CSCA08 Assignment 0

Due: October 22, 2017. 11:55pm

Introduction

The goal of this assignment is to get you started writing python functions, and to evaluate your ability to create high quality efficient code. The work you will be doing in this assignment is not harder than what you have been doing in exercises up to this point, but this code will be marked by TAs who will evaluate not only whether your code works, but also whether it is efficient and well designed. Also remember that the assignments must be completed **entirely on your own**. So if you get stuck, the TAs can help with general concepts, but cannot look at your code or help you directly with your implementation. The same goes for your fellow students. No one should ever look at your assignment code and you should never look at anyone else's (until after it's submitted, at which time you can feel free to share and compare).

What Not to Use

You can complete this assignment using only the material covered in the first four weeks of the course. In particular, you are not allowed to use loops anywhere in the assignment (there is no good reason to use them in this assignment!) or any data structures not covered in weeks 1-4.

DNA Sequencing

This assignment deals with DNA sequencing. A DNA sequence is a series of nucleotides: adenine (A), guanine (G), cytosine (C) and thymine (T). These sequences can be represented as strings of their first letters. e.g., `GCACTAG`.

Within these DNA sequences, researchers are interested in finding specific genic sequences (genes). The input we receive from DNA sequencers usually comes in a continuous stream, so the string may start with some upstream sequence (data from a previous gene), and may continue beyond the end of the specific gene that is of interest with a downstream sequence (the start of another gene). Fortunately, all genes start with the sequence `ATG`, and the sequence `ATG` cannot appear in the middle of a gene [1]. This makes it possible to isolate and analyze a specific gene from a sequence.

Your Tasks

For this assignment, you will be required to build the following 5 functions:

- `split_input`: Takes in a DNA sequence (as described above) and returns a list with three elements, the upstream data, the gene (if any is found, or an empty string if no gene is found), and the downstream data, in that order
- `get_gene`: Takes in a DNA sequence (as described above) and returns a string representation of the gene if one is present, or the string `ERROR` if no gene is present.
- `validate_gene`: Takes in a string representation of a gene, and returns True iff the gene presented is valid. For a gene to be valid it must satisfy the following critera:
  - It must start with the start codon (3 character sequence) `ATG`
  - It must contain at least one codon after the start codon
  - It must contain only full codons (i.e., it cannot end mid-way through a 3 character codon)
  - It must never contain four consecutive identical nucleotides

[1] Note: This isn't actually true in the real world, but it makes our lives a lot easier, so we'll pretend

- `is_palindromic`: Takes a string representation of a gene, and returns True iff that gene is palindromic (reads the same forwards as backwards).
- `evaluate_sequence`: Takes in a DNA sequence (as described above) and returns one of the following strings as appropriate: {`No Gene Found`, `Invalid Gene`, `Valid Gene Found`, `Valid Palindromic Gene Found`}.

Marking

Your assignment will be marked for correctness in a similar manner to your exercises. But it will also be marked by a TA for elements such as:

- Programming style: Your variable names should be meaningful and your code as simple and clear as possible.
- Commenting: Your documentation should be clear and concise and allow a user who has not read this handout to fully understand how to use and manipulate your functions.
- Code re-use: You should have as little duplicated code as possible. If you find yourself repeating code, there's a good chance you could find a simpler (lazier) method.
- Testing coverage: Your test cases should not only cover all input categories, but should also be clearly labelled and organized in a sensible manner.

What to Submit

Submit `a0.py` and `a0.test` on MarkUs. Your file must be named exactly as given here (check that MarkUs says you have submitted all required files after you're done submitting).

Before you submit:

- Ensure that you have read & added your name and login to the header at the top of the file
- Test your code for PEP-8 compliance
- Run DocTest and make sure you pass all your own cases
- Re-test all examples

Happy Sequencing!