A Little Python – Part 1

Introducing Programming with Python

Some Basics

Preface

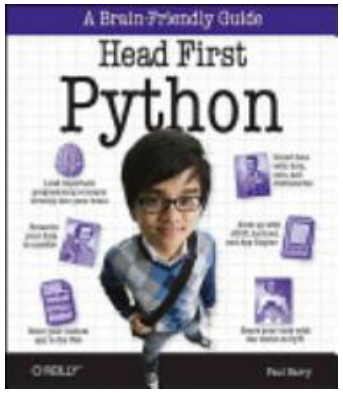
- Not a complete course in a programming language
  - Many details can't be covered
  - Need to learn as you go
- My programming 'style' is not considered 'pythonic'
  - I program like a very careful C programmer
  - "Python" people generally hate my code
    - I'll do my best to give examples in good style here

Learning a New Programming Language

- Syntax – what statements have meaning
- Variables, types, structure
- Control flow, loops, branching, testing
- I/O, read/write files
- Procedures, subroutines

Additional Resources

- Online tutorial – very good (for some people)
  - http://docs.python.org/tutorial/index.html
- Learning Python book ~ $55 ($30)
  - Great if you know another language
  - New version
- Head First Python ~ $60 ($30)
  - Good for real beginners



Python

- Interpreted
- Interactive

```
[dwmc-mbpro:~] dwmc% python
Python 2.6.1 (r261:67515, Jun 24 2010, 21:47:49)
[GCC 4.2.1 (Apple Inc. build 5646)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Just “try it” in the interpreter
  - If it works in the interactive interpreter – then it works

Example

- Launch python
- Assign some variables
- Print variable values

```
>>> def counter(i):
...     k = 0
...     while( k<i ):
...             print k
...             k += 1
...
>>> counter(5)
```

Basic Data Types

- Empty value - `None`
- Boolean
- Strings
- Numbers
  - Integer values
  - Reals/Floating point values

Basic Data Types

- An empty value
  - Special marker, `None`
    - Not the same as `""` or `[]` or `{}`
- Two values
  - `True`
  - `False`

Basic Data Types - Strings

- Assigning string values

```
>>> foo = "this is a string"
>>> bar = 'this is also a string'
>>> special = '''python allows
... multi line
... string contants'''
>>>
```

- Unicode strings
  - Important for web work
  - The ‘u’ designation makes a string unicode
  - More on this later

Basic Data Types - Integers

- Assigning integer values

```
>>> n = 123
>>> k = 111
>>> i = 1
>>> print n+i
124
>>> print n+k
234
>>> print k+i
112
>>>
```

- Integers support arbitrary & dynamic size

Basic Data Types - Float

- Assigning real/floating point values

```
>>> n = 123
>>> k = 1.11
>>> i = 1.3
>>> print n+k
124.11
>>> print n+i
124.3
>>> print i+k
2.41
>>>
```

- Float support arbitrary & dynamic size, automatic conversion

Operations

- Operations are the way you change variable values, compare, or manipulate values

```
x = y
x or y
x and y
not x
x + y, x - y, -x
x * y, x / y
(), [], {}
x==y, x<y, x>y, x<=y, x>=y, x!=y
x in y, x not in y, x is y, x is not y
```

- Try a few of these

Examples – Some operators

```
>>> n = 124
>>> m = 2
>>> k = 10.5
>>> i = 1.3
>>> n / m
62
>>> n * m
248
>>> n / i
95.384615384615387
>>> n // i
95.0
>>> (i==k)
False
>>> (i<=k)
True
>>> (i!=k)
True
```

Examples – Some operators

```
>>> foo = "this is a string"
>>> bar = "this is a string"
>>> foo+bar
'this is a stringthis is a string'
>>> foo-bar
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'str'
>>> foo*bar
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can't multiply sequence by non-int of type 'str'
>>> foo in bar
True
>>> foo is bar
False
>>> bar in foo
True
```

Sidebar – import statement

- Mechanism to extend what language can do
  - Things that don't get used all the time
- Special math functions, random numbers, operating system specific features, etc.

```
>>> import math
>>> import random
>>> import aflac
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named aflac
>>> random.random()
0.14584529741440777
>>> random.randint(5,151)
30
```

Operators - Strings

- Strings have additional operators

```
s[i]
s[i:j]
len(s)
s.find('is')
s.replace('is','as')
s.split('-')
s.isdigit()
s.lower()
```

- Try a few of these

Example – String operators

```
>>> s = "this is a string"
>>> s2 ="a-string-with-no-spaces"
>>> s3 = "30"
>>> s[5]
'i'
>>> s[5:6]
'i'
>>> s[5:7]
'is'
>>> s.find('is')
2
>>> s.replace('is','as')
'thas as a string'
>>> s2.split('-')
['a', 'string', 'with', 'no', 'spaces']
>>> s3.isdigit()
True
>>>
```

Generating Output

- Simple output
- print statement

```
>>> s1 = "the value of k is"
>>> k = 1.45
>>> print s1,k
the value of k is 1.45
>>> print k,s1
1.45 the value of k is
>>> s2 = "bob's big boy"
>>> print s1,s2
the value of k is bob's big boy
>>> print s1+s2
the value of k isbob's big boy
>>>
```

Program Structure

- Subroutines, procedures, methods

```
def loopCounter(i):
          k = 0
          while( k<i ):
                    print k
                    k += 1
```

Simple Bob Program

```
def bob():
        print “Bob is great!”

def notBob():
        print “Bob is a fink!”

def liveBob():
        print “Long live Bob!”

def allBob():
        bob()
        notBob()
        liveBob()
```

Bob Program Output

```
>>> bob()
Bob is great!
>>> notBob()
Bob is a fink!
>>> liveBob()
Long live Bob!
>>> BOB()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'BOB' is not defined
>>> allBob()
Bob is great!
Bob is a fink!
Long live Bob!
>>>
```

Python Programming Environments

- IDLE (sucks)
- Python Additions to Eclipse
  - PyDev
- Komodo Edit (nice)
  - Komodo IDE (expensive)
- Plain old text editor
  - Old school

Assignment

- Write 4 short programs
  - They are a few lines each

1. Calculate and print 13! (13 factorial)
2. Output Happy New Year! using 3 different strings
3. Write three procedures that each write one of Happy, New, and Year!
4. Write a procedure that uses the procedures in 3 to print Happy New Year!