BTRY 4381/6381 – Bioinformatics Programming

Problem Set 1

Due Friday, February 19, 11:59 PM

In order to complete this assignment, you will need to log into the teaching server (bscb-teaching.cb.bscb.cornell.edu). Your assignment will be graded based on the content of the directories and files in your home directory on the teaching server at the time of collection ***(DO NOT EMAIL YOUR SOLUTIONS)****. Some grading steps will be automated so it is very important that you follow all instructions. Late submissions and submissions not adhering to the guidelines will be penalized.*

✓ Problem 1 Unix Exercises

a. Make a new directory in your home directory called **hw1**. All files you create for this assignment should be saved in this directory.

b. Assign the following permissions to the hw1 directory:

rwx r-- ---

c. Move into the hw1 directory. Create an empty file called 'test.txt'. Assign it the following permissions:

rwx rw- r--

d. There is a public directory on the server used for storing files that will be necessary for the completion of this and future assignments. It is located in the directory one level higher than your home directory. Use the ls command to output a list of the files in public/hw1_files to a file called 'ls_simple.txt'.

e. Use the ls man pages to find the combination of single-letter arguments needed to meet the following criteria when listing all files in a directory:

- use a long listing format (list each file on its own line and show file info)
- sort by modification time
- use reverse ordering
- print file sizes in human readable format

Use the ls command with these arguments to output a list of all files in public/hw1_files to a file called 'ls_xxxx.txt' where the four x's are replaced by the four single-letter arguments you used for ls.

f. Move into the public directory. Use the pwd command to print the absolute path of the public directory and write this output to a file in your hw1 directory. Call this file 'pwd_public.txt'.

The spirit of the problem statement: *The following problems ask you to write scripts using Python. When a starting data structure is given, such as a list, you may not redefine the data structure by hard-coding any of its values, and you may not print any output that is not a result of manipulating the given data using python functions. In other words, your solution must be generalizable. Though we may provide you with a specific input, your script should work on any similar input without changing the rest of your code.*

If you are unsure whether a step in your solution adheres to the spirit of the problem statement, please ask well ahead of the deadline! This policy applies to all future problem sets as well.

Templates*, including the starting data structures for each of the following problems, are available in public/hw1_files. Save each of your own scripts in the hw1 directory you made in Problem 1.*

✓ Problem 2 Python Control Structures

monty_python = ["John Cleese", "Terry Gilliam", "Eric Idle", "Terry Jones", "Michael Palin", "Graham Chapman"]

a. Write a script, 2a.py, that prints the average number of characters of names in this list (including spaces). Do NOT round to the nearest whole integer.

b. Write a script, 2b.py, that prints the length in characters (including spaces) of each person's name in the given list and formats the output as : "x's name contains y characters". The names should be listed in reverse alphabetical order by first name with one statement per line. i.e the first printed line should be:

```
Terry Jones's name contains 11 characters
```

c. Write a script, 2c.py, to print the members of Monty Python in alphabetical order by first name, but only those whose last names start with letters that come before M in the alphabet. Print one name per line.

✓ Problem 3 Python Collections

random_codons = ['TTA', 'TGT', 'TCT', 'AAT', 'GTA', 'TGT', 'GGG', 'TGT', 'AAG', 'GGA', 'CCT', 'CGC', 'CTC', 'AAT', 'TGT', 'TAA']

a. Write a script, 3a.py, that prints all unique codons from the list given above in reverse alphabetical order. Print one codon per line.

stop_codons = ['TAG', 'TAA', 'TGA']

b. Write a script, 3b.py, that prints (one per line) the codons in the order given in 3a (not the order you calculated), but only lists each codon the first time it is encountered in the list. If the codon is a stop codon, instead of printing the codon, print 'STOP'. You must use the given list of stop codons in your solution.

seq = 'ATGTTATGCTAGCTTACTACTGCGCACTGTCGTGGCTAGCTGATCGATCGATCGCTGATCGTAGCTAAA'

c. Write a script, 3c.py, to traverse the given sequence in the three forward reading frames. (starting at the first nucleotide, the second, and the third). Compile three strings, one for each reading frame, each one a sequence of symbols corresponding to the type of codon observed at that triplet of nucleotides. When a non-stop codon is encountered, the symbol should be a period (.), and when a stop codon is encountered, the symbol should be an exclamation point (!) AND the string should terminate (read in no more codons). Print each of the three strings on a separate line and in the correct order. *i.e. the first string should correspond to the reading frame starting with nucleotide 1, the second with a reading frame starting with nucleotide 2, and so on.*

example: The sequence 'ATGTTAATTTAG' would return:

. . . !

. !

. . .