What's Cython?

http://www.cython.org

Craig Citro (in lieu of Robert Bradshaw)
UCLA Math / UW Math

January 31, 2009

Outline

Outline

Where's Rob?

Rob couldn't make it today, due to other obligations:

What's Cython?

Cython is a language extremely close to Python that allows you to:

- write **extremely** fast code,
- stay happily oblivious to the Python/C API,
- easily mix Python and C types, and
- use C/C++ libraries from Python with a minimal amount of pain and heartache.

Examples

```
In [1]: def mysum(N):
   ...:     s = 0
   ...:     for k in range(N):
   ...:         s += k
   ...:     return s

In [2]: %time mysum(10**6)
CPU times: user 0.28 s, sys: 0.00 s, total: 0.29 s
Wall time: 0.29 s
Out[3]: 499999500000L

In [4]: def mysum2(N):
   ...:     return sum(range(N))

In [5]: %time mysum2(10**6)
CPU times: user 0.21 s, sys: 0.00 s, total: 0.22 s
Wall time: 0.22 s
Out[6]: 499999500000L
```

Examples

```
def mysum_c(N):
    cdef int k
    cdef long long s = 0

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

So we compile this bit of Cython code, and we have:

```
In [7]: from examples import mysum_c

In [8]: %time mysum_c(10**6)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
Wall time: 0.00 s
Out[9]: 499999500000L
```

Examples

Yeah, this one is just a **wee** bit faster:

```
In [10]: %timeit mysum(10**6)
10 loops, best of 3: 283 ms per loop

In [11]: %timeit mysum_c(10**6)
100 loops, best of 3: 1.22 ms per loop

In [12]: 283/1.22
Out[12]: 231.96721311475412
```

Examples

Of course, there are limitations:

```
In [15]: mysum_c(10**10)
...
OverflowError: long int too large to convert to int
```

Outline

Cython

Cython (`http://www.cython.org`) lets you:

- declare attributes for your classes with C datatypes
- declare methods to take and return C datatypes
- interface with your existing C/C++ libraries

Cython

No one wants to declare types for all of their objects, and manually allocate and deallocate our C objects – this is one of the reasons we aren't using C in the first place!

We don't have to. The Cython development model:

- Write code in Python.
- Get it working **correctly**.
- Profile the code.
- Move the inner loops to Cython.

Cython: It Works

Jason Grout:

> I spent two or three days working on this. Here is the end result: 0.24
> seconds compared to 150 seconds. Such is the power of Cython :). That's
> a speedup of a factor of 150.64/0.24=627!

This particular function, because it is so fast now, has become a regular tool in our research and has led to discovering at least one counter-example to a conjecture that was open for several months.

One `def` to rule them all ...

There are three ways to declare a function in Cython:

- `def`: The usual Python declaration; uses Python calling conventions, and takes Python types
- `cdef`: A C declaration; uses C calling conventions, takes Python or C types
- `cpdef`: The best of both worlds

Different `def`s for different folks ...

Let's see an example:

```
def extend_py(self, d):
    self._length += d

cdef extend_c(self, int d):
    self._length += d

cpdef extend(self, int d):
    self._length += d
```

Different `def`s for different folks ...

```
In [3]: %time b.time_test(1, 10**7, 'def')
CPU times: user 1.55 s, sys: 0.00 s, total: 1.56 s
Wall time: 1.57 s

In [5]: %time b.time_test(1, 10**7, 'cdef')
CPU times: user 0.07 s, sys: 0.00 s, total: 0.07 s
Wall time: 0.07 s

In [7]: %time b.time_test(1, 10**7, 'cpdef')
CPU times: user 0.09 s, sys: 0.00 s, total: 0.09 s
Wall time: 0.09 s
```

Different `defs` for different folks ...

```
In [4]: %time for _ in range(10**7): b.extend_py(1)
CPU times: user 2.74 s, sys: 0.15 s, total: 2.89 s
Wall time: 2.93 s

