Advanced Programming for Data Science Lab

DS-GA 3001.002

Ma. Elena Villalobos Ponte
mvp291@nyu.edu

An introduction to Cython

- *Back in our first lab*
  - ***Why is Java faster than Python?***
    - *Dynamic typing*
      Types have to be checked at run-time
    - *Can we have **[optional]** static typing to make our Python code more efficient?*
    - *Given that we can, can we translate our Python code into optimized native C code and export it as a module for Python?*

An introduction to Cython

- **Cython** is an **optimising static compiler** for both the **Python** programming language and the **extended Cython** programming language.

- It makes writing C extensions for Python as easy as Python itself.

Cython: C-Extensions for Python
http://cython.org/

An introduction to Cython

- The **Cython programming language** is a **superset** of the Python language that additionally supports:
  - Calling C functions
  - Declaring C types on variables, function parameters and class attributes.
- This allows the compiler to generate very efficient C code from Cython code.

Cython: C-Extensions for Python
http://cython.org/

Building Cython code

- Cython code must, unlike Python, be compiled. This happens in two stages:

1. A **.pyx** file is compiled by Cython to a **.c** file, containing the code of a Python extension module

2. The **.c** file is compiled by a C compiler to a **.so** file (or **.pyd** on Windows) which can be imported directly into a Python session.

Cython: C-Extensions for Python
http://docs.cython.org/src/quickstart/build.html

Building Cython code

- There are several ways to build Cython code:
  - Write a distutils setup.py.
  - Use pyximport, importing Cython .pyx files as if they were .py files (using distutils to compile and build in the background).
  - Run the cython command-line utility manually to produce the .c file from the .pyx file, then manually compiling the .c file into a shared object library or DLL suitable for import from Python. (These manual steps are mostly for debugging and experimentation.)
  - **Use the [IPython] notebook or the [Sage] notebook, both of which allow Cython code inline.**

Cython Syntax
Types

Basic C types

Type	Description
`char`	8 bit integer
`short`	16 bit integer
`int`	32 bit integer
`long`	64 bit integer
`long long`	64 bit integer
`float`	32 bit floating point
`double`	64 bit floating point
`long double`	80 bit floating point
`float a[10][30]`	2-dimensional array
`char *s`	pointer
`struct foo`	structure
`union bar`	union
`enum type`	enumeration

Language Basics — Cython 0.23.2 documentation
http://docs.cython.org/src/userguide/language_basics.html

Cython Syntax
Variable Declaration

How to define variables?

- Use **`cdef`** statement

```
cdef int i, j, k
cdef float f, g[42], *h
```

and C **struct**, **union** or **enum** types:

```
cdef struct Grail:
    int age
    float volume

cdef union Food:
    char *spam
    float *eggs

cdef enum CheeseType:
    cheddar, edam,
    camembert

cdef enum CheeseState:
    hard = 1
    soft = 2
    runny = 3
```

Language Basics — Cython 0.23.2 documentation
http://docs.cython.org/src/userguide/language_basics.html

- cdef block for multiple declarations

```
cdef:
    struct Spam:
        int tons

    int i
    float f
    Spam *p

    void f(Spam *s):
        print s.tons, "Tons of spam"
```

- Also **`ctypedef`** statement to give a name

```
ctypedef unsigned long ULong

ctypedef int* IntPtr
```

Cython Syntax

Function definition

Python	C	Hybrid
`def`	**`cdef`**	**`cpdef`**
Can be called from Python and Cython module	Can only be called by Cython module	Can be called from Python and Cython module
Basically, it's Python	Basically, it's C	It's both

Language Basics — Cython 0.23.2 documentation
http://docs.cython.org/src/userguide/language_basics.html

Cython Syntax
Function definition

- Parameters of either type of function can be declared to have C data types, using normal C declaration syntax.

```
def spam(int i, char *s):
    ...

cdef int eggs(unsigned long l, float f):
    ...
```

Notice the type of the returning value is specified

is equivalent to

```
def spam(python_i, python_s):
    cdef int i = python_i
    cdef char* s = python_s
    ...
```

Language Basics — Cython 0.23.2 documentation
http://docs.cython.org/src/userguide/language_basics.html

Cython Syntax
Function definition

- When a parameter of a Python function is declared to have a C data type, it is passed in as a Python object and automatically converted to a C value, if possible.
- Automatic conversion is currently only possible for numeric types, string types and structs.
- Attempting to use any other type for the parameter of a Python function will result in a compile-time error.
- C functions can have parameters of any type, since they're passed in directly using a normal C function call.
- If no type is specified for a parameter or return value, it is assumed to be a Python object.

Language Basics — Cython 0.23.2 documentation
http://docs.cython.org/src/userguide/language_basics.html