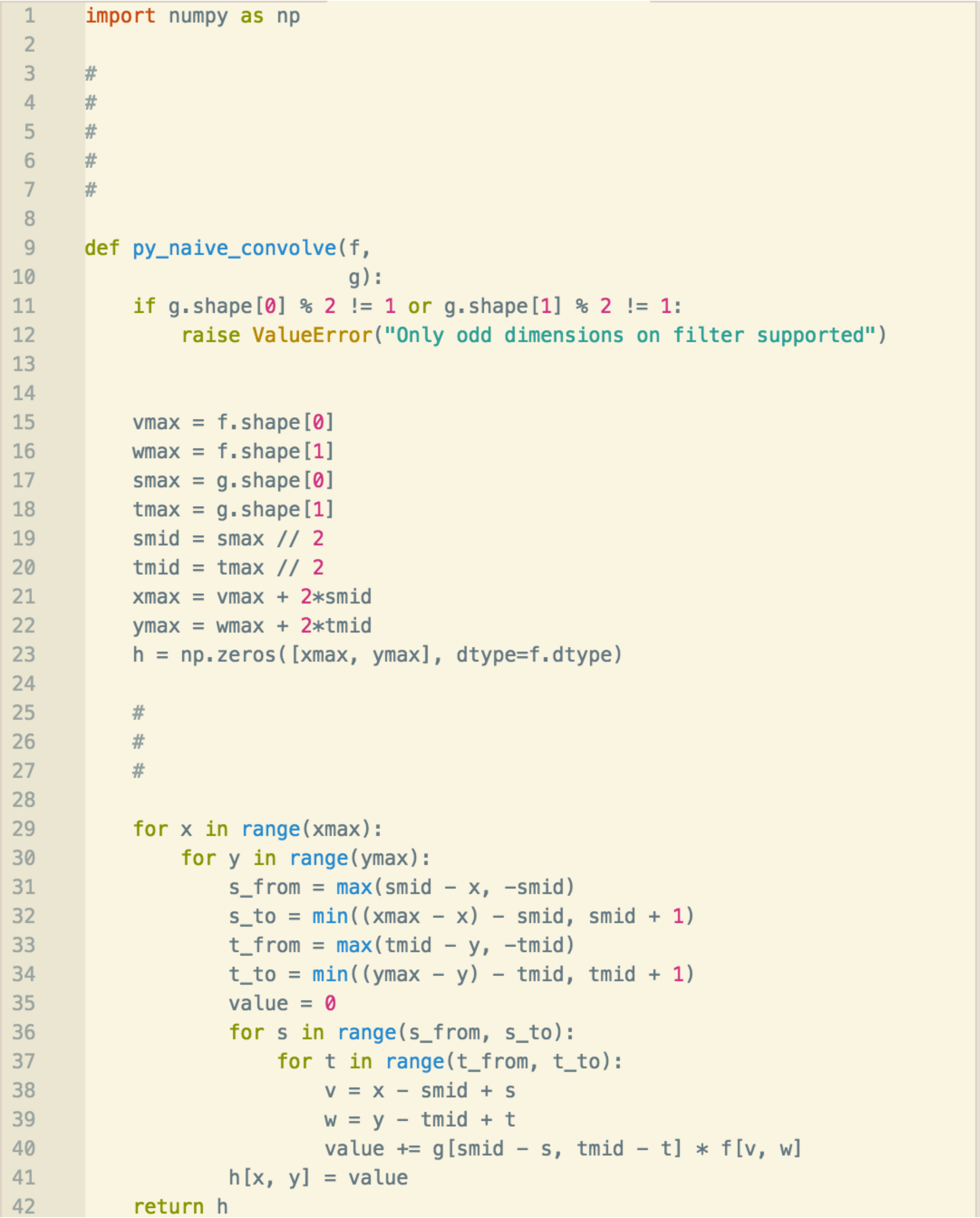
```
1      import numpy as np
```

```
.reshape((N,N))
reshape((9, 9)) ✓

g) 3 loops, best of 3: 3.31 s per loop

100 loops, best of 3: 13.6 ms per loop
```

