CSE160_HW2

January 15, 2019

1 Homework 2: DNA analysis

Due: at 11pm on Wednesday, Jan 23, 2019. Submit via Canvas.

1.1 Learning Objectives:

- Gain experience writing Python code using loops, conditionals (if statements), functions and string manipulation
- Become familiar with running Python programs from the command line and using command line parameters to locate data files
- Write Python code to analyze DNA data sets

NOTE: Although you will fill in the body of one function and call it, you do not need to write additional functions for this assignment. You are encouraged to examine the provided function filename_to_string, which opens and reads in data files, but you do not need to understand anything about opening and reading files in order to do this assignment. We have already covered everything you need to know to be able to complete this entire assignment. For all of our assignments (except your final project) you should **NOT** use parts of Python not yet discussed in class or the course readings.

Advice from previous students about this assignment: 14wi 15sp

This assignment is posted on Canvas

1.2 Background

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 data 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:

```
@SOLEXA-1GA-2_2_FC30DNN:1:2:574:1722
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
+SOLEXA-1GA-2_2_FC30DNN:1:2:574:1722
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
@SOLEXA-1GA-2_2_FC30DNN:1:2:478:1745
GTGGGGGTGATGTCCACGATTACGCCGACCGGCTGG
+SOLEXA-1GA-2_2_FC30DNN:1:2:478:1745
hhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhhh
```

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.

1.3 Problem 1: Obtain the files, add your name

Obtain the files you need by downloading the homework2.zip file. (This is a large download — be patient.)

Unzip the homework2.zip file to create a homework2 directory/folder. You will do your work here. The homework2 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:
- *.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.

In either file, keep the number of characters on a particular line below 80, otherwise your files become hard to read. One technique to do this in Python is to break large equations into smaller ones by storing subexpressions in variables. Breaking things down into smaller computations can also help with debugging.

```
In [ ]: # Code cell for working on Problem 1
```

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

```
GC-content: ___
AT-content: ___
G count: ___
C count: ___
A count: ___
T count: ___
Sum of G+C+A+T counts: ___
Total count: ___
Length of nucleotides: ___
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.

1.4 Tips

Check your output - You should validate your dna_analysis.py program's output using the Diff Checker before submitting your assignment. You can compare your output to the files given in the expected_output directory of the homework2 files. Drag a file to a window in the Diff Checker. Select text in a command prompt window and paste it into the other window and select "Find Difference!".

Editing text files - You will submit **answers.txt** as a text file. Plain text is the standard for communicating information among programmers, because it can be read on any computer without installing proprietary software. You can edit text files using any text editor. If you use a word processor, then be sure to save the files as text. Windows users should never use Notepad, as Notepad will mangle the line endings in the file; WordPad or Notepad++ are better alternatives.

Command Prompt - In the past some students have had trouble running from the command line when folder names had spaces in them. You may find it easiest to avoid this (that is, rename your folders so they do not have spaces in the name).

To cut and paste things from the Windows command prompt, left click and drag the mouse to select text from anywhere in the command prompt window. When you have highlighted what you want, hit return. Now you can use control-V to paste into Diff Checker or a text file.

You do not need to modify *answers.txt* **or** *dna_analysis.py* **for this problem.**

1.5 Problem 2: Run the program

When writing programs that analyze data (or any other type of program) it is important to check the correctness of your programs. One way to do this is by comparing the output of your program 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. This file is small enough that you can easily open it up in a text editor and calculate the GC content by hand. Then, run your program to verify that it provides the correct answer for this file.

You should be able to open up *test-small.fastq* and other input and output files in a text editor of your choice (although just clicking on a .fastq file is not likely to work since the .fastq file extension is not one that is known to your operating system).

(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.)

If you have already used the Diff Checker, you might notice that some of your results are different than the example results. Don't worry about this — this issue will be resolved in Problem 6.

Cut and paste the line of output produced by your program regarding GC-content when run on *sample_1.fastq* into your answers.txt file. See "Tips" at the top of this page for info on cutting and pasting things from the Windows command prompt. (Note, this could take a minute or so to run.) For example, your answer might look like:

GC-content: 0.42900139393 (**Note:** This is not the answer you should expect to get; this is just an example of the format that your answer should be in.)

```
In [ ]: # Code cell for working on Problem 2
```

1.6 Problem 3: Remove some lines

In your program, comment out this line:

```
gc_count = 0
```

by prefixing it by the # character. Save the file and then re-run the program from the comand line, just as you did for Problem 2.

In *answers.txt*, explain what happened, and why it happened.

Now, restore the line to its original state by removing the # that you added. What would happen if you commented out this line instead? (Feel free to try it!)

```
nucleotides = filename_to_string(filename)
```

Explain what happens and why in *answers.txt.*

```
In [ ]: # Code cell for working on Problem 3
```

1.7 Problem 4: 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:

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

You may use whichever approach you prefer. Add whatever new variables you need.

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.

Run your program on *sample_1.fastq*. Cut-and-paste the relevant line of output into ***answers.txt***.

