Using Cython to write C and C++ Extentions to Python

Thursday, 23 February 2012
Last Updated Thursday, 23 February 2012

If you want to easyly write C and C++ extensions (to Python) that incorporate Python syntax, C and C++ functions you can use Cython.

To install Cython from FreeBSD ports:

```
cd /usr/ports/lang/cython
make install clean
```

Then let's see a simple example that display numbers from 1 to n, where n is a given parameter to a function we will define.

First we define the pyx source file:

myfunc.pyx#!/usr/local/bin/python

```
def display_numbers(upper):
    for i in range(1,upper+1):
      print i
```

Then we define setup.py file that will build the C source code for our extension:

setup.pyfrom distutils.core import setup

from distutils.extension import Extension

from Cython.Distutils import build_ext

```
ext_modules = [Extension("myfunc", ["myfunc.pyx"])]

setup(
  name = 'myfunc',
  cmdclass = {'build_ext': build_ext},
  ext_modules = ext_modules
)
```

And then will build the binary extension to Python (run next command from shell command line):

```
python setup.py build_ext --inplace
```

Then we could run Python and from command prompt:

```
>>> from myfunc import display_numbers
>>> display_numbers(7)
1
2
3
4
5
6
7
```

Or we could write a small Python script that uses myfunc module/extenstion (run next example from command line):

view.py#!/usr/local/bin/python

import myfunc

myfunc.display_numbers(5)
Note: Cython is also usefull if you want to build an application but you do not want to share all source code. By using only Python code, the app can be reverse engineered but using Cython this will not be possible, at least to get the same source code.