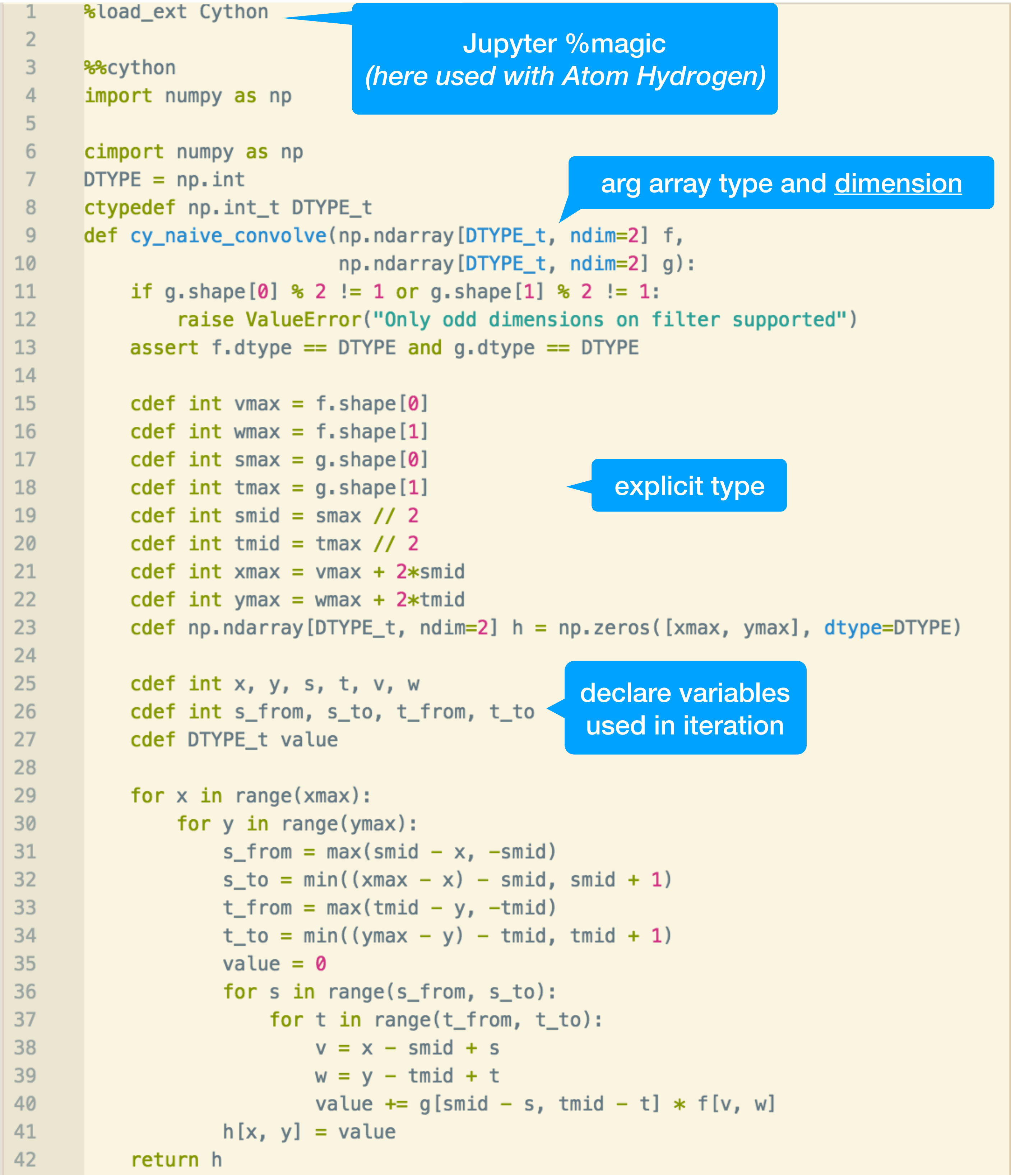
3310ms

13.6ms

/latest/src/tutorial/numpy.html



```
import numpy as np

#
#
#
#
#

def py_naive_convolve(f,
                      g):
    if g.shape[0] % 2 != 1 or g.shape[1] % 2 != 1:
        raise ValueError("Only odd dimensions on filter supported")


    vmax = f.shape[0]
    wmax = f.shape[1]
    smax = g.shape[0]
    tmax = g.shape[1]
    smid = smax // 2
    tmid = tmax // 2
    xmax = vmax + 2*smid
    ymax = wmax + 2*tmid
    h = np.zeros([xmax, ymax], dtype=f.dtype)

    #
    #
    #

    for x in range(xmax):
        for y in range(ymax):
            s_from = max(smid - x, -smid)
            s_to = min((xmax - x) - smid, smid + 1)
            t_from = max(tmid - y, -tmid)
            t_to = min((ymax - y) - tmid, tmid + 1)
            value = 0
            for s in range(s_from, s_to):
                for t in range(t_from, t_to):
                    v = x - smid + s
                    w = y - tmid + t
                    value += g[smid - s, tmid - t] * f[v, w]
            h[x, y] = value
    return h
```



```
%load_ext Cython

%%cython
import numpy as np

cimport numpy as np
DTYPE = np.int
ctypedef np.int_t DTYPE_t
def cy_naive_convolve(np.ndarray[DTYPE_t, ndim=2] f,
                      np.ndarray[DTYPE_t, ndim=2] g):
    if g.shape[0] % 2 != 1 or g.shape[1] % 2 != 1:
        raise ValueError("Only odd dimensions on filter supported")
    assert f.dtype == DTYPE and g.dtype == DTYPE

    cdef int vmax = f.shape[0]
    cdef int wmax = f.shape[1]
    cdef int smax = g.shape[0]
    cdef int tmax = g.shape[1]
    cdef int smid = smax // 2
    cdef int tmid = tmax // 2
    cdef int xmax = vmax + 2*smid
    cdef int ymax = wmax + 2*tmid
    cdef np.ndarray[DTYPE_t, ndim=2] h = np.zeros([xmax, ymax], dtype=DTYPE)

    cdef int x, y, s, t, v, w
    cdef int s_from, s_to, t_from, t_to
    cdef DTYPE_t value

    for x in range(xmax):
        for y in range(ymax):
            s_from = max(smid - x, -smid)
            s_to = min((xmax - x) - smid, smid + 1)
            t_from = max(tmid - y, -tmid)
            t_to = min((ymax - y) - tmid, tmid + 1)
            value = 0
