Python Exercise Week 5:

Programs should be written in Python according to the respective exercise description.The program must be correct in terms of syntax and semantic. If there exists a minimal solution using only a single pre-defined function, this function is not allowed. The weekly exercise should be uploaded on ILIAS as a single Jupyter Notebook (.ipynb). Theassignment will be provided after the lecture on Wednesday and must be uploaded by Tuesday 8:00. Make sure that answers to questions are contained within the JupyterNotebook that is uploaded on ILIAS. Use commented code when possible

---

Description

You are part of a research team that has recently conducted field analyses in several regions to identify viral DNA from environmental samples. The samples were sent to a partner lab, which performed genome sequencing and returned four viral genomes in FASTA format.

Now, it's your task to analyze these genomes to determine their relationships. The partner lab suspects that three of the viruses might be closely related, while the fourth is from a different family. Your first job is to explore the basic structure of these genomes and confirm the relationships using programming tools.

Exercise 1 - Opening and Exploring the Genomic Files

1a. The first step is to familiarize yourself with the genomic data you've received. Each FASTA file contains one complete viral genome. Your task is to write code to open each

file and print the first line (which contains the metadata or "header" of the sequence) and the first 100 nucleotides.

Instructions:

For each fasta file, open the file, read the first line and print the header. Next, print the first 100 nucleotides.

Expected output:

```
Viral genome in file virus_1.fasta has header:
>virus_1_complete
The first 100 nucleotides are:
ATGGAGACCTTGAGACCAACAGATTTGAAGAGCTATTGCACCATCTTTGGGATAGACTTTTTTGAGCTTCATTTGC
AATGTATCTTTTGTAATTGCTATT

Viral genome in file virus2.fasta ...
```

1b. Make a function that stores the information in a dictionary and create one dictionary that holds the information for all the genomes. Then print keys of the dictionary

Expected output:

```
The contig headers are:
...
```

Exercise 2 - Get basic statistics for the genomes

2a. Make a function that calculates the GC content (GC%) of each genome. Print the genome header followed by its GC content.

2b. Make a function that calculates the length of the genome. Print the genome header followed by its legth in Kbp (KiloBasePairs eg: 1200 nucleotides == 1.2Kbp)

2c. Make a dictionary that holds this information

2d. Print the genomes and their information in order, from largest to smallest

---

Exercise 3 - Create Kmers

In simple terms, **k-mers** are short sequences of nucleotides (the building blocks of DNA or RNA) that are used to analyze genetic sequences. The **"k"** in "k-mer" represents the length of the sequence, so a k-mer with k=3 would be a sequence of 3 nucleotides (e.g., "ATG").

Example:

Let's say you have the DNA sequence `ATGCTGA`, If you want to generate **3-mers** (k=3) from this sequence, you would break it down into overlapping sets of 3 nucleotides:

- "ATG"
- "TGC"
- "GCT"
- "CTG"
- "TGA"

Each 3-letter chunk is a **3-mer.** K-mers are widely used in biology because they allow researchers to analyze and compare long sequences of DNA efficiently.

3a. Create a function that given a genomic sequence as string, counts the occurrences of each kmer of size 3. Write the code in a way that you can easily change the kmer size.

- Write a function that given a string of size N returns every substring of size K as a list of strings.
- Create a variable that contains all the found substrings as a set
- Write a function that iterates over the set and counts the occurrences of every substring in the list and stores the result in a dictionary

3b. EXTRA Create your own implementation without following the previous steps!

3c. Iterate over all the genomes and create a dictionary for each genome that contains the k-mers as keys and their count as value. Print it one of them on screen.

3d. Let's see what is going on, iterate over every k-mer and print the count for each genome

Expected Output:

```
AAA     100     120     100     80
AAC     105     100     ...
```

Can you spot any patterns?

Exercise 4 - Use larger kmers to infer genetic relations

Viral genomes are characterized by quick evolution which may produce indels (inclusion or exclusion of individual nucleotides). Therefore, using %identity as we did in the lecture to look at relationship between sequences may not work, one individual indel may create a "shift" that makes the process of comparing base-pairs irrelevant. However, k-mer based analyses are more resistant to small indel or nucleotide changes as they "break down" the comparison on a smaller scale.

