

Python and Cython

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation
working with numpy

- 1 The Meaning of Python
a dynamic language
rapid application development
- 2 Cython
installing Cython
factorization again
- 3 The Power Method
an illustration of an intensive computation
working with numpy

MCS 275 Lecture 39
Programming Tools and File Management
Jan Verschelde, 19 April 2010

Python and Cython

The Meaning of Python

a dynamic language

rapid application development

Cython

installing Cython

factorization again

The Power Method

an illustration of an intensive computation

working with numpy

1 The Meaning of Python

a dynamic language

rapid application development

2 Cython

installing Cython

factorization again

3 The Power Method

an illustration of an intensive computation

working with numpy

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

The Development Cycle

compilation versus interpretation

Traditional build cycle:

- 1 run the application
- 2 test the behavior of the code
- 3 stop the application
- 4 edit the program code
- 5 recompile the code
- 6 relink executable
- 7 goto step 1

This is the static language build cycle.

Python eliminates steps 4 and 5.

Recompilation and relinking is not a trivial task
for systems with over 100,000 lines of code.

Python is a dynamic Language

some advertisements

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Python may be a scripting language,
but it scales well.

- “Python is executable pseudocode.”
very readable,
complexities involving memory addresses are hidden
- “Python is OOP done right.”
compared to C++,
the class mechanism in Python is much simpler

The statements are quotes taken from Mark Lutz:
Programming Python book. 2nd edition, O’Reilly 2001.

Python is a dynamic Language

some advertisements

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Python may be a scripting language,
but it scales well.

- “Python is executable pseudocode.”
very readable,
complexities involving memory addresses are hidden
- “Python is OOP done right.”
compared to C++,
the class mechanism in Python is much simpler

The statements are quotes taken from Mark Lutz:
Programming Python book. 2nd edition, O’Reilly 2001.

Python is a dynamic Language

some advertisements

Python may be a scripting language,
but it scales well.

- “Python is executable pseudocode.”
very readable,
complexities involving memory addresses are hidden
- “Python is OOP done right.”
compared to C++,
the class mechanism in Python is much simpler

The statements are quotes taken from Mark Lutz:
Programming Python book. 2nd edition, O’Reilly 2001.

Python and Cython

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation
working with numpy

1 The Meaning of Python

a dynamic language

rapid application development

2 Cython

installing Cython

factorization again

3 The Power Method

an illustration of an intensive computation

working with numpy

19 Apr 2010

The Meaning
of Pythona dynamic language
rapid application
development

Cython

installing Cython
factorization againThe Power
Methodan illustration of an
intensive
computation
working with numpy

Rapid Application Development

an answer to the software crisis

prototyping:

all Python

hybrid:

mixture of Python and C/C++

delivery:

all C/C++

Consider the slider:

prototyping

hybrid

delivery

all Python

mixture

all C/C++

Identifying the bottlenecks in a working prototype leads to a gradual development of modules in C/C++.

Python complements languages like C/C++ and Java.

19 Apr 2010

Rapid Application Development

an answer to the software crisis

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

prototyping:

all Python

hybrid:

mixture of Python and C/C++

delivery:

all C/C++

Consider the slider:

prototyping

hybrid

delivery

all Python

mixture

all C/C++

Identifying the bottlenecks in a working prototype leads to a gradual development of modules in C/C++.

Python complements languages like C/C++ and Java.

19 Apr 2010

Rapid Application Development

an answer to the software crisis

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

prototyping:

all Python

hybrid:

mixture of Python and C/C++

delivery:

all C/C++

Consider the slider:

prototyping
all Python

hybrid
mixture

delivery
all C/C++

Identifying the bottlenecks in a working prototype leads to a gradual development of modules in C/C++.

Python complements languages like C/C++ and Java.

19 Apr 2010

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Sinking the Titanic

the iceberg that sank so many large software
projects...

The Titanic refers to the iceberg. What we see is the exposed interface (the domain of Python), what is hidden are the system internals (the domain of C/C++).

"Python provides a simple but powerful rapid development language, *along with the integration tools to apply it in realistic development environments.*"

Mark Lutz, page 1195

The idea of integration is not new,
but the innovation is in providing the tools.

Python is open source and cross platform.

19 Apr 2010

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Sinking the Titanic

the iceberg that sank so many large software
projects...

The Titanic refers to the iceberg. What we see is the exposed interface (the domain of Python), what is hidden are the system internals (the domain of C/C++).

"Python provides a simple but powerful rapid development language, *along with the integration tools to apply it in realistic development environments.*"

Mark Lutz, page 1195

The idea of integration is not new,
but the innovation is in providing the tools.