            for s in range(s_from, s_to):
                for t in range(t_from, t_to):
                    v = x - smid + s
                    w = y - tmid + t
                    value += g[smid - s, tmid - t] * f[v, w]
            h[x, y] = value
    return h
```

Jupyter %magic
(here used with Atom Hydrogen)

arg array type and dimension

explicit type

declare variables used in iteration

http://docs.cython.org/en/latest/src/tutorial/numpy.html

what to remember:

- **type** your variables for easier porting
- make sure to **match types** - Python and Cython
- Jupyter Notebook **%%cython magic**
- keep the opportunity in mind
  when designing your module :)

typing in python3.6

```python
from typing import List, Tuple

Product = Tuple[str, float]

product_list: List[Product] = [
    ("Juice", 1.49),
    ("Bread", 2.70)
]
```

adding types to function declarations and variable assignments:

- **helps avoid bugs**
- **makes it easy to convert to Cython**

When developing Cython code…

bind type once and for good

```
import numpy as np
cimport numpy as np
DTYPE = np.int
ctypedef np.int_t DTYPE_t

cdef DTYPE_t my_f(DTYPE_t a, DTYPE_t b):
    return a + b
```

use numpy to match types and avoid a headache

matching type examples:

Python	int **or** np.int_
C	long **or** np.int_
Python	float **or** np.float64
C	double **or** np.float64
Python	np.float32
C	float **or** np.float32

three-step program

1. use types
2. declare variables
3. compile

resources

- **pandas performance enhancement** https://pandas.pydata.org/pandas-docs/stable/enhancingperf.html
- **Cython docs- working with numpy** http://docs.cython.org/en/latest/src/tutorial/numpy.html
- **numpy vectorize examples** https://www.programcreek.com/python/example/52272/numpy.vectorize
- **numba (check @jit and @vectorize)** http://numba.pydata.org/numba-doc/dev/user/jit.html
- **Compiling Cython** http://cython.readthedocs.io/en/latest/src/userguide/source_files_and_compilation.html
- **@iamtrask ’s blog post on mmult** https://iamtrask.github.io/2014/11/23/cython-blas-fortran/

Easily reduce runtimes with Cython

Michal Mucha

🐦 @jeremimucha