4a. Create a large_kmer_set variable for each genome [large_kmer_set_virus_1, large_kmer_set_virus_2, etc..] for K = 7 that contains a set of all k-mers of size 7 for a given genome.

4b. Using the set operations, calculate the overlap between the kmers of the different genomes

Expected Output:

```
viral_genome_1 and viral_genome_2 share X out of Y k-mers, Z%
viral_genome_1 and viral_genome_3 share X1 out of Y1 k-mers, Z1%
```

where X is the number of shared k-mers, y is the total number of unique kmers between the two, and Z is the ratio between these two numbers.

What type of pattern do you observe?

Exercise 5 - Translate the genomes into protein sequences

Proteins are encoded by sequences in DNA through a process called **translation** , which uses the genetic code to convert the nucleotide sequence into a chain of amino acids

(the building blocks of proteins). Each block of 3 nucleotides encodes for a specific amino acid.

You can use this dictionary for the translation:

```
# Dictionary mapping RNA codons to single-letter amino acid codes
codon_to_amino_acid = {
    "AUG": "M",      # Methionine (Start codon)
    "UUU": "F", "UUC": "F",    # Phenylalanine
    "UUA": "L", "UUG": "L", "CUU": "L", "CUC": "L", "CUA": "L", "CUG": "L",
# Leucine
    "AUU": "I", "AUC": "I", "AUA": "I",    # Isoleucine
    "GUU": "V", "GUC": "V", "GUA": "V", "GUG": "V",    # Valine
    "UCU": "S", "UCC": "S", "UCA": "S", "UCG": "S", "AGU": "S", "AGC": "S",
# Serine
    "CCU": "P", "CCC": "P", "CCA": "P", "CCG": "P",    # Proline
    "ACU": "T", "ACC": "T", "ACA": "T", "ACG": "T",    # Threonine
    "GCU": "A", "GCC": "A", "GCA": "A", "GCG": "A",    # Alanine
    "UAU": "Y", "UAC": "Y",    # Tyrosine
    "CAU": "H", "CAC": "H",    # Histidine
    "CAA": "Q", "CAG": "Q",    # Glutamine
    "AAU": "N", "AAC": "N",    # Asparagine
    "AAA": "K", "AAG": "K",    # Lysine
    "GAU": "D", "GAC": "D",    # Aspartic Acid
    "GAA": "E", "GAG": "E",    # Glutamic Acid
    "UGU": "C", "UGC": "C",    # Cysteine
    "UGG": "W",    # Tryptophan
    "CGU": "R", "CGC": "R", "CGA": "R", "CGG": "R", "AGA": "R", "AGG": "R",
# Arginine
    "GGU": "G", "GGC": "G", "GGA": "G", "GGG": "G",    # Glycine
    # Stop codons
    "UAA": "*", "UAG": "*", "UGA": "*"
}
```

5a. As you can see, the sequences are using the RNA code, that's because the DNA is first transcribed into RNA for translation. For each genome, transcribe the DNA in RNA. Print the first 100 ribonucleotides of each genome

5b. Find all the start codons (AUG) positions on the forward strand (find only returns the first position, can you think of other ways to find the start codons?) Print all the start codons positions for virus_1 forward strand

Expected Output:

```
Start codon number 1 for virus_1 at position 10
Start codon number 2 for virus_1 at position 23
Start codon number 3 for virus_1 at position 50
...
```

5c. Not all start codons result in a protein, it will result in a functional sequence only if the next stop codon is sufficiently distant to produce a useful protein product. For each starting codon, print the predicted protein if it has at least 20 amino acids before it encounters a stop codon during translation.

Do it for virus_1

```
Predicted protein from start codon number 1 at position 10 is
MECSPPYATTALMFSGAYRKNSMCWQQHATGAWCY*
Predicted protein from start codon number 3 at position 50 is
MFLISSPQDEGGMSTARLWAYSEPIS*
```

5d. Extra Some of the sequences seem a bit similar, that is because if there is more than one methionine in the sequence, it would be predicted as an additional transcript. Store the predicted proteins in a list, set or dictionary and remove all proteins that are sub-strings of another protein.

5e. Extra Repeat the process for all genomes including the reverse strand, and record the results in a dictionary for each genome. Print the average length and number of predicted proteins per genome, do you notice any patterns?