Python is open source and cross platform.

19 Apr 2010

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Sinking the Titanic

the iceberg that sank so many large software
projects...

The Titanic refers to the iceberg. What we see is the exposed interface (the domain of Python), what is hidden are the system internals (the domain of C/C++).

"Python provides a simple but powerful rapid development language, *along with the integration tools to apply it in realistic development environments.*"

Mark Lutz, page 1195

The idea of integration is not new,
but the innovation is in providing the tools.

Python is open source and cross platform.

Python and Cython

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation
working with numpy

- 1 The Meaning of Python
a dynamic language
rapid application development
- 2 Cython
installing Cython
factorization again
- 3 The Power Method
an illustration of an intensive computation
working with numpy

Installing Cython

to install Cython (on Mac OS X):

- download from `http://cython.org` and unpack tar ball or zip file
- in the `Cython-0.12` directory, at command prompt, type

```
python setup.py install
```

at the command prompt.

Note that the `gcc` compiler must be installed.

Trying the example on

`http://docs.cython.org/src/quickstart/build.html`

- making the file `hello.pyx`
- making the file `setup.py`

Installing Cython

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

to install Cython (on Mac OS X):

- download from `http://cython.org` and unpack tar ball or zip file
- in the `Cython-0.12` directory, at command prompt, type

```
python setup.py install
```

at the command prompt.

Note that the `gcc` compiler must be installed.

Trying the example on

`http://docs.cython.org/src/quickstart/build.html`

- making the file `hello.pyx`
- making the file `setup.py`

a hello world example

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

Content of the file `hello.pyx`:

```
def say_hello_to(name):  
    print("hello %s!" % name)
```

Content of `setup.py`:

```
from distutils.core import setup  
from distutils.extension import Extension  
from Cython.Distutils import build_ext  
  
ext_modules = [Extension("hello", ["hello.pyx"])]  
  
setup(  
    name = 'Hello world app',  
    cmdclass = { 'build_ext': build_ext },  
    ext_modules = ext_modules  
)
```

a hello world example

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

Content of the file `hello.pyx`:

```
def say_hello_to(name):  
    print("hello %s!" % name)
```

Content of `setup.py`:

```
from distutils.core import setup  
from distutils.extension import Extension  
from Cython.Distutils import build_ext  
  
ext_modules = [Extension("hello", ["hello.pyx"])]  
  
setup(  
    name = 'Hello world app',  
    cmdclass = { 'build_ext': build_ext },  
    ext_modules = ext_modules  
)
```

Building and Using It

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

At the command prompt, type

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

This creates a C program, a shared object file,
and a directory `build`.

Ready for use in a Python session:

```
$ python
Python 2.6.4 (r264:75821M, Oct 27 2009, 19:48:32)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" fo
>>> from hello import say_hello_to
>>> say_hello_to("me")
hello me!
>>>
```

Building and Using It

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

At the command prompt, type

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

This creates a C program, a shared object file,
and a directory `build`.

Ready for use in a Python session:

```
$ python
Python 2.6.4 (r264:75821M, Oct 27 2009, 19:48:32)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" fo
>>> from hello import say_hello_to
>>> say_hello_to("me")
hello me!
>>>
```

Building and Using It

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

At the command prompt, type

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

This creates a C program, a shared object file,
and a directory `build`.

Ready for use in a Python session:

```
$ python
Python 2.6.4 (r264:75821M, Oct 27 2009, 19:48:32)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" fo
>>> from hello import say_hello_to
>>> say_hello_to("me")
hello me!
>>>
```

Python and Cython

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation
working with numpy

- 1 The Meaning of Python
a dynamic language
rapid application development
- 2 **Cython**
installing Cython
factorization again
- 3 The Power Method
an illustration of an intensive computation
working with numpy

factorization again

In the file `factors_of_number.pyx` we have:

```
def factors_of_number(n):  
    """  
    Prints the factorization of the number n.  
    """  
    d = 2; s = '%d =' % n  
    while(d < n):  
        (q,r) = divmod(n,d)  
        if(r == 0):  
            s = s + ' %d' % d  
            n = q; d = 2  
        else:  
            d = d + 1  
    s = s + ' %d' % n  
    print s
```

a modified setup.py

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

```
from distutils.core import setup
from distutils.extension import Extension
from Cython.Distutils import build_ext

ext_modules = [Extension("factors_of_number",
                        ["factors_of_number.pyx"])]

setup(
    name = 'factors of number app',
    cmdclass = { 'build_ext': build_ext },
    ext_modules = ext_modules
)
```

