Cython: Stop writing native Python extensions in C

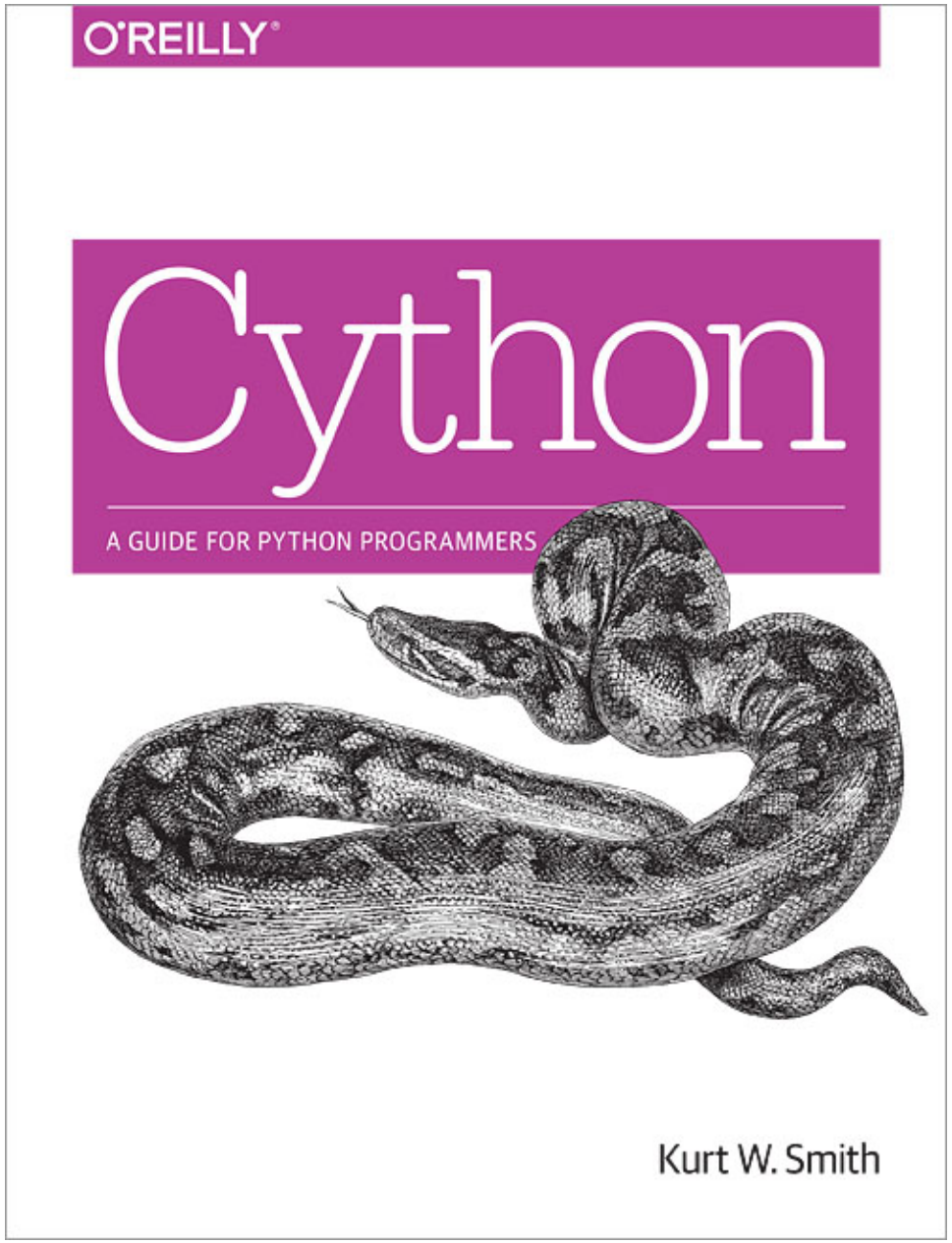
Miro Hrončok

March 29, 2016

About Cython

- cython.org
- programming language
  - similar to Python
  - static typing from C/C++
- compiler
  - from Cython language to C/C++
  - to Python extension module
  - or to standalone apps*
- feels like Python
- woks like C/C++

