cs584 (s16): Using Python with Cython

Step 1: Define a cython function in a .pyx file

Place the following inside myprime.pyx (note the use of cdef)

```
def primes(int kmax):
    cdef int n, k, i
    cdef int p[1000]
    result = []
    if kmax > 1000:
        kmax = 1000
    k = 0
    n = 2
    while k < kmax:
        i = 0
        while i < k and n % p[i] != 0:
            i = i + 1
        if i == k:
            p[k] = n
            k = k + 1
            result.append(n)
        n = n + 1
    return result
```

Step 2: Define a setup.py file

```
from distutils.core import setup
from Cython.Build import cythonize

setup(
  name = ’My prime module’,
  ext_modules = cythonize("myprime.pyx"),
)
```

Step 3: Compile the cython module

Execute from within a terminal:

```
python setup.py build_ext --inplace
```

Step4: Load the compiled module and test

```
import myprime
myprime.primes(1000000000);
```

Timing test

```
# Make sure the cython function is compiled (opotional)
from subprocess import call
call(["python setup.py build_ext --inplace"], shell = True)

import numpy as np
import myprime
import time

def primes(kmax):
#    def  n, k, i
    p = np.zeros(1000)
    result = []
    if kmax > 1000:
        kmax = 1000
    k = 0
    n = 2
    while k < kmax:
        i = 0
        while i < k and n % p[i] != 0:
            i = i + 1
        if i == k:
            p[k] = n
            k = k + 1
            result.append(n)
        n = n + 1
    return result


start_time = time.time();
primes(1000000000);
print("--- Python function: %s seconds ---" % (time.time() - start_time));
```

```
start_time = time.time();
myprime.primes(1000000000);
print("--- Cython function: %s seconds ---" % (time.time() - start_time));
```

Output

```
 --- Python function: 0.185851097107 seconds ---
--- Cython function: 0.00182819366455 seconds ---
```