In [6]: %time for _ in range(10**7): b.extend(1)
CPU times: user 2.85 s, sys: 0.04 s, total: 2.89 s
Wall time: 2.92 s
```

Cython

Not that you needed any more reasons, but here are a few more amazing things that Cython has to offer:

- Built-in profiling/annotation tools for performance analysis
- Automatic conversion between most Python and C types (whenever it would make sense)
- Cython can also be used to interface with C++ libraries (only a small amount of black magic needed!)

Outline

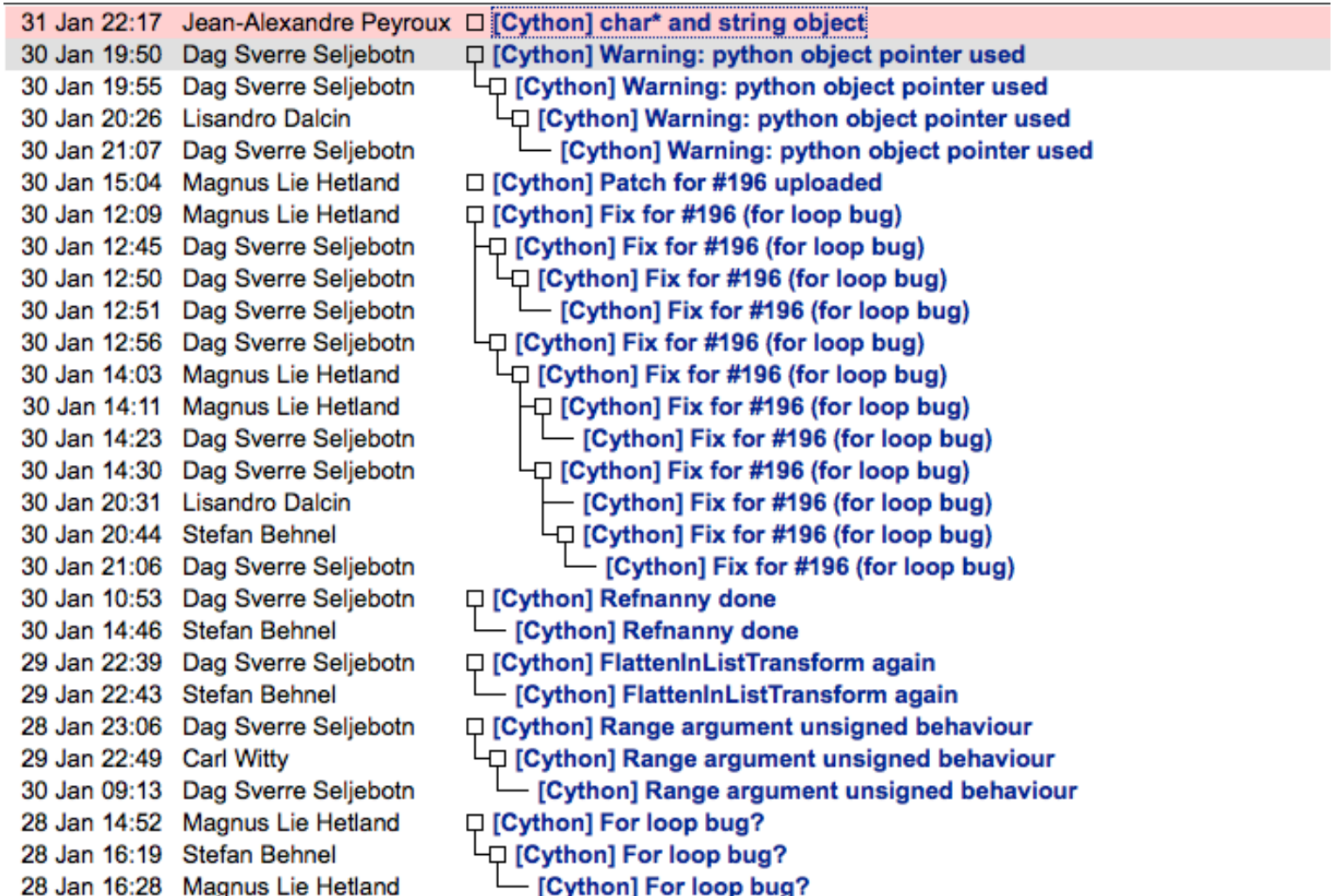
Cython

Cython is open source, freely available under the Apache License.

```
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
```

There are more than twelve Cython developers . . .

- Head developers: Stephan Behnel, Robert Bradshaw
- Dag Sverre Seljebotn (Google Summer of Code 2008): Tight integration of Cython types and buffer types (see PEP 3118), used by Numpy and PIL
- Large, active development community:



Cython under many names

A quick history:

- Cython is a fork of the Pyrex project, started by Greg Ewing (first released in 2002)
- Began life as part of the Sage project (and originally called "SageX"), in 2006
- Lots of outside interest, particularly from Stefan Behnel (who was maintaining another Pyrex fork, `lxml`)
- Cython first launched in 2007

Would you like to know more?

There's a lot of interesting stuff I didn't get to talk about ...

- Cython support for built-in types (`cdef list ls` ...)
- Exposing Cython classes (`.pxd` files for declarations, ...)
- Automatic coercion between Python types and C/C++ types

Does it cook breakfast, too?

So there are still a few things not supported in Cython. Most of these are simply just a lack of developer time so far:

- Closures
- Closures
- Closures
- Generators
- Multiple Inheritance (no plan right now ...)
- Other various bits: `http://wiki.cython.org/Unsupported`

Outline

Any questions?

Thanks for listening!