# EX1

## Part 1

Write a Python script to perform the following tasks on the strings:

String1 (it is one long string!)

-CAGGTGGTGATGGCGGAAAGGTCACACCCGAACACATCCCGAACTCGGAAGTTAAGCTTTCCAGCGCCGATG-GTAGTTGG--GGGTTTCCCCCTGCGAGAGTAGGACGTTGCCGGGCAACGGCGGTCATAGCGGTGGGGAAACGCCCGGTCCCATCCCGAACCCGGAAGCTAAGCCCACCAGCGCCGATG-GTACTGCACTC-GTGAGGGTGTGGGAGAGTAGGACGCCGCCGGACGCTGGCGACCATAGCAAGAGTGAACCACCTGATCCCTTCCCGAACTCAGAAGTGAAACCTCTTCGCGCTGATG-GTAGTGNGG-GT-GTA-CCCATGTGA-AGTAAGTCATCGCCAGCTGTCGGTGGTCATTGCGGAGGGGGAACGCCCGGTCCCATCCCGAACCCGGAAGCTAAGCCCTCCAGCGCCGATG-GTACTGCACTC-GCCAGGGTGTGGGAGAGTAGGTCGCCGCCGACA

String2 (it is another one long string!)

--CAGGTGTGATGGCGGAAAGGTCACACCCGAACACATCCCGAACTCGGAAGTTAAGCTTTCCAGCGCCGATG-GTAGTTGG--GGGTTTCCCCCTGCGAGAGTAGGACGTTGCCGGGCAACGGCGGTCATAGCGGTGGGGAAACGCCCGGTCCCATCCCGAACCCGGAAGCTAAGCCCACCAGCGCCGATG-GTACTGCACTC-GTGAGGGTGTGGGAGAGTAGGACGCCGCCGGACGC-TGGCGACCATAGCAAGAGTGAACCACCTGATCCCTCCCGAACTCAGAAGTGAAACCTCTTCGCGCTGATG-GTAGTGNGG-GT--TA-CCCATGTGAGAGTAAGTCATCGCCAGCTGTCGGTGGTCATTGCGGAGGGGGAACGCCCGGTCCCATCCCGAACCCGGAAGCTAAGCCCTCCAGCGCCGATG-GTACTGCACTC-GCCAGGGTGTGGGAGAGTAGGTCGCCGCCGACA

* Calculate the base occurrence frequency, per string. The strings contain bases ("A", "G", "T", "C"), but also contain errors (e.g., "-", "N"). An example of the output should be like (note that these are not the results!):

TOTAL BASES FOR STRING 1: 100, ERRORS: 5, "A": 45, "G": 25, "T": 15, "C": 10. TOTAL BASES FOR STRING 2: 100, ERRORS: 5, "A": 35, "G": 35, "T": 10, "C": 15.

* Calculate the number of matches (per base) and mismatches, when you align the strings. E.g., you can already see 3 matches at positions 5, 8, 9 (G<-->G, G<-->G, T<-->T), and 7 mismatches (-<-->-, C<-->-, A<-->C, G<-->A, etc) in the left area of the two strings:

-CAGGTGGT... (first string)

--CAGGTGT... (second string)

An example of the output should be like: A-A matches=10, G-G matches=20, T-T matches=15, C-C matches=5, mismatches=100.

## Part 2

Here are some DNA segments that are experimentally verified sequences with high affinity to GATA3 transcription factor:

AGATAG

AGATAG

AGATAG

AGATAG

CGATAG

AGATAC

AGATAA

GGATAC

CGATTG

CGATTG

AGATAG

AGAGAG

CGATAC

AGATTG

CGATAG

AGATAC

AGATAA

GGATAC

CGATTG

CGATTG

Write a Python script to process those strings in order to calculate the Position Frequency Matrix (PFM). The PFM is a matrix of the form:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Base/Position** | 1 | 2 | 3 | 4 | 5 | 6 |
| A | 11 | 0 | 20 | 0 | 15 | 2 |
| T | ... | ... | ... | ... | ... | ... |
| G | ... | ... | ... | ... | ... | ... |
| C | ... | ... | ... | ... | ... | ... |

The values in the cells are the frequencies of bases' occurence in strings' positions. E.g., 11 means that "A" appears 11 times in the 1st position of each string, 0 means that that "A" appears 0 times in the 2nd/4th position of each string, etc. An example of the output should be like:

A: 11, 0, 20, 0, 15, 2

T: .........

G: .........

C: .........

## Notes

* Questions only in our Teams channel.
* Prepare only one notebook, and send it to dalamag@athenarc.gr and ggeorgakilas@athenarc.gr using the subject in the exact form: PYTHON BOOTCAMP 2024 EX1 LASTNAME FIRSTNAME
* Explain your code/solutions with markdown text inside the notebook. No markdown, no pass. In any case, your notebook should work like a charm!
* Deadline (**Thu Oct 3 - 23:59**), STRICT (really).