Reading Assignment

Aug 31. Chapters 1-4 of the Python Tutorial
Sept 5. Chapters 5-8 of the Python Tutorial
Sept 7. Chapters 9-11 of the Python Tutorial

Quiz & Assignment

First Quiz - Sept 12
Assignment 1 due - Sept 12
References


Learning Python, Lutz & Ascher, O'Reilly, 1999
The Emerging Web
Three Stories
Bows verses Rifles
Listening to Piano
Shaking Heads
Python

The FIRST 20 YEARS OF MONTY PYTHON

KIM "HOWARD" JOHNSON
Running Python

Interpreter
Script file
Compiled code
Application

AI 11->python
Python 2.3.5 (#1, Mar 20 2005, 20:38:20)
Type "help" for more information.

>>> 1 + 2
3

>>>
Script File

Unix Example
File name: helloWorld.py

`#!/usr/bin/env python`

`print 'Hello World'`

Make the file executable:

`Al 45->chmod u+x helloWorld.py`

Run the file

`Al 46->helloWorld.py`
Running on Windows

See Python Windows FAQ
http://www.python.org/doc/faq/windows.html
Python on Rohan

On rohan python is located at

/opt/local/bin/python

/opt/local/bin is on the default path for accounts

So the following command should start the python interpreter

python
Python Data Types
"Static types give me the same feeling of safety as the announcement that my seat cushion can be used as a floatation device."

Don Roberts
# Numbers

<table>
<thead>
<tr>
<th>Literals</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>123, -34, 0</td>
<td>Integer (C long)</td>
</tr>
<tr>
<td>2323232323232L</td>
<td>Long Integer (unlimited size)</td>
</tr>
<tr>
<td>1.23, 3.14e-10, 4E21, 5.3e+43</td>
<td>Floating-point (C double)</td>
</tr>
<tr>
<td>052, 0x4da</td>
<td>Octal &amp; Hex</td>
</tr>
<tr>
<td>2+5j, 7.2-8j, 2j</td>
<td>Complex number</td>
</tr>
</tbody>
</table>

```python
>>> a = 5
>>> b = 2
>>> a * b
10
>>> a / 3
1
>>> a / 3.0
1.6666666666666667
>>> c = 3 + 2j
>>> a + c
(8+2j)
>>> abs(-5)
5
```

[Operations](http://docs.python.org/lib/module-math.html)
[Types](http://docs.python.org/lib/typesnumeric.html)
Variable Names

Case sensitive
Must start with underscore or letter
May contain letters, digits or underscores cat_3
Can not use reserved words
## Reserved Words

<table>
<thead>
<tr>
<th>and</th>
<th>del</th>
<th>for</th>
<th>is</th>
<th>raise</th>
</tr>
</thead>
<tbody>
<tr>
<td>assert</td>
<td>elif</td>
<td>from</td>
<td>lambda</td>
<td>return</td>
</tr>
<tr>
<td>break</td>
<td>else</td>
<td>global</td>
<td>not</td>
<td>try</td>
</tr>
<tr>
<td>class</td>
<td>except</td>
<td>if</td>
<td>or</td>
<td>while</td>
</tr>
<tr>
<td>continue</td>
<td>exec</td>
<td>import</td>
<td>pass</td>
<td>yield</td>
</tr>
<tr>
<td>def</td>
<td>finally</td>
<td>in</td>
<td>print</td>
<td></td>
</tr>
</tbody>
</table>
## Operator Precedence

<table>
<thead>
<tr>
<th>Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x or y</td>
<td>Lazy logical or</td>
</tr>
<tr>
<td>x and y</td>
<td>Lazy logical and</td>
</tr>
<tr>
<td>not x</td>
<td>Logical negation</td>
</tr>
<tr>
<td>&lt;, &lt;=, &gt;, &gt;=, ==, &lt;&gt;, !&gt;, is, is not, in, not in</td>
<td>Comparison, identity tests, sequence membership</td>
</tr>
<tr>
<td>X</td>
<td>y</td>
</tr>
<tr>
<td>x ^ y</td>
<td>Bitwise exclusive or</td>
</tr>
<tr>
<td>x &amp; y</td>
<td>Bitwise and</td>
</tr>
<tr>
<td>x&lt;&lt;y, x&gt;&gt;y</td>
<td>Shift x left(right) by y bits</td>
</tr>
<tr>
<td>x+y, x-y</td>
<td>Addition/Concatenation, subtraction</td>
</tr>
<tr>
<td>x*y, x/y, x%y</td>
<td>Mult/repetition, division, remainder/format</td>
</tr>
<tr>
<td>-x, +x, ~x</td>
<td>Negation, identity, bitwise complement</td>
</tr>
<tr>
<td>x[k], x[i:j], x.y, x(...)</td>
<td>Indexing, slicing, qualification, functions call</td>
</tr>
<tr>
<td>(...)</td>
<td>Tuple, list, dictionary, conversion to string</td>
</tr>
</tbody>
</table>
Strings

Immutable - once create can not change

aString = 'Cat in the hat'  # a Comment
bString = "Cat in the hat"
multiLineString = """"Cat in
the
hat"""

>>> a = 'cat'
>>> a[0]
'c'
>>> a[1]
'a'
>>> len(a)
3
>>> a[-1]  # a[len(a) - 1] that is index from end
't'

>>> 'a' in a
True
>>> 'a' not in b
True

>>> a + b
'catdog'

>>> a * 3
'catcatcat'

>>> min(a)  # min element in a
'a'
>>> max(b)
'o'
Slicing

```python
>>> six = '012345'
>>> len(six)
6
>>> six[2:4]   #from index 2 up to index 4
'23'
>>> six[:4]    #from start up to index 4
'0123'
>>> six[3:]    #from index 3 to end
'345'
>>> six[:]     #Make a copy - from start to end
'012345'
>>> six[1:5:2]  #from index 1 up to 5 step 2
'13'
```

