Introduction to Cython - Week 1

Richard Killam

Friday 30th October, 2015

Outline

1. Motivation
2. I/O
   - Exercise
3. Conditionals
   - if, else, and elif
   - While
4. Iteration
   - Arrays
   - for loop
   - Exercise
5. Functions
   - Exercise
6. Importing
   - From Python Path
   - From Current Folder
7. Classes

Motivation

Sum the numbers from 1 to 100,000,000

Python	12.898s
Compiled Python	12.177s
Cython	0.432s
C	0.406s

Hello World

hello_world.py

```python
print("Hello World")
```

Hello World

hello_world.py

```
print("Hello World")
```

```
python hello_world.py
>> Hello World
```

My name is...

my_name_is.py

```
name = raw_input("Name > ")
print("Hello World, my name is
    {NAME}".format(NAME=name))
```

My name is...

my_name_is.py

```python
name = raw_input("Name > ")
print("Hello World, my name is {NAME}".format(NAME=name))
```

```
python my_name_is.py
>> Name >
>> Name > Richard
>> Hello World, my name is Richard
```

Exercise - My full name is...

Exercise - My full name is...

my_full_name_is.py

```
first_name = raw_input("First name > ")
last_name = raw_input("Last name > ")

print("Hello World, my name is {FIRST} {LAST}".format(
    FIRST=first_name,
    LAST=last_name
))
```

Getting numbers in

name_and_age.py

```python
name = raw_input("Name > ")
year_of_birth = int(raw_input("Year of Birth > "))

print("{NAME} is {YEARS} old.".format(
    NAME=name,
    YEARS=2015 - year_of_birth
))
```

if, else

True_False.py

```python
if True:
    print("The conditional was True")

else:
    print("The conditional was False")
```

if, else

True_False.py

```python
if True:
    print("The conditional was True")

else:
    print("The conditional was False")
```

Note the indentation.

if, else with input

even_or_odd.py

```
num = int(raw_input("Number > "))

if num % 2 == 0:
    print("{NUM} is even".format(NUM=num))

else:
    print("{NUM} is odd".format(NUM=num))
```

elif

a_or_b.py

```
choice = raw_input("Choose a or b ")

if choice == "a":
    print("You chose a")

elif choice == "b":
    print("You chose b")

else:
    print("You did not follow instructions...")
```

While + and

force_a_or_b.py

```
choice = ""

while choice != "a" and choice != "b":
     choice = raw_input("Choose a or b >")

if choice == "a":
    print("You chose a")
elif choice == "b":
    print("You chose b")
else:
    # Dead code
    print("How did you get here?!?!")
```

While + or + not

force_a_or_b.py

```
choice = ""

while not (choice == "a" or choice == "b"):
    choice = raw_input("Choose a or b >")

if choice == "a":
    print("You chose a")
elif choice == "b":
    print("You chose b")
else:
    # Dead code
    print("How did you get here?!?!")
```

Array declaration

array_declaration.py

```python
a = [1, 2, 3, 4, 5]
print(a) # >> [1, 2, 3, 4, 5]

a = range(1, 6)
print(a) # >> [1, 2, 3, 4, 5]

a = range(5)
print(a) # >> [0, 1, 2, 3, 4]
```

Array indexing

array_indexing.py

```
a = range(5)

print(a) # >> [0, 1, 2, 3, 4]
print(a[0]) # >> 0

print(a[len(a) - 1]) # >> 4
print(a[-1]) # >> 4

print(a[len(a) - 2]) # >> 3
print(a[-2]) # >> 3
```

C-Style for loop

c_style_for_loop.py

```python
a = range(5, 10)

for i in range(len(a)):
    print("a[{I}] = {AI}".format(I=i, AI=a[i]))
```

Python-Style for loop

python_style_for_loop.py

```python
a = range(5, 10)

for ai in a:
    print("AI = {AI}".format(AI=a[i]))

```

Python-Style for loop

python_style_for_loop.py

```python
a = range(5, 10)

for ai in a:
    print("AI = {AI}".format(AI=a[i]))

for i, ai in enumerate(a):
    print("a[{I}] = {AI}".format(I=i, AI=ai))
```

Exercise - Sum the Numbers from 1 to n

sum_nums.py

Exercise - Sum the Numbers from 1 to n

sum_nums.py

```
n = int(raw_input("n (-1 to quit) > "))
while n > 1:
    s = 0

    # n + 1 because range goes to n - 1
    for i in range(1, n+1):
         s += i

     print("n = {N} -> {S}".format(N=n, S=s))

    n = int(raw_input("n (-1 to quit) > "))
```

def

sum_nums_func.py

```python
def sum_nums(n):
    s = 0

    # n + 1 because range goes to n - 1
    for i in range(1, n+1):
        s += i

    return s

print("sum_nums({N}) => {SNN}".format(
    N=10,
    SNN=sum_nums(10)
))  # >> 55
```

Default Arguments

func_default_args.py

```
def default_args(a=1, b=3):
    print("a was {A}; b was {B}".format(A=a, B=b))

default_args()          # >> a was 1; b was 3
default_args(7)         # >> a was 7; b was 3
default_args(7, 11)     # >> a was 7; b was 11
default_args(7, b=12)   # >> a was 7; b was 12
default_args(a=8, b=12) # >> a was 8; b was 12
default_args(b=13, a=5) # >> a was 5; b was 13
default_args(a=10, 14)  # >> SyntaxError: non-keyword
    arg after keyword arg
```

Exercise - Sum the Numbers from low to high

sum_range.py

Exercise - Sum the Numbers from low to high

sum_range.py

```python
def sum_range(low=1, high=10):
    s = 0

    # high + 1 because range goes to high - 1
    for i in range(low, high + 1):
        s += i

    return s
```

Basics

basic_import_from_python_path.py

```python
# imports that math library and references it as math
import math
print(math.log(100, 10)) # >> 2.0 (log 100 base 10)

# sys.argv allows for command line arguments
import sys
print(sys.argv)
```

Different styles of importing

```python
# imports the sparse module from the scipy library as scipy.sparse
import scipy.sparse

# imports the sparse module from the scipy library as sp_sparse
import scipy.sparse as sp_sparse

# imports the sparse module from the scipy library as sparse
from scipy import sparse
```

From sum_range.py

import_sum_range.py

```python
# This is the only style that I will use in the workshop
import sum_range # Note that there is no .py
print(sum_range.sum_range(low=2, high=10)) # >> 54

import sum_range as sr
print(sr.sum_range(low=2, high=10)) # >> 54

from sum_range import sum_range
print(sum_range(low=2, high=10)) # >> 54

from sum_range import sum_range as sr
print(sr(low=2, high=10)) # >> 54
```

Definition, Instantiation, and Usage

animal.py

```python
# Explicit inheritance from object class
class Animal(object):
    # Explicit (and necessary) passing of self object
    def __init__(self, name_in, noise_in):
        self.name = name_in #  ≠  name = name_in
        self.noise = noise_in #  ≠  noise = noise_in

dog = Animal("Rex", "woof")
print("{DOG} makes a {NOISE} noise".format(
    DOG=dog.name,
    NOISE=dog.noise
))
```

Inheritance

animal.py

```python
class Dog(Animal):
    def __init__(self, name):
        # This gets better in Python 3
        super(Dog, self).__init__(name, "woof")

    # NEED to accept self
    def wag_tail(self):
        print("{NAME} is happy".format(NAME=self.name))

rex = Dog("Rex")
rex.wag_tail() # DON'T need to explicitly pass self
```