Homework 2

Instructions: This homework is due at the beginning of lab, Wednesday, September 13th. You can edit this file and copy/paste into it by opening it in Libreoffice on Linux, PDFfiller Google App, Acrobat, MS Word, and others. Bioinformaticians are especially agile at using the internet to gather and share information, so if you do not remember or see the answer in class materials, feel free to use the web, but do not plagiarize it. Also, like anything you can obtain from the internet, information about bioinformatics may be wrong or only partially correct. Be sure to check the source of the information to help judge its quality. This particular homework will require you to write three Python scripts. I would like you to submit a digital copy of these scripts to me, along with your answers to questions 1 and 2 and output files for questions 3 and 4, all compressed into one .zip file. Make sure to run each script to ensure they work. I also encourage you to write and test each script one line at a time and test every variable you define.

1. Write a Python script that calculates the volume of a cylinder given a height of 6.1cm and a radius of 3.2cm. What is the volume?

Solution: The volume is 196.24cm^3, calculated using the following python script:

```
import math


height = 6.1

radius = 3.2


#vol = pi*r^2*h

volume = math.pi * radius**2 * height

print volume
```

2. Quick questions about strings and lists:
    a. What does mutable mean?
    b. Are strings mutable?
    c. Are lists mutable?
    d. Which data types can be stored in a list?
    e. How does the list sort function sort lists of lists?

Solutions:

a. Mutable means that it can be modified in place
b. No
c. Yes
d. Any data type
e. Lists of lists are sorted by the first element of each sublist

3. Write a Python script that imports the fasta file, “hw2seqs.fasta”, found in the “/ptmp/bcbio444/hw2” folder in hpc-class, and translates all sequences starting with the first “AUG” in the sequence and stopping at the first STOP codon. Put the translated sequences in a new fasta file, preserving the original names.

Solution: I accept both the standard input/output that was used here as well as dynamic input/output. See the following Python script and the script’s output below

Script:

```
inp = open("hw2Seqs.fasta")

out = open("hw2Seqs.fasta.prot", "w")




#define translator dictionary

translator = {"UUU":"F", "UUC":"F", "UUA":"L", "UUG":"L"\

, "UCU":"S", "UCC":"S", "UCA":"S", "UCG":"S"\

, "UAU":"Y", "UAC":"Y", "UAA":"STOP", "UAG":"STOP"\

, "UGU":"C", "UGC":"C", "UGA":"STOP", "UGG":"W"\

, "CUU":"L", "CUC":"L", "CUA":"L", "CUG":"L"\

, "CCU":"P", "CCC":"P", "CCA":"P", "CCG":"P"\

, "CAU":"H", "CAC":"H", "CAA":"Q", "CAG":"Q"\

, "CGU":"R", "CGC":"R", "CGA":"R", "CGG":"R"\

, "AUU":"I", "AUC":"I", "AUA":"I", "AUG":"M"\

, "ACU":"T", "ACC":"T", "ACA":"T", "ACG":"T"\

, "AAU":"N", "AAC":"N", "AAA":"K", "AAG":"K"\

, "AGU":"S", "AGC":"S", "AGA":"R", "AGG":"R"\

, "GUU":"V", "GUC":"V", "GUA":"V", "GUG":"V"\

, "GCU":"A", "GCC":"A", "GCA":"A", "GCG":"A"\

, "GAU":"D", "GAC":"D", "GAA":"E", "GAG":"E"\

, "GGU":"G", "GGC":"G", "GGA":"G", "GGG":"G"}
```

```
#Go through fasta file, search for the first "AUG", translate all codons from
#"AUG" to the first STOP codon.  Put the translated sequences in a new dictionary
protDict = {}
for line in inp:
    if line.startswith(">"):
        name = line.strip()
    else:
        seq = line.strip()
        start = seq.index("AUG")
        allAA = ""
        for i in range(start,len(seq),3):
            codon = seq[i:i+3]
            aa = translator[codon]
            if aa == "STOP":
                break
            else:
                allAA = allAA + aa
        protDict[name] = allAA

#Go through protein dictionary and output data in the fasta format
for name, seq in protDict.items():
    out.write(name + "\n")
    out.write(seq + "\n")



inp.close();out.close()
```

output file:

```
>seqA
MYKGISSAE
>seqB
MVDTLSPPFQ
```

```
>seqC
MPP
```

4. Write a Python script to import the file “atv10_asm.fa.fa” , found in the “/ptmp/bcbio444/hw2” folder in hpc-class, a real fasta file for the Arabidopsis thaliana genome, and calculate the GC content of each chromosome in the file. Output the info into a tab separated 2 column file where the first column is the chromosome and the second is GC content. You must use dynamic inputs and outputs for this script (no hardcoded file names). Also, you must calculate GC content using a function.

Solution: One thing I did in my script that may not have been obvious to everyone is that I excluded “N” characters from the GC content calculation. This is done because the “N” character means that the base pair is unknown, meaning that it could be a G or C and therefore including them as “not G or C” will skew your GC content calculation. Also, There are other IUPAC nucleotides that need to be dealt with in a smart way. See the following Python script below, and below that see the script’s output.

UNIX command:

```
python hw2_q4.py atv10_asm.fa.fa atv10_asm.fa.fa.gcContent
```

Python Script:

```
import sys
inp = open(sys.argv[1])      #atv10_asm.fa.fa
out = open(sys.argv[2], "w") #atv10_asm.fa.fa.gcContent



#Function to calculate GC content, not counting "N"s and others
def GetGC(seq):
    #remove all uninformative bases
    newSeq = ""
```

```
        for bp in seq:
            if not bp in ["N", "R", "Y", "K", "M","B","D","H","V"]: #all these are bases that contain no
    relevant information regarding GC content
                newSeq = newSeq + bp

        #Go through informative bases and count GC bases
        gcCnt = 0
        for bp in newSeq:
            if bp in ["G","C","S"]: #GC bases
                gcCnt = gcCnt + 1

        #calculate GC content and return the result
        gcContent = float(gcCnt) / len(newSeq)

        return gcContent



    #Go through input fasta file, calculate GC content, and output the data
    for line in inp:
        if line.startswith(">"):
            name = line.strip().strip(">")
        else:
            seq = line.strip()
            gc = GetGC(seq)
            out.write(name + "\t")
            out.write(str(gc) + "\n")



inp.close();out.close()
```

Output:

```
Chr1    0.358734686899
Chr2    0.358642974177
```

```
Chr3	0.363310749414
Chr4	0.362040216932
Chr5	0.359389263742
chloroplast	0.362938411942
mitochondria	0.447694890495
```