A Little Python – Part 2

Introducing Programming with Python

Data Structures, Program Control

Outline

- Python and the System
- Data Structures
  - Lists, Dictionaries
- Control Flow
  - if, for, while

Reminder - Learning ANY Programming Language

- **Syntax**
  - What is a valid statement
  - What statements have meaning
- **Variables**, **data types**, data structure
- Control flow, branching, testing, loops, iteration
- Input/Output, I/O, read/write files
- **Procedures, subroutines**
- Objects, encapsulation of code + data

Reminder - Learning ANY Programming Language

- **Syntax**
  - What is a valid statement
  - What statements have meaning
- Variables, data types, **data structure**
- **Control flow, branching, testing, loops, iteration**
- Input/Output, I/O, read/write files
- Procedures, subroutines
- Objects, encapsulation of code + data

Python Path

- Python environment variable
  - Where to find python packages
- PYTHONPATH (on Mac or Linux)
  - cshell
    - setenv PYTHONPATH "/home/dwmc/development/python"
  - bash
    - PYTHONPATH="/home/dwmc/development/python"
    - export PYTHONPATH

Python Path

- PYTHONPATH (on Windows)
  - Google
    - “set pythonpath windows”
  - Sample
    - http://stackoverflow.com/questions/25153802/how-to-set-python-path-in-windows-7

Python Modules

- System or Distribution Modules
  - System modules - come pre-installed
  - Distribution modules - installed with `easy_install` or `pip`
- User modules (your own code)
  - Python searches the PYTHONPATH directories
    - In the order specified
  - User Module *name* same as a directory *name*
    - Directory must have an `__init__.py` file

Import Python Modules

- Access a module with the "import" command
  - Saw an example of this last time
  - Import

```
>>> import math
>>> import random
>>> import aflac
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named aflac
```

- Importing user modules is the same

Import Variant

- Import all package variables, methods

```
import random
print random.randint(3,100)
```

- Import using a "from" clause (import parts of a module)

```
from random import randint
print randint(3,100)
```

- Note: there are different ways to use "from" clause

Some Important Modules

`import sys`	Access to system features
`import os`	Python operating system hooks
`import urllib`	Manipulate URLs, web access
`import random`	Random number generator

Data Structures

- Lists
- Dictionaries
- Tuples (not covering these, fixed list)

Data Structures - Lists

- List an unordered collection of arbitrary items

```
l = []
l = [1, 'two', 3, 4]
l = ['a', ['b', 'c']]
l[i]
l[j][k]
l[x:y]
l.append('abc'), l.sort(), l.pop(), l.remove(3)
del l[k]
l[2] = 'abc'
```

- Try a few of these

Example - lists

```
>>> l1 = []
>>> l2 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
>>> print l1
[]
>>> print l2
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
>>> print l2[2]
c
>>> print l2[4:7]
['e', 'f', 'g']
>>> print l2[:3]
['a', 'b', 'c']
>>> print l2[7:]
['h', 'i', 'j']
```

Example - lists

```
>>> l1 = []
>>> l2 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
>>> print l1
[]
>>> print l2
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
>>> del l2[5]
>>> print l2
['a', 'b', 'c', 'd', 'e', 'g', 'h', 'i', 'j']
>>> print l2.pop()
j
>>> print l2
['a', 'b', 'c', 'd', 'e', 'g', 'h', 'i']
>>> l2.append(45)
>>> print l2
['a', 'b', 'c', 'd', 'e', 'g', 'h', 'i', 45]
>>> l2.append(l1)
>>> print l2
['a', 'b', 'c', 'd', 'e', 'g', 'h', 'i', 45, []]
>>> print len(l2)
10
```

Data Structures - Dictionaries

- Key/Value stores, arbitrary items
  - Keys are always strings
  - Keys are immutable (fixed), but values can be changed

```
d = {}
d = {'size':4, 'name':"bob", 'list':[1,2,3], 'dict':{}}
d['size']
d['list'][0]
d.keys()
d.values()
del d['name']
```

- Try a few of these

Example - dictionaries

```
>>> d2 = {'size':4, 'name':"bob", 'list':[1,2,3,'a','b'], 'dict':
{'name':"booboo", 'value':123}}
>>> print d2
{'dict': {'name': 'booboo', 'value': 123}, 'list': [1, 2, 3, 'a', 'b'], 'name':
'bob', 'size': 4}
>>> print d2['size']
4
>>> print d2['list'][0]
1
>>> print d2['dict']['name']
booboo
>>> print d2.keys()
['dict', 'list', 'name', 'size']
>>> print d2.values()
[{'name': 'booboo', 'value': 123}, [1, 2, 3, 'a', 'b'], 'bob', 4]
>>> d2['gob']="This is a string."
>>> print d2
{'dict': {'name': 'booboo', 'value': 123}, 'gob': 'This is a string.', 'list':
[1, 2, 3, 'a', 'b'], 'name': 'bob', 'size': 4}
>>> print len(d2)
5
```

Control Flow

- Conditional Tests
  - if
- Looping
  - while
  - for

Conditional - if

- Conditional Branch

```
if <test_condition>:
   statements (block)
elif <test_condition>:
   statements (block)
else:
   statements (block)
```

Example - if

```
def iftest(goo):
            if goo == 1:
                        print "Found a digit"
            elif goo=="one":
                        print "Found one string"
            else:
                        print "Not sure"

>>> goop = "one"
>>> iftest(goop)
Found one string
>>> iftest(1)
Found a digit
>>> iftest("two")
Not sure
>>> iftest(one)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'one' is not defined
```

Looping - while

- Conditional Loop

```
while <test_condition>:
    statements (block)
```

- Special statements

  `break`

  - Exit the loop

  `continue`

  - Restart loop from this continue statement, back to top

Example - while

```
def loopTest1(c=0,b=100):
    i = 0
    while i<c:
          print i
          if i==b:
               print "break"
               break
          i += 1

>>> loopTest(5)
0
1
2
3
4
>>> loopTest(5,2)
0
1
2
break
```

Looping - for

- Iterator looping

```
for <item> in <iterable_object>:
    statement (block)
```

- Special looping statements

  `break`

  `continue`

- Particularly good for iterating through lists

Example - for

```
l2 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']

def loopTest2(l=[]):
    if l:
        for item in l:
            print "Got item: \"%s\""%(str(item))
    else:
        print "List parameter was empty"
```

Example - for

```
>>> print l2
['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j']
>>> loopTest2()
List parameter was empty
>>> loopTest2(l2)
Got item: "a"
Got item: "b"
Got item: "c"
Got item: "d"
Got item: "e"
Got item: "f"
Got item: "g"
Got item: "h"
Got item: "i"
Got item: "j"
```

Assignment 2

- Write 4 short programs

1. Write a procedure that accepts one parameter (count) and generates a list of (count) random integers, between 0 and 1000, and puts those integers into a list, and returns the list.

2. Write a procedure called "no_5xx" which searches a list (like the one from #1 above) and removes any integer value in the range 500 to 599, and returns the resulting list.

3. Write a procedure that takes four parameters (lastname, firstname, score, grade) and returns a new dictionary item with those four items.

4. Write a procedure called "update_lastname" that takes two parameters (a dictionary, like from #3 above) and a string value and updates the value for the "lastname" key in the dictionary.

In class Activity

- Do activity – 15 minutes