# Code Optimization and Cython/C
# Physics 91SI

May 19, 2015

# How to write fast code

# How to write fast code

1. Think before starting – come up with a good algorithm
    1.1 Ex: Need to sort a list

# How to write fast code

1. Think before starting – come up with a good algorithm
    1.1 Ex: Need to sort a list
2. Implement using good programming style
3. ???
4. Profit

# How to make existing code faster

# How to make existing code faster

1. **Profile** your code and identify bottlenecks
    1.1 A **profiler** is a tool that runs your program while measuring the time taken to execute every part of the program
    1.2 Can help you find the computational "bottlenecks" in your code

# How to make existing code faster

1. **Profile** your code and identify bottlenecks
   1.1 A **profiler** is a tool that runs your program while measuring the time taken to execute every part of the program
   1.2 Can help you find the computational “bottlenecks” in your code
2. Examine bottlenecks and figure out how to make those parts of code less computationally expensive

# How to make existing code faster

1. **Profile** your code and identify bottlenecks
    1.1 A **profiler** is a tool that runs your program while measuring the time taken to execute every part of the program
    1.2 Can help you find the computational “bottlenecks” in your code
2. Examine bottlenecks and figure out how to make those parts of code less computationally expensive

# Profiling in Python

- Python has a profiler cProfile
- Import the cProfile pacakge and then give it the argument of the function to run (as a string)

```
import cProfile

cProfile.run('string indicating thing to run')
```

Lets try writing and then profiling some code!

# Example

Task: Find the first $N$ prime number

# Example

Task: Find the first $N$ prime number
Let's implement this in python and profile it!
What are the most costly functions that we should try to optimize?

# The C Programming Language

- Lower level language than python
  - Need to deal with your own memory management, pointers, etc.
  - Less error checking and less user friendly
- Compiled language (python is read by an interpreter)
- Static typed language
- However, being a low-level language makes it **much** faster than languages such as python, thus it is often useful to be able to leverage C's computational efficiency in many large problems.
  - Thus C/C++ used for most large computational science codes.

# Example

Let's look at some code and talk about syntax differences. We will examine the same prime number solver code as before, just in *C* this time.

# Example

Let's look at some code and talk about syntax differences. We will examine the same prime number solver code as before, just in *C* this time.

What differences do we see?

How do we run the code?

Also note that although it is faster, development is harder and more time consuming typically.

# Cython

- Cython is a superset of the python programming language
    - Supports almost all standard python syntaxes, but also allows C/C++ routines and static types

# Cython

- Cython is a superset of the python programming language
  - Supports almost all standard python syntaxes, but also allows C/C++ routines and static types
- Cython actually translates the python code into C and compiles it to later use when executing
  - Will not be as fast as pure C though since still uses the python interpreter and libaries to allow for this to run

# Cython

- Cython is a superset of the python programming language
  - Supports almost all standard python syntaxes, but also allows C/C++ routines and static types
- Cython actually translates the python code into C and compiles it to later use when executing
  - Will not be as fast as pure C though since still uses the python interpreter and libaries to allow for this to run
- Still very useful as can use the C routines/types for the computationally intense parts of our code, while keeping the rest of the code in python still so easy for implementation

# Cython Example

Let's write our prime number finder in Cython now!

# Cython Example

Let's write our prime number finder in Cython now!
How fast is it in comparision to the previous codes?
How difficult was this to do compared to the previous codes?