Basic Python 2.7 Syntax

Gina Schmalzle & Caroline Harbitz

Python interactive shell

```
Start Python:
      $ python
Exit Python:
      >>> exit()
```

Reserved words

```
and, as, assert, break, class,
continue, def, del, elif, else,
except, exec, finally, for, from,
global, if, import, in, is, lambda,
not, or, pass, print, raise, return,
try, while, with, yield
```

Strings

```
a = 'hellothere'
Get 1st element:
      a[0]
Get 'hello':
      a[0:5]
Reverse:
      a[::-1]  # 'olleh'
Concatenate two strings:
      a + 'you'

Some useful methods:
.strip(), .split(), .find(),
.uppercase(), .lowercase()
```

Lists

```
my_list = ['one', 'two', 'three']
Get 1st element:
      my_list[0]
Append another string:
      my_list.append('four')
Append an integer:
      my_list.append(5)
Remove 'one':
      my_list.remove('one')
Sort the list:
      my_list.sort()
```

Dictionaries

```
animals = {'c': 'cat'}
Get a value from a key:
    animals['c']  # 'cat'
Delete a key:
    del animals['c']
Is a key in the dictionary?
    if 'd' not in g:
      animals['d'] = 'dog'
Update a key:
    g['c'] = 'cayman'
Combine two dictionaries:
    animals.update({'e': 'emu'})

Some useful methods:
.items(), .keys(), .values()
```

Built-In Data Types

```
a = 5  # integer
b = 5.0  # float
c = 'hello'  # string
d, e = True, False  # Boolean
f = ['hello', 'world']  # list
g = {'key1': 'value1', 'key2':
      'value2'}  # dictionary
h = (5,3)  # tuple
i = None # Python's Null/undefined
```

Operations on type bool

- **a** and **b** is **True** if **a** and **b** are **True**; **False** otherwise
- **a** or **b** is **True** if at least one of **a** or **b** is **True**; **False** otherwise
- **not a** is **True** if **a** is **False**, and **False** if **a** is **True**

Control flow statements

```
break, pass, continue, return,
try/except
```

Iteration

for

```
my_list = ['one', 'two', 'four']
for item in my_list:
      print item

for idx,item in enumerate(my_list):
      print 'Item index: ', idx
      print 'Item name: ', item
```

while

```
count = 0
while count < 10:
    count += 1
    print count
```

Branching

```
numbers = [1,2,3,4,5,6]
for n in numbers:
  if n % 2 == 0 and n % 3 == 0:
     print n, ' is divisible by 2 &
       3!'
  elif n % 2 == 0:
    print n, ' is even!'
  else:
     print n, ' is odd'
```

Other useful functions

- dir() # list object attributes
- help()
- len()
- str()
- int()
- sort() # for lists
- range()
- seq()

Reading a file

```
import csv

filename = '/location/myfile.csv'
with open(filename, 'rb') as csvfile:
   reader = csv.reader(csvfile,
            delimiter=',')
   headers = next(reader)
   for row in reader:
       var1, var2, var3 = row
```

Reading a file this way handles file opening/closing.

Writing to a file

```
sample = [('the answer', 42),
          ('question', 24)]
myfile = 'output.csv'
header = ['text', 'number']
```

Method 1

```
import csv
with open(myfile, 'w') as csvfile:
   mywriter = csv.writer(csvfile,
              delimiter=',')
   mywriter.writerow(header)
   mywriter.writerows(sample)
```

Comparison operators

== (equal)
!= (not equal)
> (greater)
>= (at least)
< (less)
<= (at most)

Operators on type int and float

Addition:
The sum of **i** and **j** is **i+j**. If **i** and **j** are both of type **int**, the result is an **int**. If either are type **float**, the result is a **float**.

Subtraction:
i-j. Same type rule as for addition applies.

Multiplication:
i*j. Same type rule as for addition applies.

Integer division:
i//j. For example, the result of 6//2 is the integer 3. The value of 6 // 4 is the integer 1. Integer division returns only the quotient, not the remainder.

"Normal" division:
i/j. Python2.7: the result is an integer if both **i** and **j** are integers. If one is a **float**, the result is a **float**. Python3: Always returns a **float**.

Modulo:
i%j is the remainder when the integer **i** is divided by the integer **j**.

Exponents:
ij** is **i** raised to the power **j**. Same type rule as for addition applies.

References

- Guttag, John V.: Introduction to Computation and Programming Using Python. MIT Press, 2013.
- Harrison, Matt: Treading on Python, vol 1: Foundations. CreateSpace, 2013.