Bonus Assignments II

1 Bonus Assignment - Functions

Open an editor and save your new program. In this program we will create a few functions.

1. Define the two functions `similarity` and `distance`:

$$similarity(a,b) = \begin{cases} 1, & \text{if} \quad a = b \\ 0.5, & \text{if} \quad a \neq b, \text{a and b are both purines or pyrimidines} \\ 0, & \text{if} \quad a \neq b, \text{a and b are both not purines or pyrimidines} \end{cases}$$

$$distance(a,b) = \begin{cases} 0, & \text{if} \quad a = b \\ 0.5, & \text{if} \quad a \neq b, \text{a and b are both purines or pyrimidines} \\ 1, & \text{if} \quad a \neq b, \text{a and b are both not purines or pyrimidines} \end{cases}$$

   Note: Purines are `A` and `G`, pyrimidines are `C` and `T`.

2. Write two functions `compare1` and `compare2`, which calculate for sequences of same length the similarity and distance, respectively.

3. Calculate the similarity and distance for the following sequences. Read these sequences from the command line and print out their similarity and distance.

   a) `ACGT` and `TGCA`

   b) `ATAG` and `ACAC`

   c) `ACGC` and `ATTT`

   d) `AGTT` and `ACTT`

   e) `TCGC` and `AGAG`

2 Bonus Assignment - Modules

In this exercise we will write three different programs.

1. Write two Python programs (e.g., `Sim.py` and `Dist.py`) which contain respectively the two modules `similarity` and `distance` as defined previously.

2. Write a third Python program that calculates for each combination of two sequences stored in list `l` the similarity and distance using the modules defined previously.
   `l = ["ATCCGGT", "GCGTTAC", "CTACTGC", "TTGCAGT", "AGTCACC"]`

3. Extend your third program. Determine the alignment with the highest similarity of all sequences stored in list `l`. Write these two sequences and the alignment into a new file, called `similar_sequences.txt`.

   For example for two given sequences: `"ATC"` and `"ACC"` The alignment would be:

   ```
   ATC
   | |
   ACC
   ```

   And this alignment should be written to a new output file.

   Hint: A line-break in Python can be made by adding `'\n'` to the end of the line.