

Fast Code with Cython

Robert Bradshaw

University of Washington

Math 480, April 9, 2008



What is Cython?

*Cython is a **optimized**, more **feature-rich** fork of **Pyrex**, motivated primarily by the needs of **Sage**.*

What is Pyrex?

Pyrex lets you write code that mixes Python and C data types any way you want, and compiles it into a C extension for Python.

— Greg Ewing (Author)

What is Cython?

- Pseudo-Python to C **compiler**

What is Cython?

- Pseudo-Python to C **compiler**
- Language extensions for **statically declaring types**
 - Potentially massive speedups
 - Integration with external libraries

What is Cython?

- Pseudo-Python to C **compiler**
- Language extensions for **statically declaring types**
 - Potentially massive speedups
 - Integration with external libraries
- Python **memory management** and Python object \leftrightarrow c data type **conversions** done automatically.
 - malloc, realloc, free still used for C memory management

Why Cython?

- Optimize **only** what you need

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration
 - **Code** and **developers**

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration
 - **Code** and **developers**
 - Piece by piece

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration
 - **Code** and **developers**
 - Piece by piece
- Focus on algorithm

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration
 - **Code** and **developers**
 - Piece by piece
- Focus on algorithm
 - ...not the boring, tedious, error-prone boilerplate code

Why Cython?

- Optimize **only** what you need
 - Most time spent in little code
- Easy migration
 - **Code** and **developers**
 - Piece by piece
- Focus on algorithm
 - ...not the boring, tedious, error-prone boilerplate code
 - Use anything from Sage, and from elsewhere

Using Cython in Sage

- From the **command line**
 - Load or attach normal Sage scripts with an **.spyx** extension.

Using Cython in Sage

- From the **command line**
 - Load or attach normal Sage scripts with an **.spyx** extension.
- In **Sage library** code
 - Create files with a **.pyx** extension and add to `setup.py`

Using Cython in Sage

- From the **command line**
 - Load or attach normal Sage scripts with an **.spyx** extension.
- In **Sage library** code
 - Create files with a **.pyx** extension and add to `setup.py`
 - There are also `.pxd` (declaration) and `.pxi` (include) files.

Using Cython in Sage

- From the **command line**
 - Load or attach normal Sage scripts with an **.spyx** extension.
- In **Sage library** code
 - Create files with a **.pyx** extension and add to `setup.py`
 - There are also `.pxd` (declaration) and `.pxi` (include) files.
- From the **Notebook**
 - Directly and **interactively** in a **%cython** block.

Demo

Cython vs. Python

Axiom

Python is a developer friendly language.

Cython vs. Python

Axiom

Python is a developer friendly language.

Fact

Python is slow.

Cython vs. Python

Python is **slow** because of...

- its interpreter

Cython vs. Python

Python is **slow** because of...

- its interpreter
- dictionary lookups

Cython vs. Python

Python is **slow** because of...

- its interpreter
- dictionary lookups
- complicated calling conventions

Cython vs. Python

Python is **slow** because of...

- its interpreter
- dictionary lookups
- complicated calling conventions
- object-oriented primitives

Cython vs. Python

Python is **slow** because of...

- its interpreter
 - *Cython is compiled*
- dictionary lookups
- complicated calling conventions
- object-oriented primitives

Cython vs. Python

Python is **slow** because of...

- its interpreter
 - *Cython is compiled*
- dictionary lookups
 - *Cython has cdef attributes*
- complicated calling conventions

- object-oriented primitives

Cython vs. Python

Python is **slow** because of...

- its interpreter
 - *Cython is compiled*
- dictionary lookups
 - *Cython has cdef attributes*
- complicated calling conventions
 - *Cython has cdef functions*
- object-oriented primitives

Cython vs. Python

Python is **slow** because of...

- its interpreter
 - *Cython is **compiled***
- dictionary lookups
 - *Cython has **cdef attributes***
- complicated calling conventions
 - *Cython has **cdef functions***
- object-oriented primitives
 - *Cython has **cdef types***

The magic word `cdef`

The keyword `cdef` can be used for

The magic word `cdef`

The keyword `cdef` can be used for

- Local variable declarations

The magic word `cdef`

```
def my_sum(N):  
    s = 0  
    for k in range(N):  
        s += k  
    return s
```


The magic word `cdef`

```
def my_sum(long N):  
    cdef long k, s = 0  
    for k in range(N):  
        s += k  
    return s
```

The magic word `cdef`

The keyword `cdef` can be used for

- Local variable declarations
- Function declarations

The magic word `cdef`

```
def my_sum(long N):  
    cdef long k, s = 0  
    for k in range(N):  
        s += k  
    return s
```

The magic word `cdef`

```
cdef long sum(long N):  
    cdef long k, s = 0  
    for k in range(N):  
        s += k  
    return s
```

The magic word `cdef`

The keyword `cdef` can be used for

- Local variable declarations
- Function declarations
- Classes

The magic word `cdef`

The keyword `cdef` can be used for

- Local variable declarations
- Function declarations
- Classes
 - `cdef` methods

The magic word `cdef`

The keyword `cdef` can be used for

- Local variable declarations
- Function declarations
- Classes
 - `cdef` methods
 - `cdef` attributes

The magic word cdef

The `cdef` keyword is also used to interface with **external libraries**

```
def test_mpz():  
    cdef mpz_t a  
    mpz_init(a)  
    mpz_set_ui(a, 3)  
    mpz_pow(a, a, 100)  
    mpz_clear(a)
```


Disadvantages of cdef

By default, cdef attributes and methods are not accessible from Python.

- cdef **public** attribute

Disadvantages of cdef

By default, cdef attributes and methods are not accessible from Python.

- cdef **public** attribute
- **cpdef** functions

Disadvantages of cdef

By default, cdef attributes and methods are not accessible from Python.

- cdef **public** attribute
- **cpdef** functions

But sometimes you want things to be private.

Common Pitfalls

- Too much Python
- Unnecessary conversions
- Untyped objects

Common Pitfalls

Too much Python

```
def fib_list(long N):  
    cdef long i  
    L = [0,1]  
    for i in range(N):  
        L.append(L[-1] + L[-2])  
    return L
```

Common Pitfalls

Too much Python

```
def fib_list(long N):  
    cdef long i  
    L = [0,1]  
    for i in range(N):  
        L.append(L[-1] + L[-2])  
    return L
```

Common Pitfalls

Unnecessary conversions

```
def sum_squares(long N):  
    cdef long k, s = 0  
    for k from 0 <= k < N:  
        a = k*k  
        s += a  
    return s
```

Common Pitfalls

Unnecessary conversions

```
def sum_squares(long N):  
    cdef long k, s = 0  
    for k from 0 <= k < N:  
        a = k*k  
        s += a  
    return s
```


Common Pitfalls

Untyped Objects

```
cdef class Dice:
    cpdef int roll(self):
        return randint(1,7)

def n_rolls(n):
    dice = Dice()
    cdef int i, s
    for i from 0 <= i < n:
        s += dice.roll()
    return s
```

Common Pitfalls

Untyped Objects

```
cdef class Dice:
    cpdef int roll(self):
        return randint(1,7)

def n_rolls(n):
    dice = Dice()
    cdef int i, s
    for i from 0 <= i < n:
        s += dice.roll()
    return s
```

Cython References



Web page: <http://www.cython.org>

Mercurial: <http://hg.cython.org>

Wiki: <http://wiki.cython.org>

Bugtracker: <https://launchpad.net/cython>

Mailing list: cython-dev@lists.berlios.de

And, of course, it comes free with every copy of Sage.

Questions?

