**BE490/Bio 7.91 Python Mini-tutorial—DAY 2**

**More unix commands:**

- mv : move a file
  ```
  mv ~/optput/files/outputfile.txt .
  ```
- rm: remove a file
  ```
  rm ../output/junk/*
  ```
- mkdir: make a directory
  ```
  mkdir new_directory
  ```
- cp: copy a file
  ```
  cp ../output/junk/* ~/newjunk/
  ```
- >: redirects
  ```
  cat test1.txt test2.txt test3.txt > test_all.txt
  ```
- chmod: changes the permissions on a file
  ```
  chmod +x my_script
  ```
  ←makes a script executable

**More Python:**

- array length: find an array length using the command len e.g.
  ```
  a=[1,1,1,1]
  len(a)
  ```
  ← produces length 4
- multidimensional arrays
  ```
  x=[[1,2],[3,4]]
  a=[1,2,3,4]
  b=[a,a,a,a]
  ```
  ← note this is copy by reference!! To copy by value use the copy command:
  ```
  import copy
  c=[copy.copy(a),copy.copy(a), copy.copy(a),
  copy.copy(a)]
  ```
  This version is copied by value, so later changes in a will not change c.
- # : comments. Lines started with # are ignored by Python, so you can use these to document code.
- Splitting strings: sometimes you want to split a string on a character
  e.g.
  ```
  s='hi there, how are you?'
  t=s.split(', ')
  ```
  ← splits the string into an array divided by the character combination ", " in this case. Split strings can be done on any length of strings.
  NOTE: We will talk about better ways of doing this with regular expressions later on.
- File I/O
  ```
  Input_file_handle=open('file_name')
  Output_file_handle=open('file_name2','w')
  ```
  ←w means write
  ```
  Output_file_handle.write('some stuff')
  Some_text=input_file_handle.read()
  Output_file_handle.close()
  ```
  ←closes a file

- HTML Input

```
import urllib2     ←library of url handling routines
url=urllib2.urlopen('http://www.google.com')
s=url.read();
```

- Flow control

**If statements**

To make comparisons we can use if statements

```
A=1
B=2
C=2
if A==B:
     print 'A equals B'
elif A==C:
     print 'A equals C'
else:
     print 'A does not equal B or C'
```

note the terms, == (equal), != (not equal), <= (greater than or equal to), < greater than) etc.

**For Loops**

for a counter, make an array using the range function e.g.
a=range(10) ← makes an array with elements 0,1,2,..9
loops are controlled by indentation

```
a=range(10)
for element in a:
     print element
```

You can stop a loop at any point with the command break

```
for element in a:
     if element==2:
          print 'found 2'
          break
```

**While loops**

While loops can go on for an unspecified period.

```
A=1
while A<20:
     A=A+1
     print A
```

- System commands

```
import os  ←library of operating system handling routines
os.system('ls -l')
os.system('mkdir new_folder')
os.system('cp ../../help/output/*
~/new_folder/home/')
```

- Executable scripts: to make your scripts runnable without typing python myscript.py, include in your script as the first line

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

  then change its permissions

  ```
  chmod +x myscript.py
  ```

- Command line arguments: sometimes you may want to run with command line arguments

  e.g. python myscript.py 1 3 input.txt output.txt

  To have python take these input arguments include the following in your code

  ```
  import sys
  ```

  then everything is stored in an array sys.argv. You can see them by printing them out:

  ```
  print sys.argv
  ```

**In Class Project**:

Parse the string in fasta.txt to obtain the reverse compliment of the sequence section alone. Output this new string to a file called output.