Guanine-Cytosine Percentage Calculator

This project can be found at the following link: <http://rosalind.info/problems/gc/>

The aim of this project is to enter several sequences in the FASTA format, and their GC percentage is analysed in order to return the sequence with the greatest percentage.

Why is this project important? What are its uses?

Guanine, and cytosine are nucleotides – joined to a deoxyribose (pentose) sugar and phosphate backbone to form a single unit of DNA; these units are then assembled in different sequences to code for proteins. This is achieved because a single amino acid is coded for by a codon, also known as a triplet which are three nucleotides in the DNA sequence. A protein is simply a collection of amino acids.

Therefore, being able to compare several different sequences could be useful in analysing a pathology. For example, the different sequences entered could all be for the same gene, yet some of the genes could be associated with a particular pathology. Analysis of the GC percentage could reveal whether there is a hidden relation between certain nucleotides and the pathology.

It can also make it easier to locate the affected region in a DNA sequence. For example, if there is an increased GC quantity in a DNA sequence, the region of DNA with increased GC density could be the region affecting the disease. This could be useful for analysis of neurodegenerative diseases such as Huntington’s disease, which is characterised by long repetitive sequences of …….. which make Huntingtin protein

Can I add any additional features to this project?

I can add integration with matplotlib to plot the GC and AT percentages for each sequence.

How will I go about solving this project?

1. The first step is to receive an input from the user. This input is in the FASTA format which means the title of sequence is entered, and then on a separate paragraph the sequence is entered.
   1. I believe a simple solution to this is to use a dictionary. A dictionary is a list where a variable is assigned a value, known as key-value pairs. This can be implemented by assigning the title input of the sequence as the key, and then assigning the DNA sequence to the value.
      1. This solution using a dictionary does not work. This is because a dictionary does not have an innate ability to index the key-value pairs so they cannot be recalled from their position; instead, they must be called using the name of the sequence, but this will defeat the purpose of using a dictionary because to implement this solution regular lists must be used alongside.
   2. This way, the key-value pair can be retrieved when the element is called, and offers a simple and efficient solution to entering the sequence.
2. Another problem I will encounter is the number of inputs the programme can accept. I do not want to add a limit to how many can be entered because realistically there might be thousands of sequences which might need to be compared which would limit the functionality of the programme if only three could be compared at a time.
   1. This problem’s solution is aided by the use of a dictionary which can store a limitless amount of data. However, how do I go about adapting the programme to store varying amount of sequences?
   2. A potential solution is to use a *while* loop. If the user wants to quit entering the sequences, they can write *q* which sets a variable to false and prevents further looping.
   3. The termination is now sorted, however, how do I store several user inputs into different elements in the list?
      1. I need to initialise the list: example\_0 = {}
      2. To access the dictionary: example\_0[‘key’]
      3. At the end of the loop I can make a variable known as sequence\_number to which +1 is added after each cycle. Then the example\_n can be modified where example\_{sequence\_number}