Python's Odds and Ends

Luís Pedro Coelho

Programming for Scientists

January 22, 2009

Python So Far

Python

1. Basic types: int, float, list, dict
2. Control flow: for, while, if, else, elif
3. Type construction: class

List Indexing

```
students = ['Luis','Rita','Sabah','Grace']
print students[0]
print students[1:2]
print students[1:]
print students[-1]
print students[-2]
```

Tuples (I)

```
A = (0,1,2)
B = (1,)

print A[0]
print len(B)
```

Tuples (II)

Tuples are like immutable lists.

Set Type

```
numbers = set([1,2,5])
print 3 in numbers
numbers.add(4)
print numbers
numbers.add(1)
print numbers
print numbers | set(['Rita'])
print numbers - set([2,3])
```

Output:

```
False
set([1, 2, 4, 5])
set([1, 2, 4, 5])
set([1, 2, 4, 5, 'Rita'])
set([1, 4, 5])
```

Frozenset Type

```
numbers = frozenset([1,2,5])

print 3 in 5 # False
print 2 in 5 # True

numbers.add(1) # ERROR!!
```

What’s up With Immutability?

What’s up With Immutability?

You can only use immutable objects as dictionary keys!

Complex Numbers

```
A = 1+1j
print A**2
print A**4
```

prints

```
2j
(-4+0j)
```

None object

```
None
```

Object Identity

Object Identity

- `A is B`
- `A is not B`
- `id(obj)`

List Comprehensions

```
name = [ <expr> for <name> in <sequence> if <condition> ]
```

maps to

```
name = []
for <name> in <sequence>:
    if <condition>:
        name.append(<expr>)
```

List Comprehensions Example

```
squares = [x*x for x in xrange(1,20)]
evensquares = [x*x for x in xrange(1,20) if (x%2) == 0]

squares = []
for x in xrange(1,20):
    squares.append(x*x)

evensquares = []
for x in xrange(1,20):
    if (x%2) == 0:
        evensquares.append(x*x)
```

Functions

```
def max(arg0,*args):
    '''
    M = max(arg0,arg1,...)

    Returns the maximum of its arguments
    '''
    M = arg0
    for val in args:
        if val > M:
            M = val
    return M
```

```
def simulate(pop,max_iters,p_prob=.3,max_pop=None):
    '''
    Simulate a population of bacteria.

    Arguments
        * max_population: Maximum population
                          (default: 10*len(population))
        ...
    '''
    if max_population is None:
        max_population = 10*len(population)
    for i in xrange(max_iters):
        ...

population = [ ... ]
simulate(population,1000,.2)
simulate(population,max_iters=1000,p_prob=.2)
simulate(population,p_prob=.4,max_iters=1000)
simulate(population,1000,max_population=10**5)
```

Functions (III)

```
def f(arg0,arg1,*args,**kwargs):
    ...
```

Multiple Assignment

```
A, B = 1,2
```

Assign multiple elements at once.

Multiple Assignment to Return Multiple Arguments

```
def stats(values):
    '''...'''
    return mean(values), std(values)

...
values = ...
props = stats(values)
mu, std = stats(values)
```

```python
def greet(name,greeting='Hello'):
    '''
    greet(name,greeting='Hello')

    Greets person by name

    Arguments
    ---------
        * name: Name
        * greeting: Greeting to use
    '''
    print greeting, name

ret = greet('World')
```

Functions Are Objects

```
def integrate01(f):
    '''
    int_f = integrate01(f)
    ...
    '''
    res = 0.0
    for x in xrange(1000):
        res += f(x/1000.)/1000.
    return res

def identity(x):
    return x

def square(x):
    return x**2

integrate01(identity)
integrate01(square)
```

Sequences

```
for value in sequence:
    ...
```

Sequences

- Lists
- Tuples
- Sets & Frozensets
- Dictionaries
- ...

Generators

Generator: “Function”-like Sequence

```
def xrange(start,stop=None,step=None):
    '''
    xrange([start,]stop[,step]) -> xrange object

    Like range, but instead of a list, returns...
    '''
    if stop is None and step is None:
        stop = start
        start = 0
        step = 1
    elif step is None:
        step = 1

    while start < stop:
        yield start
        start += step
```

Generators

- Generators are similar to functions, but generate a sequence.
- Functions use `return`, generators use `yield`.

```python
def enumerate(iterable):
    '''...'''
    i = 0
    for val in iterable:
        yield i,val
        i += 1
```

Zip

```
names = ['Rita','Luis','Sabah']
grades = ['A','B','A']

for g,n in zip(names,grades):
    print 'Student %s had grade %s' % (g,n)
```

File Reading

```
for line in file('filename.txt'):
    print line
```

Modules & Libraries

```
import math

math.exp(1)
```

Namespaces

Namespaces are where names live.

bacteria.py

```python
...
def simulate(...):
    '''...'''
    ...
...
```

script.py

```python
import bacteria

population = [bacteria.Bacterium(...) ...]
bacteria.simulate(...)
```

Importing (II)

```
import bacteria
simulate = bacteria.simulate
Bacterium = bacteria.Bacterium
```

Importing (II)

```python
import bacteria
simulate = bacteria.simulate
Bacterium = bacteria.Bacterium

from bacteria import simulate, Bacterium
```

Importing (III)

```python
import bacteria
import bacteria
import bacteria
import bacteria
import bacteria
import bacteria
```

Import All

```
from bacteria import *
```

Import As

```
import bacteria
bac = bacteria

bac.simulate(...)
```

bac is another name for bacteria (modules are objects too!)

Import As

```
import bacteria
bac = bacteria

bac.simulate(...)
```

bac is another name for bacteria (modules are objects too!)

```
import bacteria as bac
```