Building and Using It

To build, we again type:

```
$ python setup build\_ext --inplace
```

Then we can test:

```
$ python
Python 2.6.4 (r264:75821M, Oct 27 2009, 19:48:32)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "help", "copyright", "credits" or "license" fo
>>> from factors_of_number import factors_of_number
>>> factors_of_number(1230)
1230 = 2 3 5 41
>>>
```

19 Apr 2010

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

Python and Cython

- 1 The Meaning of Python
a dynamic language
rapid application development
- 2 Cython
installing Cython
factorization again
- 3 The Power Method
an illustration of an intensive computation
working with numpy

The Power Method

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

Let A be any n -by- n matrix.

The vector $v \neq 0$ is an eigenvector if $Av = \lambda v$,
for some λ , *lambda* is then an eigenvalue.

A simple, yet powerful recipe to compute the eigenvector
corresponding to the largest eigenvalue:

```
y = x
for i in range(0,m):
    y = A*x
    y = y/norm(y)
```

The Power Method

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation
working with numpy

Let A be any n -by- n matrix.

The vector $v \neq 0$ is an eigenvector if $Av = \lambda v$,
for some λ , λ is then an eigenvalue.

A simple, yet powerful recipe to compute the eigenvector
corresponding to the largest eigenvalue:

```
y = x
for i in range(0,m):
    y = A*x
    y = y/norm(y)
```

Python and Cython

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation

working with numpy

- 1 The Meaning of Python
a dynamic language
rapid application development
- 2 Cython
installing Cython
factorization again
- 3 The Power Method
an illustration of an intensive computation
working with numpy

matrix-vector multiplication

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation

working with numpy

```
from numpy import *

def MatrixVectorMultiply(A,x):
    """
    Returns the product of A with x.
    """

    n = A.shape[0]
    y = zeros((n,1),double)
    for i in range(0,A.shape[0]):
        for j in range(0,A.shape[1]):
            y[i] = y[i] + A[i,j]*x[j]
    return y
```

matrix-vector multiplication

The Meaning of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power Method

an illustration of an
intensive
computation

working with numpy

```
from numpy import *

def MatrixVectorMultiply(A,x):
    """
    Returns the product of A with x.
    """
    n = A.shape[0]
    y = zeros((n,1),double)
    for i in range(0,A.shape[0]):
        for j in range(0,A.shape[1]):
            y[i] = y[i] + A[i,j]*x[j]
    return y
```

normalizing

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

```
def Norm(x):  
    """  
    Returns the 2-norm of the vector x.  
    """  
    z = 0  
    for i in range(0,x.shape[0]):  
        z = z + x[i]*x[i]  
    return sqrt(z)
```

```
def NormalizedVector(x):  
    """  
    Returns x/Norm(x)  
    """  
    n = Norm(x)  
    y = x  
    for i in range(0,x.shape[0]):  
        y[i] = y[i]/n  
    return y
```

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation
working with numpy

```
def Norm(x):
    """
    Returns the 2-norm of the vector x.
    """
    z = 0
    for i in range(0,x.shape[0]):
        z = z + x[i]*x[i]
    return sqrt(z)

def NormalizedVector(x):
    """
    Returns x/Norm(x)
    """
    n = Norm(x)
    y = x
    for i in range(0,x.shape[0]):
        y[i] = y[i]/n
    return y
```

the power method

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation

working with numpy

```
def PowerMethod(A,x,n):  
    """  
    Given a square numpy matrix in A,  
    multiplies x n times with A.  
    """  
    y = x  
    for i in range(0,n):  
        y = MatrixVectorMultiply(A,y)  
        y = NormalizedVector(y)  
    return y
```

the main program

The Meaning
of Python

a dynamic language
rapid application
development

Cython

installing Cython
factorization again

The Power
Method

an illustration of an
intensive
computation

working with numpy

```
def main():  
    """  
    Test on the power method.  
    """  
    n = input('give the dimension : ')  
    a = random.normal(0,1,(n,n))  
    x = random.normal(0,1,(n,1))  
    m = input('give number of iterations : ')  
    y = PowerMethod(a,x,m)  
    print y  
    ans = raw_input('See all eigenvectors ? (y/n)  
    if ans == 'y':  
        [L,V] = linalg.eig(a)  
        print V
```

Summary + Assignments

The Meaning of Python

a dynamic language
rapid application development

Cython

installing Cython
factorization again

The Power Method

an illustration of an intensive computation

working with numpy

We returned to lecture 2.