CHEATSHEET PYTHON 5 TESTING & CODE HYGIENE

Good coding practices:

- Modular coding: make each function do one thing and do it well. It's easier to compose and test minimal, single-purpose functions.
- Test a function/line of code before you write another one!
- Comment a lot and add doc strings once you have a functioning function. Future you will be so happy.
- Use informative names for your variables. Function names should be verbs while instances should be nouns or noun phrases. In general, avoid short abbreviations. Single letters (i, n, ...) are typically reserved for counting variables.
- Once you're a seasoned python-er, keep your style up to standard by reading 'best practices' (see Lesson)

Ways to test your code:

1. If you're working with large files, use a reduced file to test your code, or only test one file at first.

```
head -1000 largeDataFile.csv > test.csv         OR        function_name('file1.txt')
```

2. As you're writing functions, use print to check that your output is what you expect.

```
import os
def make_filelist(directory, file_ending):
    """This function makes a list of files with a certain ending in a certain directory."""
    files=os.listdir(directory)
    files_pdb=[]
    for file in files:
        if file.endswith(file_ending):
            files_pdb.append(file)
    print files
    print files_pdb
    return files_pdb
make_filelist('.', '.pdb')
```

3. Python's assert clause allows you to test a comparison and exits the script if not true. You can have it print an error message upon exiting.

```
assert a == b, "Error: comparison %s == %s is false" %(value1,value2)

def sum_num(num_list):
    num_list=[3,52,6,'b',2,463,'a']
    totalSum=0
    for item in num_list:
        assert type(item)==int, "Uh oh, item '%s' not the right type! Exiting now." %item
        totalSum+=item
    print totalSum

sum_num(num_list)
```

4. Python's try-except clauses allow you to trigger error messages for specific types of errors without killing the program.

```
def sum_num(num_list):
    num_list=[3,52,6,'b',2,463,'a']
    totalSum=0
    for item in num_list:
        try:
            totalSum+=item
        except TypeError:
            print "Warning, item '%s' not the right type! Continuing to next item" %item
    print totalSum

sum_num(num_list)
```

5. Print stdout to a log file with `python script.py > logfile`

```
#### global variable
import time
logfile=open('jobname.log','a')

def function_name(arguments):
    start=time.clock()
    code doing things
print >>logfile,"%s was parsed in %fs."
       %(input_file,(time.clock() - start))

function_name(arguments)
```

OR, when submitting to a cluster:

```
#### open and close each time
import time

def function_name(arguments):
    start=time.clock()
    code doing things
with open('jobname.log','a') as f:
        f.write("%s was parsed in %fs."
               %(input_file,(time.clock() - start)))

function_name(arguments)
```