Programming Exercises

Before you start programming exercises, you need to become familiar with the command line and ideally with using git and GitHub.

For and introduction to those, please use my Self-Paced Learning Activity Tutorials (SPLATs): http://www.bioinf.org.uk/teaching/splats/

Exercise 1

This exercise is going to focus on translating a DNA sequence into a protein sequence using a Python dictionary to map DNA codons to amino acids.

Part 1a

Write some code to translate a DNA sequence (stored as a string) into a protein sequence.

For this first part you can make a number of simplifications and assumptions:

First, simply specify the DNA sequence within your code:

```
dna = 'caggtgaagctgcagcagtcaggggctgagctggtgaggcctggagcttcagtgaagctgtcctgcaaggcttctggctactccctcaccagctactggatgaactgggtgaagcagagg'
```

Second, assume that the DNA sequence is in-frame and shown in the correct forward direction.

The output for the shown DNA should be something like:

```
GlnValLysLeuGlnGlnSerGlyAlaGluLeuValArgProGlyAlaSerValLysLeuSerCysLysAlaSerGlyTyrSerLeuThrSerTyrTrpMetAsnTrpValLysGlnArg
```

or

```
QVKLQQSGAELVRPQASVKLSCKASGYSLTSYWMNWVKQR
```

Part 1b

Improve your translation code

There are 7 incremental steps you can take to improve the code you have written:

- Take the sequence from the command line instead of 'hard-coding' it within your program - e.g. you would run your program as

```
./translate.py caggtgaagctgcagcagtcaggggctgagctggtgaggcctggagcttcagtgaagctgtcctgcaaggcttctggctactccctcaccagctactggatgaactgggtgaagcag
```

- Read the sequence from a file instead of from the command line. At this stage, assume that the file contains one sequence and nothing else, but that it could be split across multiple lines - e.g. you would run you program as

```
./translate.py sequence.txt
```

- Read the sequence from a file in FASTA format - in other words the file might contain multiple sequences and the sequences may contain spaces for formatting. Just translate the first sequence. In FASTA format, each sequence is given a header so you read the first header and sequence, but stop when you come to the next header. The header has a `>` sign and a label in the form `>AB1743178`
- Improve the output format - give the option of producing 3-letter code or 1-letter code. Show only 20 amino acids per line if it is 3-letter code and 60 amino acids per line if it is 1-letter code.
- Add 3-frame (forward) translation. To do this, as well as translating as normal, simply skip the first base and translate, and then skip 2 bases and translate.
- Add 6-frame translation. To do this, you need to build on the previous step by reverse-complementing the sequence and do a 3-frame translation on that as well.
- Add reading and translation of multiple sequences from the FASTA file. This is surprisingly difficult as the FASTA format indicates the start of a sequence, but not the end of a sequence. So you need to keep track of the previous header (for the sequence you have just read) and the header you have just encountered that shows you a new sequence has just started.

Exercise 2

This exercise is going to focus on doing some simple calculations on a PDB file containing a protein structure. The PDB format is actually quite difficult to read properly (there are issues such as alternative atom positions to deal with, optional columns of information, differences in formatting of legacy files), so we are going to use a simple example from the small plant seed protein crambin which has none of these issues. The file `1crn.pdb` will be used for this purpose.

PDB files contain a keyword (of up to 6 characters) at the start of each line which indicates the type of information present. We are only interested in lines that start with `ATOM`

The format of these atom records in a PDB file is:

```
ATOM      1  N   THR A   1      17.047  14.099   3.625  1.00 13.79           N
ATOM      2  CA  THR A   1      16.967  12.784   4.338  1.00 10.80           C
ATOM      3  C   THR A   1      15.685  12.755   5.133  1.00  9.19           C
ATOM      4  O   THR A   1      15.268  13.825   5.594  1.00  9.85           O
ATOM      5  CB  THR A   1      18.170  12.703   5.337  1.00 13.02           C
ATOM      6  OG1 THR A   1      19.334  12.829   4.463  1.00 15.06           O
ATOM      7  CG2 THR A   1      18.150  11.546   6.304  1.00 14.23           C
ATOM      8  N   THR A   2      15.115  11.555   5.265  1.00  7.81           N
ATOM      9  CA  THR A   2      13.856  11.469   6.066  1.00  8.31           C
ATOM     10  C   THR A   2      14.164  10.785   7.379  1.00  5.80           C
ATOM     11  O   THR A   2      14.993   9.862   7.443  1.00  6.94           O
ATOM     12  CB  THR A   2      12.732  10.711   5.261  1.00 10.32           C
ATOM     13  OG1 THR A   2      13.308   9.439   4.926  1.00 12.81           O
ATOM     14  CG2 THR A   2      12.484  11.442   3.895  1.00 11.90           C
```

- The 1st column is the record type (`ATOM`)
- The 2nd column is the atom number
- The 3rd column is the atom type
- The 4th column is the amino acid type
- The 5th column is the chain label (to deal with quaternary structure or crystal structures where there is more than one copy of the protein)
- The 6th column is the amino acid number (this can be followed by an insert code, but we won't worry about that for this exercise)
- The 7th-9th columns are the x,y,z coordinates
- The 10th column is the occupancy
- The 11th column is the B-factor (temperature factor)
- The 12th column is the element type

Part 2a

- Write a program that will take the filename of a PDB file on the command line and will calculate (and display) the centre of geometry of the protein. In other words, it will read the x,y,z coordinates and calculate the average x, average y and average z.
- Modify the program such that it only calculates the centre of geometry for the C-alpha (CA) atoms.

Part 2b

- Write a program that will display the ATOM records for those atoms with a B-factor greater than a cutoff also specified on the command line.
- Modify the program such that it displays the ATOM records for the whole amino acid where any atom has a B-factor greater than the cutoff
- Modify the program such that you calculate the mean and standard deviation of the B-factors and display atoms with a B-factor greater than the mean plus a specificable number of standard deviations.

(Ideally these different options can be specified on the command line.)

Part 2c

- Write a program that takes a PDB file on the command line and outputs the amino acid sequence as 3-letter codes (just look at the C-alpha (CA) so you only look at one atom from each amino acid).
- Modify the program such that it outputs 1-letter code and uses FASTA format.
- Modify the program such that it outputs the sequence for multiple chains separately (use PDB file `1yqv.pdb` for this purpose).

Part 2d

- Write a program that takes a PDB file on the command line and displays the amino acid usage. (i.e. the number of each type of amino acid present).
- Modify the program such that it also displays the percentage of each amino acid type
- Modify the program such that it also displays the overall charge of the protein (assume GLU and ASP are negative while ARG, LYS and HIS are positive).
- Modify the program such that it does these calculations separately for each chain in a PDB file (use PDB file `1yqv.pdb` for this purpose).
- Modify the program such that it can take either a PDB file as input or a FASTA sequence file

Part 2e

- Write a program that takes an amino acid identifier (e.g. A20 - the chain and the amino acid number), a PDB file and a distance from the command line. It then prints all the identifiers for all amino acids which have a C-alpha (CA) within the specified distance of the C-alpha of the specified amino acid.
- Modify the program such that it looks for residues with *any* atom within the specified distance of *any* atom in the specified amino acid. (Note that this is a bit more tricky since you only want to display each in-distance amino acid once. You can use a dictionary to record which identifiers you have already displayed.)