Cython: A Guide for Python Programmers



Python extension modules

- Extending Python with C or C++
- I refer to them as "native extensions" a lot
- performance
- interacting with C/C++ libraries/code
- `#include <Python.h>`
- it hurts to write
- it hurts to read
- it hurts to maintain
- it hurts to keep it compatible with both Pythons
- Cython makes all this easy

Fibonacci

```
def pyfib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = a + b, a
    return a


# In IPython, try to run it like:
%timeit pyfib(190)
```

Fibonacci

```
def pyfib(n):
    a, b = 0, 1
    for i in range(n):
        a, b = a + b, a
    return a
```

```
def cyfib(int n):
    cdef int i
    cdef long a = 0, b = 1
    for i in range(n):
        a, b = a + b, a
    return a
```

Fibonacci

```
%load_ext Cython

%%cython
def cyfib(int n):
    cdef int i
    cdef long a = 0, b = 1
    for i in range(n):
        a, b = a + b, a
    return a

%timeit cyfib(190)
```

setuptools

```
from setuptools import setup, Extension
from Cython.Distutils import build_ext

setup(
    cmdclass={'build_ext': build_ext},
    ext_modules=[
        Extension('fib', ['fib.pyx'])],
    classifiers=[
        'Programming Language :: Cython'],
)
```

Typing

```
a = [x + 1 for x in range(12)]
b = a
a[3] = 42.0
assert b[3] == 42.0
a = 13
assert isinstance(b, list)
```

Typing

```
def dummy_func():
    cdef int i
    cdef int j
    cdef float k

    j = 0
    i = j
    k = 12.0
    j = 2 * k
    assert i != j
```

Typing

```
def several_at_once():
    cdef int i, j, k
    cdef float price, margin
    #...

def optional_initial_value():
    cdef int i = 0
    cdef long int j = 0, k = 0
    cdef float price = 0.0, margin = 1.0
    #...
```

Pointers, Arrays, bint...

```
cdef int *p
cdef void **buf = NULL
cdef void (*func)(int, double)

cdef size_t arr[3]

cdef double golden_ratio
cdef double *p_double

p_double = &golden_ratio
p_double[0] = 1.618
print(golden_ratio)
print(p_double[0])

cdef bint ok
```

Structs, union

```
cdef struct coord:
    float x
    float y
    float z

cdef union uu:
    int a
    short b, c

ctypedef struct coord:
    #...

ctypedef union uu:
    #...

cdef uu myvar
```

Structs initialization

```
cdef coord a = coord(0.0, 2.0, 1.5)

cdef coord b = coord(x=0.0, y=2.0, z=1.5)

cdef coord c
c.x = 42.0
c.y = 2.0
c.z = 4.0

cdef coord d = {'x':2.0,
                'y':0.0,
                'z':-0.75}
```

Cython's Three Kinds of Functions

- **def**
- **cdef**
- **cpdef**

```
def cyfib(int n):
    cdef int i
    cdef long a = 0, b = 1
    for i in range(n):
        a, b = a + b, a
    return a

cdef long cyfib(int n):
    #...

cpdef long cyfib(int n):
    #...
```

Classes and Extension Types

```
from libc.stdlib cimport malloc, free

cdef class Matrix:
    cdef readonly unsigned int nr, nc
    cdef double *_matrix

    def __cinit__(self, nr, nc):
        self.nr, self.nc = nr, nc
        self._matrix = <double*>malloc(nr
            * nc * sizeof(double))
        if self._matrix == NULL:
            raise MemoryError()

    def __dealloc__(self):
        if self._matrix != NULL:
            free(self._matrix)
```

Knapsack Problem

- problem from combinatorial optimization
- Wikipedia
- you have a knapsack with a given capacity
- you get bunch of items with costs and weights
- get the most expensive combination that fits in
- bruteforce is $O(2^n)$
- get it from github.com/hroncok/cython-workshop

Knapsack Problem

```
sack = {
    'id': 1,
    'count': 15,
    'capacity': 18.5,
    'items': [
        {'weight': 2.0, 'cost': 2.5},
        #...
    ],
}

solver = BruteSolver(sack)
maxcombo, maxcost = solver.solve()
```

Further information

- docs.cython.org:
  - Using C libraries
  - Working with NumPy
  - plenty of other topics
- Cython book
- examples from the book
- contact me
  - miro@redhat.com
  - mhroncok @ freenode
  - @hroncok on Twitter or GitHub