Python Quick Reference

General

```
# indicates a comment

Blocks are indicated by indentation
(spaces versus tabs matter) –
consistently use four spaces for
indentation.
```

Strings

strip()

```
Strip off leading/trailing whitespace
>>> ip = ' 192.168.1.1   '
>>> new_ip = ip.strip()
>>> new_ip
'192.168.1.1'

lstrip/rstrip for leading/trailing strip
>>> ip = ' 192.168.1.1\n'
Strip trailing newlines
>>> ip.rstrip('\n')
' 192.168.1.1'
```

split()

```
Split a string into a list using
delimiter.  Default delimiter is
consecutive whitespace.
>>> ip = '192.168.1.1'
>>> octets = ip.split('.')
>>> octets
['192', '168', '1', '1']
```

join()

```
Join a list using a separator.
>>> octets = ['192', '168', '1', '1']
>>> ip = '.'.join(octets)
>>> ip
'192.168.1.1'
```

count()

```
Count of substring in string
>>> ip = '192.168.1.1'
>>> ip.count('.')
3
```

substring in string

```
>>> ip = '192.168.1.1'
>>> '192.168.' in ip
True
>>> '10' in ip
False
```

concatenation

```
>>> a = '192.168.1.'
>>> b = '10'
>>> a + b
'192.168.1.10'
```

Lists

```
octets = [192, 168, 1, 1]
```

append()

```
Add an entry to the end of a list
>>> ip = [192, 168, 1]
>>> ip.append(10)
>>> ip
[192, 168, 1, 10]
```

len(list)

```
Determine the length of a list
>>> ip = [192, 168, 1, 10]
>>> len(ip)
4
```

pop()

```
Remove an element from a list (by
default it is the last element).
>>> ip = [192, 168, 1, 10]
>>> last_octet = ip.pop()
>>> last_octet
10
>>> ip
[192, 168, 1]
Pop the first element off the list
>>> ip.pop(0)
192
>>> ip
[168, 1]
```

slices

```
Construct a new list from a section
of an existing list.  First number is
starting index (included in slice);
last number is end index
(excluded from slice)
>>> ip[0:3]
[192, 168, 1]
Default first index is 0
>>> ip[:3]
[192, 168, 1]
Default last index is end of list.
>>> ip[0:]
[192, 168, 1, 10]
>>> ip[:]
[192, 168, 1, 10]
Does not modify the original list.
>>> ip
[192, 168, 1, 10]
```

concatenation

```
>>> a = [192, 168, 1]
>>> b = [10]
>>> a + b
[192, 168, 1, 10]
```

Lists (continued)

del

```
delete index specified
>>> ip
[192, 168, 1, 10]
>>> del ip[3]
>>> ip
[192, 168, 1]
```

remove()

```
remove the first occurrence of value
>>> ip = [192,1,21,1]
>>> ip.remove(1)
>>> ip
[192, 21, 1]
```

insert()

```
insert an element at the index specified
>>> ip
[192, 21, 1]
>>> ip.insert(0, 192)
>>> ip
[192, 192, 21, 1]
```

sort()

```
>>> a = [7,3,10,2]
>>> a.sort()
>>> a
[2, 3, 7, 10]
```

Functions

```
def a_function1(var1, var2):
    print var1, var2
    return None

def a_function2(a, b=10):
    return a+b
```

calling functions

```
Positional parameters
c = a_function1('test', 'test2')
d = a_function2(20)
d = a_function2(5,7)
Named parameters
d = a_function2(b=5,a=7)
```

arithmetic operators

```
+ - / * %
```

boolean operators

```
not and or
```

comparison operators

```
==      equals
!=      not equals
<, <=, >, >=
```

Files

```
f = open('temp.file')
f.close()

# Go to the beginning of the file.
f.seek(0)

# iterate over file one line at a time.
for line in f:
    print line

# Returns a list one entry per line in
# the file.
f.readlines()

# Read one line from the file
f.readline()

# Open a file for writing
f = open('new_file', 'w')

# write a line to file
f.write('test\n')

# flush buffer
f.flush()

# open file for append
f.open('existing_file', 'a')

# open a file using with
with open('temp.file') as f:
```

Flow control

```
if expression:
    BLOCK
elif expression:
    BLOCK
else:
    BLOCK

while expression:
    do_something()
    break or set expression to False

for value in [a, b, c, d]:
    print value
    continue    # go to next value
    break       # break out of for

for i in range(10):
    print i
prints 0, 1, ... 9
```

Dictionaries

```
a_device = {'ip':'192.168.1.1',
'username':'testuser', 'password':'fake'}

>>> a_device['ip']
'192.168.1.1'
>>> a_device['ip'] = '172.16.31.11'
```

iterating over keys

```
for key in a_device.keys():
    print key
```

iterating over items

```
for k,v in a_device.items():
    print k,v
```

del

```
delete the key specified
>>> del a_device['username']
```

get()

```
An alternate way of retrieving values from
dict that avoids KeyError.  Instead of
KeyError returns None (by default)
>>> a_device.get('ip')
'172.16.31.11'
>>> a_device['device_type']
...
KeyError: 'device_type'
>>> a_device.get('device_type')
```

field in dict

```
>>> a_device
{'username': 'testuser', 'ip': '172.16.31.11',
'password': 'fake'}
>>> 'username' in a_device
True
>>> 'device_type' in a_device
False
```

Miscellaneous

raw_input

```
>>> a = raw_input("Test: ")
Test: whatever
>>> print a
whatever
```

print

```
print 'test %s %s' % (var1, var2)
```