Part A

Read Chapter III (for class 3) of the text material.

Part B – all codes in Python

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 more extended coding regions have even higher GC content.

Regions of DNA with higher GC content require higher temperatures for some chemical reactions, such as when copying/duplicating the DNA.

GC content can be used in determining the 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 specific sequencer – that's just an example:

```
@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. Get the files for HWK 2:

Obtain the data 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 includes 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 within a particular line below 80. One technique to do this in python would be to break large equations into smaller ones by storing subexpressions in variables.

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

```
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.

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 IDLE or another text editor. If you use a word processor, then be sure to save the files as text. Windows users should never use Notepad for any purpose because Notepad will mangle the line endings in the file; WordPad or Notepad++ are better alternatives.

2. Run the program

It is a good idea to check the correctness of your application 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 shell or command prompt (not IDLE's Python interpreter), navigating to your **`homework2`** directory, then typing the following command.

On Mac/Linux:

```
python dna_analysis.py data/test-small.fastq
```

On Windows:

```
python dna_analysis.py data\test-small.fastq
```

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 **`homework2`**, or you mistyped the file name.

After you have confirmed that your program runs correctly on **`test-small.fastq`**, run your application on each of the 6 real **`sample_N.fastq`** files provided, by executing 6 commands such as

```
python dna_analysis.py data/sample_1.fastq
```

or if you are a Windows user,

```
python dna_analysis.py data\sample_1.fastq
```

You will have to change **`sample_1.fastq`** to a different file name in the subsequent commands. Be patient — you are processing a lot of data, and it might take a minute or so to run.

If you have already used the Output Comparison Tool (referenced at the bottom of the page), you might notice that some of your results are different than the example results. Don't worry about this — this issue will be resolved later — item 6.

Cut and paste the line of output regarding **GC**-content from **`sample_1.fastq`** **into your `answers.txt` file. For example, your answer might look like**

```
GC-content: 0.42900139393
```

(Note that this is not the answer you should expect to get, this is just an example of the format that your answer should be in.)

3. Remove some lines

In your program, comment out these lines

```
seq = ""
linenum = 0
```

by prefixing them by the **#** character. Re-run the program, just as you did for the previous part.
In **answers.txt**, explain what happened, and why it happened. Now, restore the lines to their original state by removing the **#** that you added.

What would happen if you commented out this line?

```
gc_count = 0
```

Explain (in **answers.txt**).

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 base pair. You will now have two loops, one of which calculates the **GC** count, and one of which calculates the **AT** count.

Add more statements into the existing loop, so that one loop computes both the **GC** count and the **AC** count.

You may use whichever approach you prefer.

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.**

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, 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_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).

6. 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 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.

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).

8. 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 conten*t".
If the **GC** content is below 40%, the organism is considered "*low GC conten*t".
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_1.fastq**. Cut-and-paste just the relevant line of output from your program into **answers.txt**.

9. Correlations: NOT GRADED:
    a. Go back to the MS Excel Worksheet **Sample_01** from HWK 1. Write a code in Python to count the number of missing values, mistaken values, and outlier values for *Prop_01*, …, *Prop_07*.
    b. Write a code in Python to compute the correlation matrix between all columns with and without missing values, mistaken values, and outlier values, replacing letters by numbers (for example, A→1, B→2, etc.).