Python Cheat sheet (from your instructor)

Note: All of these commands are based on Python 2.x versions not 3.x
This by no means is comprehensive, but it will help you with a few tasks in lab 1 and HW 1.

1) Reading a file into the python environment

```
file='myfile.txt'
fl=open(file,"r")
```

2) Writing and calling executable python files

If you are writing a function that you will store in a file and call from the terminal command line, you need to include the path to python (see above) as the first line in the file. For the following example it is this: #!/usr/bin/python
The path varies depending on operating system. To find out where your path is, type

```
which python
```

at the terminal

```
#!/usr/bin/python
```

To call it on the command line, you must make the file executable by using the command chmod on the terminal line:

```
chmod 777 myscript.py
```

To run it use ./ before you call it:

```
./myscript.py
```

if your script must read a data file, you call it as so:

```
./myscript.py datafile
```

Okay now you try:
Copy this script into a text editor, save it with any name you like with a .py extension and run on the command line.

```
#!/usr/bin/python
import math
import numpy as np
import matplotlib.pyplot as plt

x=np.arange(0,10,0.2)
#arange function creates an array of values
# in this case we start with 0, end with 10 and step
# in increments of 0.2
y=np.sin(x)
plt.plot(x,y)
plt.show()
```

3) Defining a function:

A. In the python environment:
Example 1:

```
def simple_ave(a,b):
    foo = (a+b)/2
    return foo
#call it in the python environment as so:

val1 = simple_ave(4.0,5.0)
print val1

val2 = simple_ave(77.0,14.3)
print val2
```

Example 2:

```
def pythag(a,b):
    foo = (a**2+b**2)**0.5
    return foo

val1 = pythag(3.0,4.0)
print val1

val2 = pythag(8.0,5.0)
print val2
```

B. Creating a function inside a script is very useful when you need to call a task several times.

```
#!/usr/bin/python
import math
import numpy as np
import matplotlib.pyplot as plt

#Sine function with amplitude a.
def myfun(a,t):
 foo=a*np.sin(t)
 return foo

x=np.arange(0,10,0.2)
amp=5
#run the function
y=myfun(5,x)
#Now plot it.
plt.plot(x,y)
plt.show()
```

4) Reading column data. There are many variations on this and many more elegant than the one

presented here. However, I know this one works.

Suppose you have already read a data file into the python environment. Everything is organized into 10 columns. The variable for the file is called "fl"

```
for lines in fl:
  lines=lines.strip() # removes the return character at end
  col = lines.split() # splits the line into separate columns
# extract the first column and save it to a variable:
  x = col[0]
#Extract the 10th column and save to a variable:
  y = col[9]
```

5) Lists and Arrays

Create an empty list:

```
stuff =[]
```

create a list of 100 elements all equal to 5:

```
stuff= [5]*100
```

create a list of words:

```
stuff = ['I','like','peanut','butter','and','jelly']
```

Access 3rd element in a list

```
stuff[2]
```

Change an element:

```
stuff[5]='chocolate'
print stuff
```

Change more than one element to the same value

```
stuff = [1]*10
stuff[3:5]=2
print stuff
```

Add an element to a list/array:

```
stuff = [1]*5
stuff.append(2)
print stuff
```

Return the length of an array:

```
len(stuff)
```

iterate over an array:

method 1:

```
stuff = ['I','like','peanut','butter','and','jelly']
for word in stuff:
    print word
```

method 2:

```
for i in range(len(stuff)):
     print stuff[i]
```

Arrays are multidimensional lists - they can also be just 1D too. Create a 1D array sequence using the package NumPy

```
import numpy as np
stuff=np.arange(0,10,0.2) #(start, end, increment)
```

NumPy is nice because you can utilize more powerful array functions than just a simple python array (list)

Other NumPy Array Creation:

random array

```
stuff = np.random.rand(10)
```

create array of 100 elements of the same value (3):

```
stuff = np.array([3]*100)
```

change elements based on criteria:

```
stuff = np.arange(10)
stuff
    ## output array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
stuff[stuff>4] = 20
stuff
    ## output array([ 0,  1,  2,  3,  4, 20, 20, 20, 20, 20])
```