Python Exercise Week 4:

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

---

In this exercise we are writing code to analyse the gene expression of a subset of genes related to drought tolerance in arabidopsis thaliana, a plant often used in plant research. 3 plants were grown in normal conditions (control) and 3 plants were grown under drought conditions (treatment). We have extracted and sequenced mRNA from each of those plants. Analysing how plants with the same genetic background react to the stress (treatment) by regulating the expression of different genes, we can speculate on which molecular processes are involved in the physiological response to the stress. If you are very unfamiliar with the topic, you may read a bit into it here: https://www.genome.gov/genetics-glossary/Gene-Expression or watch the first two minutes from this video: https://www.youtube.com/watch?v=tIf6wYJrwKY. Regardless, you don't need to know in detail all the processes.

Out of all the transcripts sequenced, we already narrowed down to 10 candidate genes and we obtained the transcripts count for all of them across the treatments. **Note** You don't need to do every exercise perfectly and completely to continue. It's possible to continue without completing the previous ones.

Exercise 1:

1a. Thankfully, someone already wrote some code to do preliminary analysis. Unfortunately, we are not sure which variables are local and which are global, can you comment each variable and specify if it's global or local? replace the (?) Additionally, whoever wrote this, forgot to comment and describe the steps, can you please do that?

1b. Did you run the script? Whoever wrote this did not really try to make it easy to analyse the output, those float values are making it hard to draw a conclusion. Can you edit the print statements with the float formatting from week 2 lecture to round the values to the 2nd decimal when printing averages?

1c. Oh, there is something weird about one of those sequences, and I think there are interesting patterns for some of the genes. We may have to dig deep into it and figure out what is going on there! do you already have a theory?

```
#add comment
rna_sequences = [
    'GGGCUAGCGUAUCGAUGCUA', 'CGCUAUAAAUGCGUAGGUC', 'CUAGCGGUAAUCGCAUUA',
    'AUGCUAGGCUAUUAGCGA', 'UAGGCAUAAAGGCUAUGC', 'CUAGCGGUAAUCGCAUU',
'GCUAUGCAGGCAAUUGCUA', 'GGGCCGUAUCGAUGCUAGCGUAC', 'AGGCGAUAUAAACGCUUA',
'GGGCGGUCAUUAACGGAUUUGA'] #(?)
# ADD comment

treatment_1 = [120, 520, 180, 250, 500, 2, 300, 130, 530, 110] #(?)
treatment_2 = [125, 510, 185, 245, 505, 0, 310, 135, 540, 115]
treatment_3 = [118, 515, 175, 255, 495, 2, 295, 128, 525, 113]

control_1 = [140, 290, 178, 240, 300, 0, 305, 150, 310, 135]
control_2 = [145, 285, 182, 248, 295, 0, 298, 148, 300, 137]
control_3 = [138, 288, 176, 252, 305, 3, 302, 152, 315, 132]

# Add comment
def process_treatment_counts():
    total_treatment_counts = [] #(?)
    for i in range(len(rna_sequences)):
        average_count = (treatment_1[i] + treatment_2[i] + treatment_3[i]) /
3 #(?)
        total_treatment_counts.append(average_count) #(?)
    print(f"Average treatment counts per RNA sequence:
{total_treatment_counts}")

# Add comment
def process_control_counts():
    total_control_counts = [] #(?)
    for i in range(len(rna_sequences)):
        average_count = (control_1[i] + control_2[i] + control_3[i]) / 3 #
(?)
        total_control_counts.append(average_count) #(?)
    print(f"Average control counts per RNA sequence:
{total_control_counts}")
```

```
# Add comment
def calculate_average_count_per_experiment(expression_countsm
experiment_type):
    total_counts = 0 #(?)
    for count in expression_counts:
        total_counts += count #(?)
    average_count = total_counts / len(expression_counts) #(?)
    print(f"Average count for {experiment_type} experiment: {average_count}
total counts: {total_counts}")

# Add comment
process_treatment_counts()
process_control_counts()
for control_experiment in [control_1, control_2, control_3]:
    calculate_average_count_per_experiment(control_experiment, 'control')
for treatment_experiment in [treatment_1, treatment_2, treatment_3]:
    calculate_average_count_per_experiment(treatment_experiment,
'treatment')
```