```
In [ ]: # Code cell for working on Problem 4
```

1.8 Problem 5: 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 - the number of **C** nucleotides.

Add whatever new variables you need.

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. At this point you should also feel free to modify the code we have given you if another structure of *if statements* makes more sense to you. We just caution you against looping through the data more times than you need to as this could cause your code to run **very slowly**.

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_1.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).

```
In [ ]: # Code cell for working on Problem 5
```

1.9 Problem 6: Sanity-check the data

For each of the eleven .fastq files you have been given, calculate and print the following three quantities:

- the sum of: the **A count, the C count, the G count, and the T count** (store this in a new variable called **sum_counts**)
- the total_count variable (total number of nucleotides)
- the length of the nucleotides string variable. You can compute this with len(nucleotides).

In other words, compute the three quantities for *test-small.fastq* and then do the same for *test-high-gc-1.fastq*, etc.

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

Explaining why (or debugging your code if all three quantities were the same in all .fastq files) might require you to do some detective work.

This exercise is meant to expose you to a **situation you might encounter when processing a data file of your own** (say on your Final project). When your program does not give the results you expect, there are two likely sources of the problem. **One is that your program contains a bug!** Check your code carefully to be sure you are calculating all values correctly. We will talk about testing in more detail later but for now, try walking through your code with a very small data set and calculating values by hand. A second source of unexpected results that is very common with data files is that there is **something you were assuming about the contents of the data files that was an incorrect assumption**. This could include things like assuming each line would contain a certain number of characters or words, or that all characters would be uppercase or lowercase, or that values might only be in a certain range. If you wrote your program assuming something about your data files that was not correct, your program may not give correct results.

To track down a wrong assumption about a data file, think about ways you can modify your program to help you determine what is happening. This could include having it **print out values** when they do not meet some asumption you are making about the file. You could also try just loading a data file into a **text editor** and examining it with your eyes to see if you see something you did not expect. (Although if you try this approach we strongly suggest that you start with the **smallest data file** for which the three quantities are not all the same.) Another approach would be to **modify your program, or create a new program**, to compute the three quantities for each line of a data file separately (as opposed to for the file as a whole as you have been doing): if the quantities differ for an entire file, then they must differ for at least one specific line in that file. Examining that/those line(s) will help you understand the problem.

If all of the three quantities that you measured in problem 6 are the same, then it would not matter which one you used in the denominator when computing the GC content. However, you saw that the three quantities are not all the same. In ***answers.txt***, state which of these quantities should be used in the denominator and which should not, 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 your program, and also update any incorrect answers elsewhere in your ***answers.txt*** file. It is fine to change the code we provided you if needed.

If you are unsure if you are calculating things correctly, now would be a good time to validate your *dna_analysis.py* program's output using the Diff Checker. You can compare your output to the files given in the expected_output directory of the homework2 files. You have not yet completed the assignment, so your output will not be identical. But things like GC-content, AT-content and individual counts should be identical. You will produce the last two lines of output in the expected_output files in Problem 7 and Problem 8 below.

```
In [ ]: # Code cell for working on Problem 6
```

1.10 Problem 7: Compute the AT/GC 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_1.fastq*. Cut-and-paste the relevant lines of output into ***answers.txt*** (the line that indicates the AT/GC ratio).

```
In [ ]: # Code cell for working on Problem 7
```

1.11 Problem 8: Categorize organisms

The GC content can be used to categorize microorganisms. Fill in the body of the classify function to return the classification of the organism ("high", "moderate" or "low") described in the data file given using these classifications:

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 that your program works on some data files with known outputs. 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*.

You will find a "skeleton" or "stub" for the classify function near the top of dna_analysis.py, just before where the main program begins. There is an assignment statement inside of the function body that is there only as a placeholder. You should edit/remove it once you have added your code. The function classify takes the gc_content as an input parameter and returns an appropriate string indicating the GC Classification. Once you have filled in the body of the classify function you should call the function from your main program in the appropriate place and use the string it returns to print out a message that matches what is expected. For example, be sure that your output for *test-moderate-gc-1.fastq* matches *test-moderate-gc-1-expected.txt* exactly using Diff Checker.

After your program works for all the test files, run it on *sample_1.fastq*. Cut-and-paste just the relevant line of output from your program into ***answers.txt***.

```
In [ ]: # Code cell for working on Problem 8
```

1.12 Submit your work

You are almost done!

We recommend doing a quick search on this web page for answers.txt to confirm that each place we asked you to answer a question, you have answered it.

At the bottom of your *answers.txt* file, in the "Collaboration" part, state which students or other people (besides the course staff) helped you with the assignment, or that no one did.

Submit the following files via this turnin page.

```
dna_analysis.py
answers.txt
```

Please be sure to validate your dna_analysis.py program's output using the Diff Checker before submitting your assignment. You can compare your output to the files given in the expected_output directory of the homework2 files. Be sure that both the values are correct AND that the messages you are printing are formatted correctly and have no typos.

Answer a **REQUIRED** survey (See Canvas site) asking how much time you spent and other reflections on this assignment.

Now you are done!

```
In [ ]: # Code cell for checking dna_analysis.py file code
```