Project 1

Objectives

- ➢ Using File input/output
- ➢ Practice with loops and conditionals

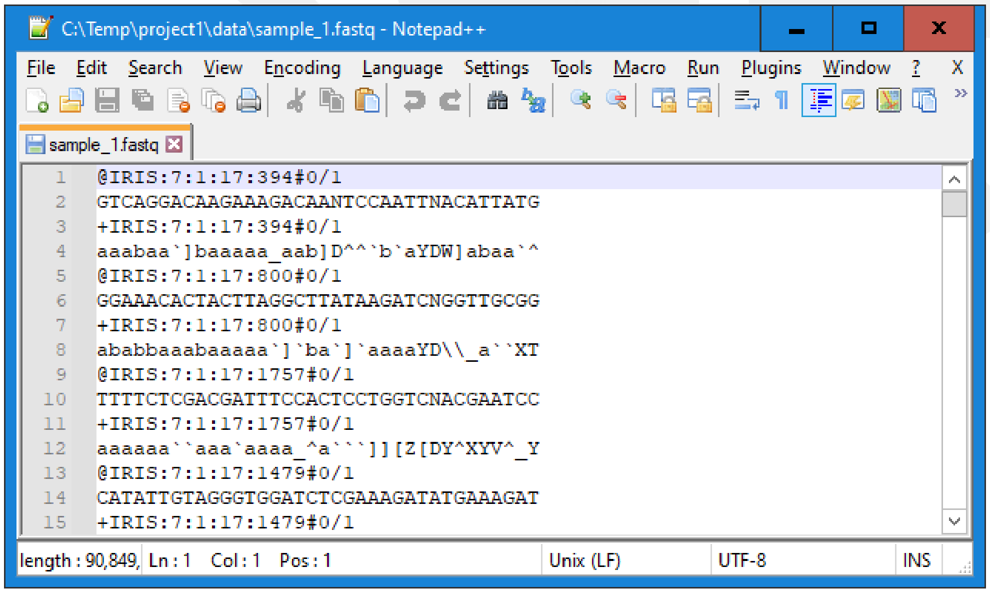
This project is based on the DNA Analysis homework in Michael Ernst's CSE 140 class at UW.

Introduction

You will use, modify, and extend a program to compute the GC content of DNA data. The GC content of DNA is the percentage of nucleotides that are either G or C. DNA can be thought of as a sequence of nucleotides. Each nucleotide is adenine, cytosine, guanine, or thymine. These are abbreviated as A, C, G, and T. A nucleotide is also called a nucleotide base, nitrogenous base, nucleobase, or just a base. Biologists have multiple reasons to be interested in GC content.

GC content can identify genes within the DNA, and can identify types of genes. Genes tend to have higher GC content than other parts of the DNA. Genes with longer coding regions have even higher GC content. Regions of DNA with higher GC content require higher temperatures for certain chemical reactions, such as when copying/duplicating the DNA. GC content can be used in determining classification of species. If you are curious, Wikipedia has more information about GC content. That reading is optional and is not required to complete this assignment.

Your program will read files produced by a high-throughput sequencer — a machine that takes as input some DNA, and produces as output a file containing a sequence of nucleotides. Here are the first 8 lines of output from a particular sequencer:


C:\Temp\project1\data\sample_1.fastq - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ? X
sample_1.fastq
length : 90,849, Ln : 1 Col : 1 Pos : 1 Unix (LF) UTF-8 INS

```
 1  @IRIS:7:1:17:394#0/1
 2  GTCAGGACAAGAAAGACAANTCCAATTNACATTATG
 3  +IRIS:7:1:17:394#0/1
 4  aaabaa`]baaaaa_aab]D^^`b`aYDW]abaa`^
 5  @IRIS:7:1:17:800#0/1
 6  GGAAACACTACTTAGGCTTATAAGATCNGGTTGCGG
 7  +IRIS:7:1:17:800#0/1
 8  ababbaaabaaaaa`]`ba`]`aaaaYD\\_a``XT
 9  @IRIS:7:1:17:1757#0/1
10  TTTTCTCGACGATTTCCACTCCTGGTCNACGAATCC
11  +IRIS:7:1:17:1757#0/1
12  aaaaaa``aaa`aaaa_^a```]][Z[DY^XYV^_Y
13  @IRIS:7:1:17:1479#0/1
14  CATATTGTAGGGTGGATCTCGAAAGATATGAAAGAT
15  +IRIS:7:1:17:1479#0/1
```

The nucleotide data is in the second line, the sixth line, the tenth line, etc. Your program will not use the rest of the file, which provides information about the sequencer and the sequencing process that created the nucleotide data.

Part 1a: Obtain the files and add your name

Obtain the files you need by downloading the project1.zip file from the Canvas site for this project. (This is a large download — be patient.)

