**Bi 410/510: Introduction to Programming for Biologists**

Spring 2018

# Project 5: I/O

To complete this project you need to write five small Python programs that read and write data files. Put all five programs in a single folder named `io` and compress the folder:

```
$ zip -r io.zip io
```

Upload `io.zip` to Canvas as your submission for Project 5.

**Note:** The examples for each program show the results of calculations. You do not have to print your results in the same format – we'll be looking for correct results and good style but not the same exact output format.

### Data

For project 2 (Shell Commands) you worked with a folder named `sim`. You'll need that folder again for this project. We suggest either moving `sim` to the directory you're using for this project or downloading `sim.zip` and making a fresh new copy.

You'll also need copies of two other files on Canvas, `genetic_code.csv` and `amino_acids.py`. Download these files and save them in the same folder as your programs.

### Program 1: Clusters

A subfolder of `sim` named `clusters` has a file named `clusters.txt`, which is a TSV file produced by a program that analyzes DNA sequences and organizes them into clusters of similar sequences. The relevant information for this project is in the first two columns of each line:

- Column 1 has a sequence ID, which is a long string that starts with a series of numbers and ends with `'size='` (you'll use the size in the next project, but ignore it for now).
- Column 2 has a string, either `'otu'` or `'match'`.

Write a program named `clusters.py` that reads a cluster description file and prints the sequence ID field for every OTU, *i.e.* for every record that has `'otu'` in the second column. This is what the first three lines should look like for this data set:

```
$ python clusters.py sim/clusters/clusters.txt
715:51:674:768;id=2007;size=36
810:412:436:962;id=119;size=14
259:174:253:24;id=2281;size=11
...
```

**Program 2: Cluster Size**

Write a program named `cluster_size.py` that will read a cluster description file and compute some statistics about cluster sizes.

> **Hint:** see the lecture notes `(IO.html)` for examples of code that extracts the cluster size from the ID string on each line.

The program should print the number of clusters, the total number of sequences in all clusters, and the mean cluster size. For example, suppose there are three clusters in a file:

```
30:321:33:30;id=940;size=10  otu   74.8  * 715:51:74:768;id=20
61:325:49:19;id=295;size=7   otu   83.0  * 715:51:74:768;id=20
93:074:93:31;id=200;size=5   otu   85.4  * 715:51:74:768;id=20
```

The sizes are 10, 7, and 5, so the total number of sequences is 22 and the mean cluster size is 7.33.

This command shows how to run the program on the `sim` data and the result I got for `clusters.txt`:

```
$ python cluster_size.py sim/clusters/clusters.txt
number of clusters:                11
number of sequences in clusters:   99
mean cluster size:                 9.0
```

**Program 3: Translate**

Write a program named `translate.py` that will translate a DNA sequence into an amino acid sequence. The command line arguments are the name of a file with a genetic code and the DNA sequence to translate.

For the previous project you wrote a program that iterated over a DNA string and printed each codon. On this project you want to look up each codon in a dictionary to find the corresponding amino acid letter and then append that letter to your output string.

The dictionary that maps codon strings to amino acid letters should be defined by reading the genetic code file you downloaded from Canvas (`genetic_code.csv`). The lecture notes (`Dictionaries.html`) show how to do create this dictionary.

Here is an example with a valid input string, where all letters are in the "DNA alphabet" and the length of the input is a multiple of 3:

```
$ python translate.py genetic_code.csv GATTACATG
DYM
```

This example shows how to handle other cases. There is a non-DNA letter in the second codon, and the input has 10 letters. The output has question marks for both cases:

```
$ python translate.py genetic_code.csv GATXAAATGA
D?M?
```

**Program 4: Codon Lists**

Write a program named `codon_lists.py`. The command line argument is the name of a CSV file with a genetic code, the same as in the previous project.

This program should create a dictionary named `codons` that maps an amino acid letter to a list of codons for that amino acid. After you build the dictionary print it with a `for` statement that iterates over all the keys in `codons` to print each list.

Here is an example of how to run the program and the first three lines of output you might see:

```
$ python codon_lists.py genetic_code.csv
A : ['GCT', 'GCC', 'GCA', 'GCG']
R : ['CGT', 'CGC', 'CGA', 'CGG', 'AGA', 'AGG']
N : ['AAT', 'AAC']
```

Note that Python might print the lines in a different order, but the list associated with each letter should always be the same.

**Program 5: Molecular Weight**

Write a program that computes the molecular weight of a protein. The command line argument passed to the program will be an amino acid sequence. The output is just the sum of the weights of each amino acid. Here's an example:

```
$ python molecular_weight.py MLSVII
764.96
```

Molecular weights are defined in a dictionary named `amino_acids`. A Python file named `amino_acids.py` has an assignment statement that creates this dictionary. Download the file from Canvas, and then either copy and paste the definition into your program, or save the file in the same folder as your program and add this statement to your program:

```
from amino_acids import *
```

Each item in the dictionary is a list of attributes for an amino acid. This example from an IPython session shows how to import the dictionary definition and use the dictionary to look up the attributes of Alanine, which has the symbol A:

```
In [1]: from amino_acids import *

In [2]: amino_acids['A']
Out[2]: ['Alanine', 'Ala', 'A', 89.09]
```

The molecular weight is the last item in the list. To get the molecular weight of Alanine:

```
In [3]: amino_acids['A'][-1]
Out[3]: 89.09
```