More on slicing
http://docs.python.org/ref/slicings.html
like sprintf in C

```python
>>> template = 'We have %d too many %s'

>>> template % (10, 'crashers')
'We have 10 too many crashers'

>>> template % (5, 'hurricanes')
'We have 5 too many hurricanes'
```

Formatting Details
String Methods

```python
>>> string = 'this is a example'
>>> string.capitalize()
'This is an example'

>>> string.split()
['this', 'is', 'an', 'example']

>>> string.split('i')
['th', 's ', 's an example']

>>> string.split('ia')
['this is an example']

>>> string.split('an')
['this is ', ' example']
```

All string methods
Iterating

for variableName in aString:

>>> a = 'c1t2'
>>> for c in a:
...     if c.isdigit(): print c
...
1
2
Compiler requires Indentation

Compile Error

```python
a = 'c1t2'
for c in a:
    if c.isdigit(): print c
```
Mixing tabs & spaces confuses the compiler
Collections

Sequences
- Strings
- Lists [ 5, 8 ]
- Tuples ( 5, 8 )
- Unicode Strings
- Buffer
- Xrange objects

Dictionary
{"cat": 'बिल्ली', 'dog': 'कुत्ता' }

Sequence Operations
_lists
mutable ordered collection

```python
>>> aList = ['cat', 'dog', 3, 'mouse']
>>> aList[2]
3
>>> aList[1:3]
['dog', 3]

>>> aList.sort()
>>> aList
[3, 'cat', 'dog', 'mouse']

>>> aList[0] = 'tea'
>>> aList
['tea', 'cat', 'dog', 'mouse']

>>> aList.append('trap')
>>> aList
['tea', 'cat', 'dog', 'mouse', 'trap']

>>> del aList[1]
>>> aList
['tea', 'dog', 'mouse', 'trap']

>>> del aList[1:3]
>>> aList
['tea', 'trap']
```
>>> a = ['cat', 'dog']
>>> b = [1, a, 2]
>>> b
[1, ['cat', 'dog'], 2]

>>> len(b)
3
>>> b[1]
['cat', 'dog']

>>> b[1][0]
'cat'

>>> b[1][0] = 'new'
>>> b
[1, ['new', 'dog'], 2]
>>> a
['new', 'dog']
Tuple

```python
>>> aTuple = ('cat', 'dog', 'mouse')
('cat', 'dog', 'mouse')

>>> aTuple[2]
'mouse'

>>> for item in aTuple:
...     print item
...     print item
...     print item
...     print item
...

cat
dog
mouse

>>> aTuple[1:2]
('dog',)

>>> tupleAlso = 1, 2, 3
>>> tupleAlso
(1, 2, 3)

>>> emptyTuple = ()
>>> emptyTuple
()```

Immutable ordered collection
Mutable list
Dictionary

```python
>>> offices = { 'whitney': 'GMCS 561', 'beck':'GMCS-407B' }
>>> offices['whitney']
'GMCS 561'

>>> offices.keys()
['beck', 'whitney']
>>> offices.values()
['GMCS-407B', 'GMCS 561']

>>> offices.has_key('lewis')
False

>>> offices['lewis'] = 'GMCS-544'

>>> offices.has_key('lewis')
True
>>> offices.has_key('Lewis')
False

>>> del offices['whitney']
>>> offices.keys()
['beck', 'lewis']
```

Mutable map (or hashtable)
Keys must be immutable

Details
http://docs.python.org/lib/typesmapping.html
Use a "\"" to continue code on the next line

```python
if 1900 < year < 2100 and 1 <= month <= 12 \
    and 1 <= day <= 31 and 0 <= hour < 24 \
    and 0 <= minute < 60 and 0 <= second < 60:  #Valid date
    return 1
```

Not needed inside [], (), and {}

```python
month_names = ['Januari', 'Februari', 'Maart',
               'April', 'Mei', 'Juni',
               'Juli', 'Augustus', 'September',
               'Oktober', 'November', 'December']
```
Files

File modes: 'r', 'w', 'a' (append)

File modes: 'r', 'w', 'a' (append)

Details
http://docs.python.org/lib/bltin-file-objects.html
Equality

a is b
Do a & b point to same object

a == b
Do a & b have point to objects that have same structure and value

```python
>>> a = [1, ('cat', 3)]
>>> b = [1, ('cat', 3)]
>>> c = a

>>> a == b, a is b, a == c, a is c
(True, False, True, True)
```
Comparing

```python
>>> a = [1, ('cat', 3)]
>>> d = [1, ('cat', 4)]
>>> a < d, a == d, a > d
(True, False, False)

>>> {1:1, 2:2} < {1:2, 2:1}, {1:1, 2:2} > {1:2, 2:1}
(True, False)

23 < 'cat' , 23 == 'cat', 23 > 'cat'
(True, False, False)

>>> 23 < [1, 'cat']
True

>>> ['cat'] < ('cat'), ['cat'] == ('cat'), ['cat'] > ('cat')
(True, False, False)
```
Comparing

Numbers are compared as number

Strings are compared lexicographically

Lists & tuples are compared by comparing each component, from left to right

Dictionaries are compared as sorted (key, value) lists

Can comparing different types, but results can be meaningless
Boolean

True
False

However other values can be evaluated as boolean

```python
>>> if 'cat': print 'hi'
hi
```

Values equivalent to False

None
Numeric zero of all types
Empty sequences: " [] ()
Empty mappings: {}

All other values are interpreted as True