Unzip the project1.zip file to create a project1 directory/folder. You will do your work here. The project1 directory/folder contains:

- dna_analysis.py, a partial Python program that you will complete
- answers.txt, a file where you will answer textual questions
- data, a directory. Which contains the data that you will process: It contains *.fastq files, which are output from DNA sequencers; this is the data that the program analyzes
- expected_output, a directory containing example runs of the final result of your dna_analysis.py program.

You will do your work by modifying two files — dna_analysis.py and answers.txt — and then submitting the modified versions. Add your name to the top of each of these files.

Each problem will ask you to make some changes to the program dna_analysis.py (or to write text in the answers.txt file, or both). When you do so, you will generally add to the program. Do not remove changes from earlier problems when you work on later problems; your final program should solve all the problems.

By the end of the assignment, we would like dna_analysis.py to produce output of the exact form as given below:

```
GC-content: ___
AT-content: ___
G count: ___
C count: ___
A count: ___
T count: ___
Sum count: ___
Total count: ___
seq length: ___
AT/GC Ratio: ___
GC Classification: ___
```

Where ___ is replaced by values that you will calculate. Of course, the exact values in each category will vary depending on the input data that you are using. We expect the formatting of your program output to exactly match this.

Tasks:

- Add your name at the top of dna_analysis.py and answer.txt

Part 1b: Run the program

It is a good idea to check the correctness of your program by comparing it to a computation done in some other way, such as by hand or by a different program. We have provided the test-small.fastq file for this purpose. First, examine the file by hand to determine the GC content. Then, run your program to verify that it provides the correct answer for this file.

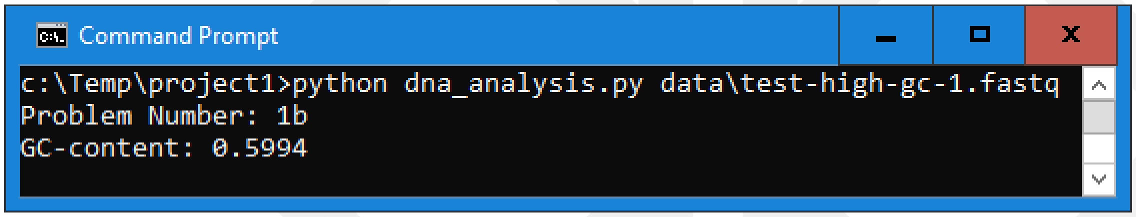
Run your program by opening a terminal and navigating to your project1 directory, then typing the following command.

- python dna_analysis.py data/test-small.fastq (python3 on Mac)

If you get a "can't open file 'dna_analysis.py'" error or a "No such file or directory" error, then perhaps your working directory is not project1, or you mistyped the file name.

After you have confirmed that your program runs correctly on test-small.fastq, run your program on each of the 6 real sample_N.fastq files provided by executing the command as shown above. Replace the test-small.fastq file with the 6 sample files Be patient — you are processing a lot of data, and it might take a minute or so to run.

Provide a screenshot of the executed file sample_3.fastq and the output. Below is a sample screenshot using the data file test-high-gc-1.fastq:



```
c:\Temp\project1>python dna_analysis.py data\test-high-gc-1.fastq
Problem Number: 1b
GC-content: 0.5994
```

(If you are interested, sample_3.fastq and sample_4.fastq are from Streptococcus pneumoniae TIGR4, and sample_5.fastq is from Human herpesvirus 5.)

Tasks:

- Use the file small.fastq to run the initial program and compare it to a manual count (just for testing)
- Execute the program on the six files (sample_N.fastq) (just for testing)
- Create a screenshot of the executed file sample_3.fastq including output (part of assignment upload)

Part 2a: Compute AT content

Augment your program so that, in addition to computing and printing the GC ratio, it also computes and prints the AT content. The AT content is the percentage of nucleotides that are A or T. Two ways to compute the AT content are, you may use whichever approach you prefer:

- Copy the existing loop that examines each base pair. You will now have two loops, one of which computes the GC count and one of which computes the AT count.
- Add more statements into the existing loop, so that one loop computes both the GC count and the AT count.

Check your work by manually computing the AT content for file test-small.fastq, then comparing it to the output of running your program on test-small.fastq.

Tasks:

- Make your program modifications
- Add a comment after each line you add: # Problem 2a
- Update the problem number at the end of the program to 2a
- Run your program on sample_1.fastq:
  - Provide screenshot of executed file and output
  - Cut-and-paste the relevant line of output into answers.txt

Part 2b: Count nucleotides

Augment your program so that it also computes and prints the number of A nucleotides, the number of T nucleotides, the number of G nucleotides, and the number of C nucleotides.

When doing this, add at most one extra loop to your program. You can solve this part without adding any new loops at all, by reusing an existing loop.

Check your work by manually computing the results for file test-small.fastq, then comparing them to the output of running your program on test-small.fastq.

Run your program on sample_2.fastq. Cut-and-paste the relevant lines of output into answers.txt (the lines that indicate the G count, C count, A count, and T count).

