Programming and Databases Lab 5

Introduction

This lab has two problems. Like last week, they are slightly larger, and intended to consolidate your understanding of the material we have covered so far by getting you to use the tools you have learnt about in slightly different ways.

The only new topic covered this week is **regular expressions**.

Problem 1

This builds on the regular expression problem that we started in the lab yesterday.

The YXXΦ motif is commonly used in eukaryotic organism to target proteins to the endosome. In this problem, you will find proteins containing this motif in the genome of Toxoplasma gondii.

To remind you, this motif consists of:

- A Tyrosine (Y)
- Followed by any two amino acids
- Followed by a bulky hydrophobic amino acid (tyrosine, phenylalanine or threonine)

This motif can occur anywhere in the last 10 amino acids of the protein.

To help you out:

This is an online tool for testing regexes: https://regex101.com/
This is the one-letter amino acid code: https://www.fao.org/3/Y2775E/y2775e0e.htm

You will also need to two files for this task:
Tgondii_Proteins.fasta
Tgondii_product_descriptions.txt

As usual, you can download these from Moodle, or you can copy them from my directory if you are on the summerschool server.

```
cp ~kathryn/BIOL4292/Lab5/Tgondii_Proteins.fasta .
cp ~kathryn/BIOL4292/Lab5/Tgondii_product_descriptions.txt .
```

Remember, this is a Linux command, and should be run in the terminal, not in your Python script.

The Goal

1. Write a regex to find the YXXΦ motif, remembering that it must occur in the last 10 amino acids of the protein. Test it on some small examples (**note:** you could build this test into your code by using assert statements)
2. Iterate through the fasta file. This fasta file contains protein sequences. Find proteins have the YXXΦ motif
3. For proteins that have the motif, look up the product description from the txt file
4. Write out these proteins in a 3-column file, in which the columns contain:
    1. The gene id
    2. The product description
    3. The last 30 amino acids of the sequence with the regex match in lower case

Hints and Tips

You will want to look up gene product descriptions. What data structure do you use for **fast lookups**? You probably want to make this data structure **before** you read the fasta file.

Problem 2
When you do RNA sequencing with Illumina, the general procedure is to:

1. Align your sequence data to a reference genome
2. Count the number of reads that align to each gene in each sample
3. Normalise the counts using TPM
4. Compare TPM values between samples

You will learn more about the nuances of this process in other courses.

For this lab, I have given you count data for one sample. This represents the output of step 2 above. Your task is to write a programme to do step three – normalise these values using TPM.

TPM is calculated for each gene as follows:

- Divide the count for each gene by the length of the gene in kilobases. This gives you reads per kilobase or **rpk**
- Sum up the rpk values for all the genes
- For each gene, TPM is calculated as rpk / (sum of rpk values / 1000000)

You have two input files:
Trypanosoma_brucei_counts.txt
Trypanosoma_brucei.fasta

As usual, you can download these from Moodle, or copy them from my workspace if you are on the summerschool server.

```
cp ~kathryn/BIOL4292/Lab5/Trypanosoma_brucei.fasta .
cp ~kathryn/BIOL4292/Lab5/Trypanosoma_brucei_counts.txt .
```

Remember this is a Linux command that should be run in your terminal, not in a Python script.

The Goal
The goal is pretty simple – you need to read in the counts file, and create a two-column tab delimited file containing the TPM value for each gene.

Note that if you get this right, the sums of all your TPM values should be 1 million. In reality, it's often not exact because of cumulative rounding errors, but it should be close.

Hints and Tips

1. You will need the length of each gene for the calculation. How can you get this from the fasta file?
2. As in previous problems, remember to create dictionaries that you need to look things up in before you need to use them.
3. We have emphasised previously that in most cases you need to look things up in dicts using the keys rather than iterating over them. However, there are some cases where you do need to iterate over a dict. This problem contains an example of this.

Extras
It’s good to get into the habit of writing good code as well as just finishing the problem. Go back through your solutions, and think about:

- Should you make any functions? If you do make any, write some assertions for them.
- Can you adapt either or both of your scripts to take files at the command line?
- Are there any places where you might see run time errors? Can you write try/except blocks?