Introduction to Python

June Andrews

May 25, 2010

- Logistics
- Intro to Python
  - Basic Operators
  - Logicals
  - Types
  - Tuples, Lists, & Dictionaries
- Recursive Functions
  - 'and'
  - 'or'
  - Building Functions
- Style Guide
- Homework

Labs

From a non-lab computer visit:
http://www.csuglab.cornell.edu/userinfo
Running your own python setup, as well, is encouraged.

Book

- The book can be bought and shipped quickly for $40-$50.
- 2 copies will be on hold at Carpenter Library (1 in so far)
- It is on the order of recommended as in - required - but I can't get it into the bookshop quickly enough.
- You will use this book as long as you code in Python.
- We will cover highlights of Ch. 4 today. By the end of the semester you should know material from Ch.4, 5, 6, 7, 8, 9 off the top of your head, and have coding experience with Ch. 12, 13, 14, 15

Python has an interactive environment that allows you to try out assignments. It is very handy for learning and experimenting with everything we cover today. This environment is started by typing python (ipython if you have it) on unix systems(or Cygwin (or mac)).

Python in windows varies on your installation.

+, -, *, /	ordinary math operators
a**b	exponentiation, a^b
//	division with rounding towards $-\infty$
<<, >>, &, ^, |	bitwise operators (shifts, and, xor, or)
<, <=, >, >=, !=, ==	comparisons

or, and, not	boolean operators
in, not in	membership test
is, is not	identity test (point to the same thing)

'hello', “Hello“	strings
3, 4, 5	ints
3.14	floating points
3j	imaginary numbers
None	None type

Tuples:

1. Immutable
2. Indexed from 0
3. Created with (content1, content2, ...) or tuple(iterable),
4. Content can be any data type in Python, including tuples and mutable data.

Lists

1. Mutable
2. Indexed from 0
3. Very manipulable, with fast mutation implementation in Python
4. Will not use for now, but several built in functions.

Dictionaries

1. created with { key: value, key: value, ...} or dict(key=values, ...)
2. augmented with trial_dict[key] = value
3. dictionaries are not ordered

Basic Functions

Functional Programming is all about evaluating expressions.
So, let's rewrite the function length (works only on lists):

```
def Length(structure):
    if structure == []:
        return 0
    else:
        return 1 + Length(structure[1:])
```

Let's look at some expressions:

```
int(True)  -> 1
int(False)  -> 0
bool(7)  -> True
bool(True)  -> True
(0 == False) -> True
(True and 7) -> 7
(7 and True) -> True
(0 and False) -> 0
(False and 0) -> False
```

What's going on?
Python moves left to right, evaluating each term as a boolean and returns the first value that allows determination of the expression as False or True.

Python moves left to right, evaluating each term as a boolean and returns the first value that allows determination of the expression as False or True.

```
 (2 and 6 and 7) -> ?
 (True and max) -> ?
not ( min and max) -> ?
 (0 and False) -> ?
```

That was how Python evaluates the 'and' in an expression. Here's how Python evaluates 'or':

```
(True or 7) -> True
(7 or True) -> 7
(0 or False) -> False
(False or 0) -> 0
```

Python moves left to right, evaluating each term as a boolean and returns the first value that allows determination of the expression as False or True.

Can we build things up with this?
Normally:

```
if cond1:
    fun1(x)
else:
    fun2(x)
```

Equivalent:

```
(cond1 and fun1(x)) or (fun2(x))
```

Back to rewriting the function Length with expressions only.

```
def Length(structure):
    return (structure == [] and 0) or \
           (1 + Length(structure[1:]))
```

Problem: bool(([ ]==[ ] and 0)) = False
never stops.

Ex 1:

```
def Length(structure):
    return int(not(structure==[]) and \
                (1+FunctionalLength(structure[1:])))
```

Ex 2:

```
def Length(structure):
    return (int(structure == []) or \
              (2 + FunctionalLength(structure[1:]))) -1
```

Read: http://www.python.org/dev/peps/pep-0008/
All submitted code will follow these guidelines.

Homework tonight will be to read the style guide.
You will be given a module who's code does not obey the style guide.
You will be given an outline for a few simple recursive functions, fill them in.