Tasks:

- Make your program modifications
- Add a comment after each line you add: # Problem 2b
- Update the problem number at the end of the program to 2b
- Run your program on sample_2.fastq:
  - Provide screenshot of executed file and output
  - Cut-and-paste the relevant line of output into answers.txt

Part 2c: Sanity check the data

For each of the 11 .fastq files, compare the following three quantities:

- the sum of the A count, the C count, the G count, and the T count
- the total_count variable
- the length of the seq variable. You can compute this with len(seq).

In other words, compute the three numbers for test-small.fastq and determine whether they are equal or different. Then do the same for test-high-gc-1.fastq, etc.

For at least one file, at least two of these metrics will differ. In your answers.txt file, state which file(s) and which metrics. (If all the metrics are equal for each file, then your code contains a mistake.) In your answers.txt file, write a short paragraph that explains why.

Explaining why (or debugging your code if all the metrics were the same) might require you to do some detective work. For instance, to understand the issue, you may need to load a file into a text editor and examine it. We strongly suggest that you start with the smallest file for which the numbers are not all the same. Perusal of the file may help you. Failing that, you can manually compute each of the counts, and then compare your manual results to what your program computes to determine where the error lies. A final approach would be to modify your program, or create a new program, to compute the three metrics for each line of a file separately: if the metrics differ for an entire file, then they must differ for some specific line, and then examining that line will help you understand the problem.

If all of the three quantities that you measured in this part of the project are the same, then it would not matter which one you used in the denominator when computing the GC content. In fact, you saw that the numbers are not the same. In file answers.txt, state which of these quantities can be used in the denominator and which cannot, and why. If your program incorrectly computed the GC content (which should be equal to (G+C)/(A+C+G+T)), then state that fact in your answers.txt file. Then, go back and correct it, and also update any incorrect answers elsewhere in your answers.txt file.

Tasks:

- Make your program modifications
- Add a comment after each line you add: # Problem 2c
- Update the problem number at the end of the program to 2c
- Run your program on test-small.fastq for verification only.
- Run your program on the file test-high-gc-1.fastq
  - Provide screenshot of executed file and output
  - Cut-and-paste the relevant line of output into answers.txt
  - In the answer file, provide an explanation why the sum_count and total_count/seq length are different:
    - Write at least one paragraph describing how you approach this detective work
    - Write at least one paragraph to explain the differences
    - Write at least one paragraph which total(s) can be used to calculate GC content and which total(s) cannot.

Part 2d: Compute the AC/GT ratio

Sometimes biologists use the AT/GC ratio, defined as (A+T)/(G+C), rather than the GC-content, which is defined as (G+C)/(A+C+G+T).

Modify your program so that it also computes the AT/GC ratio.

Check your work by manually computing the results for file test-small.fastq. Compare them to the output of running your program on test-small.fastq.

Run your program on sample_4.fastq. Cut-and-paste the relevant lines of output into answers.txt (the line that indicates the AT/GC ratio).

Tasks:

- Make your program modifications
- Add a comment after each line you add: # Problem 2d
- Update the problem number at the end of the program to 2d
- Run your program on sample_4.fastq:
  - Provide screenshot of executed file and output
  - Cut-and-paste the relevant line of output into answers.txt

Part 2e: Categorize organisms

The GC content can be used to categorize microorganisms.
Modify your program to print out a classification of the organism in the file:

- If the GC content is above 60%, the organism is considered “high GC content”.
- If the GC content is below 40%, the organism is considered “low GC content”.
- Otherwise, the organism is considered “moderate GC content”.

Biologists can use GC content for classifying species, for determining the melting temperature of the DNA (useful for both ecology and experimentation, for example PCR is more difficult on organisms with high GC content), and for other purposes. Here are some examples:

- The GC content of Streptomyces coelicolor A3(2) is 72%.
- The GC content of Yeast (Saccharomyces cerevisiae) is 38%.
- The GC content of Thale Cress (Arabidopsis thaliana) is 36%.
- The GC content of Plasmodium falciparum is 20%.

Again, test your work. The test-small.fastq file has low GC content. We have provided four other test files, whose names explain their GC content: test-moderate-gc-1.fastq, test-moderate-gc-2.fastq, test-high-gc-1.fastq, test-high-gc-2.fastq.

After your program works for all the test files, run it on sample_2.fastq. Cut-and-paste just the relevant line of output from your program into answers.txt.

Tasks:

- Make your program modifications
- Add a comment after each line you add: # Problem 2e
- Update the problem number at the end of the program to 2e
- Run your program on sample_2.fastq:
  - Provide screenshot of executed file and output
  - Cut-and-paste the relevant line of output into answers.txt

Required files to be uploaded to Canvas:

- README.txt file - A description of your project with your name and your student ID. Please include any problems you faced, any resources you used, names of friends/tutors you received help from
- Either put all the screenshots into one document or create individual image files
- Upload all the files (README.txt, dna-analysis.py, answers.txt, document containing all screenshots or individual images)
- Make sure to complete all tasks in each section
- Your code should contain some meaningful comments
- Your code should be well organized and formatted