Exercise 2:

2a One of the transcript that we selected has very low counts, it may be an error of sorts. I heard that sometime the sequencing machine can make mistakes and not read all the ribonucleotides in a sequence. Let's make a function that calculates the distance between all the sequences. Thankfully, someone already did most of the work for us, import the function SequenceMatcher and test its use by running the following.

```
from difflib import SequenceMatcher
string1 = 'AAAAA'
string2 = 'AAAAT'
string3 = 'TTTCC'

print (SequenceMatcher(None, string1, string2).ratio())
print (SequenceMatcher(None, string1, string3).ratio())
print (SequenceMatcher(None, string2, string3).ratio())
```

It seems that the function returns some sort of similarity between strings. Can you write a function that makes a loop and compares all the sequences to eachother? it should print sequences that are more than 90% similar.

2b EXTRA: can you do some or all of it in a list comprehension?

3c I knew it! something was off about that sequence, what do you think happened there? What should we do about it? can you do it with code?

Exercise 3:

Now we should focus on the experiment, you are free to come up with your process but we want for each sequence:

1. calculate the average expression in treatment
2. calculate the average expression in control
3. calculate a ratio between the two
4. separate the transcripts that are showing large up or down ratio changes (hint. append function. If you don't know what values to use, print all of them and see if you notice good "thresholds", there is no wrong choice as long as you can motivate it!)
5. print the relevant information

That's very interesting, I think we are onto something!

Exercise 4:

4a. A colleague of mine was reading articles on the subject and told me that some sequences have motifs, short subsequences that can be important to regulate their expression. Let's use the SequenceMatcher function and see if the sequences within a group have such a thing. (if you have not completed exercise 3, use exercise 2 but change the if condition statement to find sequences that are more than 60% similar). Do you notice something?

4b. Let's test this hypothesis, take the sequence GGGCUAGCGUAUCGAUGCUA and create all substrings of size 4 that are in it and count their occurrences in each other RNA sequence.

1. Create all the possible substrings
2. **for** each of the substrings: 3. **count** the occurrences of the substring in the other RNA sequences
3. Look at the results Oh, now I see what's going on!

4c. Do the same for CGCUAUAAAUGCGUAGGUC but use length 5. What do you think we discovered?

4d. EXTRA: Can you change some of the loops into list comprehensions?

Exercise 5:

5a. Now it's time to put all the information together. Use the zip function to put together information regarding:

1. RNA sequence
2. average count in control
3. average count in treatment
4. fold change (https://en.wikipedia.org/wiki/Fold_change#Fold_changes_in_genomics_and_bioinformatics)
5. if present, substring that was shared with other sequences (from exercise 4), if none was found, use "No motif" instead

print it on screen, make sure to format the floats to make it look "nice"

Extra Can you put this information you got in the form of a report? you can use excel or similar tools to make some figure and show what you found. What do you think it's happening here?

Exercise 6:

6a. We noticed that the RNA sequences have been provided in a compact format without clear separators between codons (groups of three nucleotides). For better readability and analysis, let's reformat these sequences separating them into codons and then joining them back together with hyphens (-) as separators. This will allow us to visualize the codons more clearly.

Task:

1. Write a function that splits each RNA sequence into codons (groups of 3 nucleotides).
2. Join the codons using the join() function, separating them with -.
3. Print the newly formatted RNA sequences.

Example formatting output, ('GGGCUAGCGUAUCGAUGCUA') becomes:

```
'GGG-CUA-GCG-UAU-CGA-UGC-UA'
```
