Python Basics

Sequences

List []
Tupple ()
Dic {} is not a sequence

Looping

```
for key, value in a_dictionary:
    print key, value

for value in a_list:
    print value
To print the positon, use enumerate like below:
for idx, value in enumerate(a_list):
    print idx,value
in reverse order:
for value in reverse(a_list):
    print value
```

To iterate an object, the object's class must have implemented the next() and __iter__() methods.

List comprehension

List comprehension is a way to make a list from other lists.

```
list1=[1,2,3,4,5]
list2=[x*2 for x in list1 if x%2==0]
```

Exception Handling

```
#1
try:

except Exception:
    raise
else:

finally:

#2
try:
```

```
    try:
        print '2nd try'
        raise Exception('throws 1st exception')
    finally:
        print 'finally'
    print '1st try'
    raise Exception('throws 2nd exception')
except Exception, e:
    print e
```

To create your own exceptions, write a class that extends from *Exception*.

Module

Modules are always read from top to bottom, whether executed directly or imported. Top-level attributes of the imported module are name spaced to that module. Modules are searched via the directories specified on the *sys.path* list. Modules cannot be unloaded, and they can't easily be re-loaded. While a *reload()* method exists, it is typically used in debugging or developments and not in production mode. The reason is obvious, since reload a module will mess up the sequences and modules dependencies. Try to avoid circular references of modules.

The *sys.modules* provides a listing of all of the modules that have been loaded by the interpreter.

The *__buildin__* module contains references to all build-in identifiers. This module is rarely imported, however, it could become necessary if you have code that overwrites any of the global identifiers.

The *__future__* module defines features in versions of Python that would eventually be incorporated.