Programming Assignment 5: Simple DNA Sequences

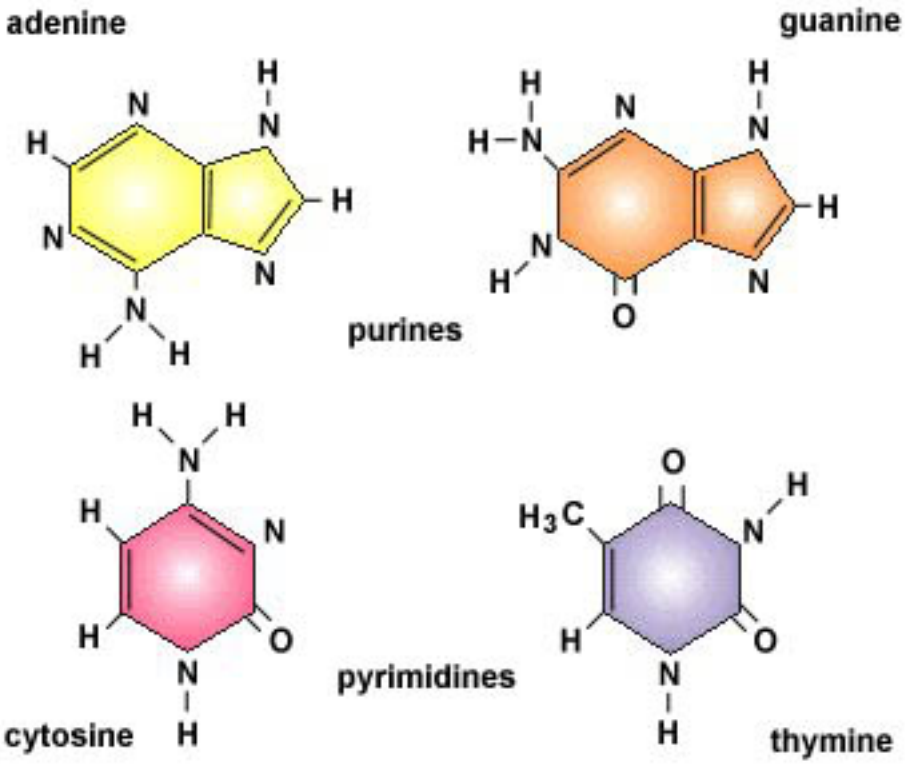
COP 3035 - Fall Term 2019
Point Value: 100 points
Project Due Date: *Tuesday 11/19/2019*

Learning Objectives

- To write a program which requires the use of a standard Python data structure, *strings* and/or *lists*
- To create *parallel* sequence structures
- To perform text file input
- To design a program on your own which is well-structured and modular, utilizing multiple functions and parameter passing appropriately

Problem Statement

DNA is at the heart of the *genetic code*, not just for humans, but for most living organisms on Earth. A DNA sequence is often represented by a string or list made up of the letters A, T, G and C. These letters represent the nucleotides adenine, thymine, guanine and cytosine. To find the complement of a DNA sequence, As are replaced by Ts, Ts by As. Gs by Cs, and Cs by Gs. For example, the complement of AATTGCCGT is TTAACGGCA.



Finding the complement of a DNA sequence is relevant for several reasons. First, DNA structure consists of a *double helix* where each pair within the helix consists of complements. Second, whenever cell division occurs DNA must be replicated, and complements must pair with one another.

Other important concepts in *computational biology* include the idea that DNA can *mutate*, and that specific *sequences* and *patterns* of elements, can be very important.

For this project, you will do these major tasks:

First, read in a DNA sequence from a file, and store it in a Python string or list. Note: we will keep our sequences very short, in comparison to DNA sequences in real life, to make programming and grading this project simpler. Print the sequence you read in, to show the contents read in from the file. This is your *original* DNA sequence.

Provide the user with a *menu*. The menu must allow the user to choose among four operations:

(1) Determine the complement of the original DNA sequence read in, and print both the original and the complement in a parallel output format, for ease of comparison.

(2) Create 5 random simulated simple mutations in the DNA sequence. That is, in 5 positions your program selects pseudo-randomly, insert an "M" into the position to replace the A, T, G or C that was previously there. Then print both the original and the mutated sequence in parallel output format, again, for ease of comparison.

(3) Allow the user to enter a substring that he or she wants to search for in the original DNA sequence. For example, the user might search for a sequence such as "AGTCA" and find out where this sequence is located. In this program, you only need to find the first instance of such a substring, and report at what index it was located at. If the substring is not found, you must report that.

