Python Exercise Week 6

Instructions

- Write your Python programs according to the exercise description.
- Ensure your code is correct in both **syntax** (it runs without errors) and **semantics** (it performs the intended task).
- **Do not** use a single pre-defined function if a minimal solution using it exists.
- Submit the exercise as a single Jupyter Notebook (`.ipynb`) file on **ILIAS**.
- The assignment will be provided after the lecture on **Wednesday** and must be uploaded by **Tuesday 8:00**.
- Include all answers in the Jupyter Notebook and use **commented code** to explain your approach.

Exercise 1 - Import Custom Script

1a.

- Import the function `create_random_genome` from the script `week6_accessory_functions.py`. The script is in the provided download folder.

1b.

- Use the `help` function to understand how `create_random_genome` works.
- Create 3 random genomes with sizes between **100,000** and **500,000** bases. The GC content should be randomly chosen between **20%** and **80%**.
- Save these genomes as `r_genome_1`, `r_genome_2`, and `r_genome_3`.

Expected Output Example:

```
# Example Output:
r_genome_1 is 324157 bases long
r_genome_2 is 245612 bases long
r_genome_3 is 401234 bases long
```

1c.

- Import the `custom_module` script from the lecture.
- Use its `gc_content_DNA` function to calculate the GC content of each genome (`r_genome_1`, `r_genome_2`, and `r_genome_3`) and print the results.

Expected Output Example:

```
GC content of r_genome_1: 48.7%
GC content of r_genome_2: 51.2%
GC content of r_genome_3: 37.5%
```

Exercise 2 - Finding Patterns with Regular Expressions

Handle cases where no matches are found using **if statements**.

2a.

- Find and print the first occurrence of the sequence `'ATGCATGC'` in each genome. If not found, print "Sequence not found".

Expected Output Example:

```
In r_genome_1, 'ATGCATGC' first appears at position 3500
In r_genome_2, 'ATGCATGC' was not found
In r_genome_3, 'ATGCATGC' first appears at position 12345
```

2b.

- Find and print the first occurrence of any of the following patterns: `'AAAA'`, `'TTTT'`, `'CCCC'`, `'GGGG'`. Indicate the sequence found and its position in each genome.

Expected Output Example:

```
In r_genome_1, 'AAAA' first appears at position 500
In r_genome_2, 'CCCC' first appears at position 2000
In r_genome_3, 'GGGG' first appears at position 1500
```

2c.

- Print all start positions for occurrences of `'AAAA'`, `'TTTT'`, `'CCCC'`, and `'GGGG'` in each genome.

Expected Output Example:

```
In r_genome_1:
'AAAA' at positions: 500, 1200, 3000, 5000
'TTTT' at positions: 4000, 4200, 5000
'CCCC' at positions: 800, 1600
In r_genome_2:
'GGGG' at positions: 100, 500, 900
```

Exercise 3 - Advanced Pattern Matching

Handle cases where no matches are found using **if statements**.

3a.

- Identify sequences of **5 or more** identical characters. Print their start position and length in each genome.

Expected Output Example:

```
In r_genome_1:
'AAAAA' at position 450 (length: 5)
'GGGGGGG' at position 1500 (length: 7)
In r_genome_2:
'TTTTTT' at position 2000 (length: 6)
```

3b.

- Find sequences with **8 or more** consecutive `'A'` and `'T'` characters only. Print their start position, length, and the actual substring for each genome.

Expected Output Example:

```
In r_genome_1:
'AAAAAAAA' at position 750 (length: 8)
'TATATATA' at position 1300 (length: 8)
In r_genome_2:
'AAAAAAAAAA' at position 5000 (length: 10)
```

3c.

- Identify the longest substring without any `'C'` characters. Print its start position, length, and the substring for each genome.

Expected Output Example:

```
In r_genome_1:
Longest substring without 'C' starts at position 1500 (length: 200) -
Sequence: 'ATGTTGAA...'
In r_genome_2:
Longest substring without 'C' starts at position 3000 (length: 150) -
Sequence: 'AATTGGAA...'
```

Exercise 4 - Identifying Open Reading Frames (ORFs)

Handle cases where no matches are found using **if statements**. You can do this exercise for one or all the genomes.

4a.

- Find and store all positions of the `"ATG"` start codon in a list called `start_position_list`. Print it.

4b.

- Find and store all positions of the stop codons (`"TAA"`, `"TAG"`, `"TGA"`) in a list called `end_position_list`. Print it.

4c.

- Write a loop to iterate through each `"ATG"` position and find the closest subsequent stop codon position.

4d. (EXTRA)

- Write a single regular expression to match complete ORFs directly (from `"ATG"` to the closest stop codon).

Exercise 5 - Analyzing Restriction Enzyme Sites

5a.

- Use regular expressions to find matches for the following restriction enzymes in one genome:

- **HinfI** recognizes `GANTC`
- **BsaAI** recognizes `YACGTR`

Expected Output Example:

```
Enzyme HinfI matches:
Start: 1000    End: 1005    Sequence: GAATC
Enzyme BsaAI matches:
Start: 1500    End: 1506    Sequence: CACGTA
```

5b.

- Calculate the fragment sizes produced by:
  - **HinfI**
  - **BsaAI**
  - **Both HinfI and BsaAI**
- Print in descending order.

Expected Output Example:

```
Fragment sizes with HinfI (descending order): [5000, 3000, 2000]
```