Why

- Python is nice, easy, development is fast
- However, Python is slow
- The bottlenecks can be rewritten:
  - SWIG
  - Boost.Python
  - numba
  - Cython
Cython

What’s Cython?

- Python with C data types
- Any* Python code is valid Cython code
- Translate the code into C/C++ code. Use it as modules
- You can call C libraries
- Code using Python values and C values can be intermixed (automatic conversions)
- The more type information you provide the better the compile
First Example

Use iPython

```python
In [1]: %load_ext cythonmagic
In [2]: %%cython
   import math
   def first_cython(int arg):
       return math.sqrt(arg**9/13 + 7*arg**3 + 29)**3
In [3]: first_cython(100)
```

A. Gómez 4 Speeding up Python
How Much Faster

Use iPython

In [1]: import math
In [2]: def first_python(arg):
   ...:     return math.sqrt(arg**9/13 + 7*arg**3 + 29)**3
In [3]: %timeit first_python(20)

In [4]: %load_ext cythonmagic

In [5]: %%cython
   ...: import math
   ...: def first_cython(arg):
   ...:     return math.sqrt(arg**9/13 + 7*arg**3 + 29)**3
In [6]: %timeit first_cython(20)

In [7]: %%cython
   ...: import math
   ...: def fast_cython(int arg):
   ...:     return math.sqrt(arg**9/13 + 7*arg**3 + 29)**3
In [8]: %timeit fast_cython(20)
Cython Functions

- Python functions are defined with `def`. They take Python objects as parameters and return Python objects.
- C functions are defined with `cdef`. They take either Python objects or C values and return Python objects or C values.
- Both can call each other within a Cython module.
- Only Python functions can be called from outside the model by Python code.
Type Declaration

- **cdef**: static typization

```python
1 cdef double var
2 cdef int arr[50]
```

- **cdef**: as C function:

```python
1 cdef double function(double arg):
2     return arg**2
```

- **cdef** class:

```python
1 cdef class MyClass:
```

- **cdef** struct:

```python
1 cdef struct my_struct:
2     int var1
3     double var2
```

- **Several declarations into the same cdef**

```python
1 cdef:
2     int i
3     double d
4     void f (arg):
5         return arg**2
```
Example

```
import math

def function(arg):
    res = 0.0
    for i in range(50000000):
        res += math.sqrt((i+1)*arg**5)
    return res

print function(10.0)
```

> time python test_python.py

A. Gómez

```
examples/2_cython/test_python.py
```

Speeding up Python
Example

```python
import math

def function(double arg):
    cdef double res = 0.0
    for i in range(50000000):
        res += math.sqrt((i+1)*arg**5)
    return res
```

```python
import myfunc1
print myfunc1.function(10.0)
```

> cython myfunc1.pyx
> icc -shared -fpic -O3
    myfunc1.c -o myfunc1.so
    -I$TACC_PYTHON_INC/python2.7/
> time python test_cython1.py
A. Gómez
9 Speeding up Python
```python
import math

cdef double f(double arg):
    cdef double res = 0.0
    cdef int i = 0
    for i in range(50000000):
        res += math.sqrt((i+1)*arg**5)
    return res

def function(double arg):
    return f(arg)
```

```python
import myfunc2
print myfunc2.function(10.0)
```

> cython myfunc2.pyx
> icc -shared -fPIC -O3
  myfunc2.c -o myfunc2.so
-I$TACC_PYTHON_INC/python2.7/
> time python test_cython2.py

A. Gómez 10 Speeding up Python
Example

```python
from libc.math cimport sqrt
# cdef extern from "math.h":
# double sqrt(double x)

cdef double f(double arg):
    cdef double res = 0.0
    cdef int i = 0
    for i in range(50000000):
        res += sqrt((i+1)*arg**5)
    return res

def function(double arg):
    return f(arg)
```

```python
import myfunc3
print myfunc3.function(10.0)
```

> cython myfunc3.pyx
> icc -shared -fPIC -O3 myfunc3.c -o myfunc3.so
-1$TACC_PYTHON_INC/python2.7/
> time python test_cython3.py

A. Gómez 11 Speeding up Python
Cython

- Easy to decorate your own code
- Lot of potential
- Iterative process
- Link to a great tutorial
License

© The University of Texas at Austin, 2014

This work is licensed under the Creative Commons Attribution Non-Commercial 3.0 Unported License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc/3.0/

When attributing this work, please use the following text: "HPC Python", Texas Advanced Computing Center, 2014. Available under a Creative Commons Attribution Non-Commercial 3.0 Unported License.