(4) Quit the program.

Input

The input files provided are named *dna1.txt*, *dna2.txt* and *dna3.txt* and are available from the course web site. You may also make up your own text files to test.

You may assume that data files are not empty, do contain one DNA sequence, and do not contain any data errors.

You must ask the user to input the filename they wish to use. If the file does not open with the filename they entered, keep asking them for a filename until the file does open successfully.

Output

- The original DNA sequence will be displayed
- For each operation the user selects, be sure to output appropriate information.
- Be sure to follow all class style guidelines for all required output elements and formatting principles

Data Structures

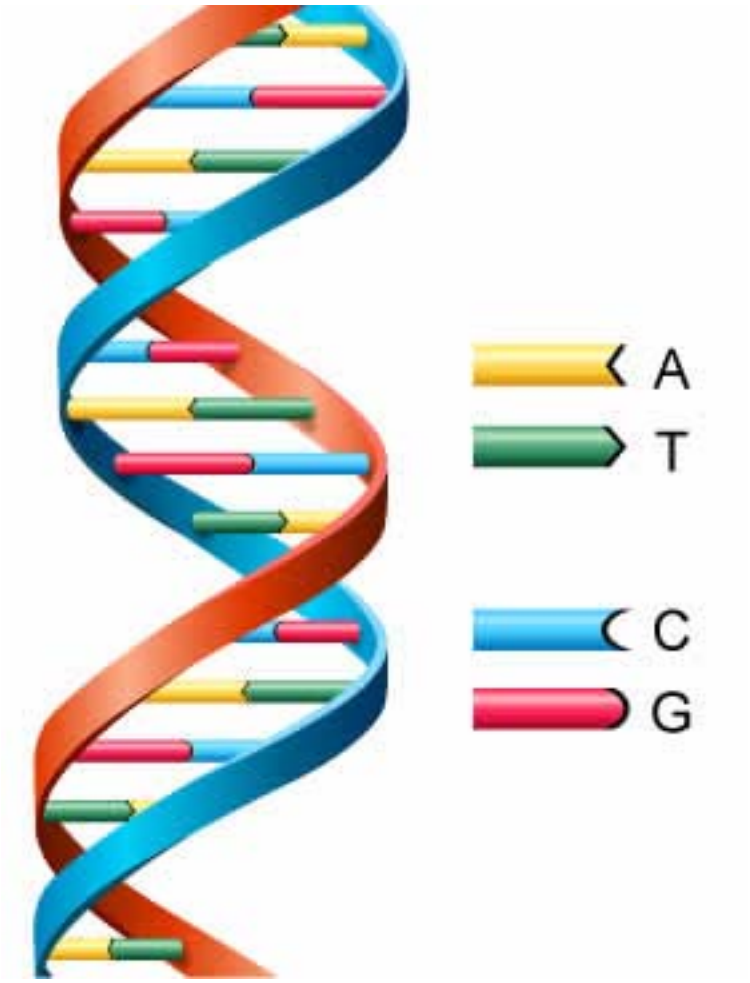
This project is intended to familiarize you with the use of strings and/or lists in Python. You may choose what data structure(s) you wish to use, but they must be strings or lists.

Note on lists: if you decide to use lists, you MUST use a list where each element is a single character. That is, *each character must have its own index number* in the list or string that you use to store the sequence. This will automatically occur with strings.

Utilize only information provided in chapters 1 through 8 of the required course Python textbook, and accompanying lecture materials.

Use of Functions and Program Modularity

Part of your grade on this and all future course programming projects will be determined by how well you utilize functions and parameters appropriately. Start by working on a good design, structure chart, etc. Your program must represent a modular, functionally cohesive design following class style guidelines or substantial grade penalties may be incurred.



What File To Turn In, File Naming Requirements, and How to Turn In Your Work using Canvas

You must turn in your Python program source file which must be named as follows (note that you will have to rename the provided file):

Use this format: *yourLastNameLowerCase_FSUID_p5.py*
Your FSUID will be unique to you, will typically be your FSU email ID, and will be something like "ab23c." Hence file names will look something like "smith_ab23c_p5.py"

Submit your Python file (.py) to Canvas using the Submit button for this assignment. Be sure to download the file after you submit it in order to check that you submitted the correct program file to Canvas and that it was successfully received by Canvas.

Last Update: 7/29/2019 A. Ford Tyson
