Bioinformatics — exercise 3 — Python in bioinformatics

Deadline for the result files submission: 10 days

Rules applicable to the subject, file with grades

The resulting files must be submitted via MS Teams. A minimum of 5 out of 7 points is required to pass.

In Tasks 1 and 2, the assessment will be based on code correctness, compliance with the task instructions, and the methods used.

Objective: The aim of this exercise is to familiarise students to the practical use of the Biopython library for analyzing biological data, such as DNA and RNA sequences.

Theory



Biopython is a Python library designed to assist bioinformaticians and molecular biologists in analyzing biological data. It is widely used in bioinformatics projects and enables seamless integration of biological sequence analysis (DNA, RNA, proteins) with Python. The library offers a rich set of features, including:

- loading and analyzing DNA, RNA, and protein sequences,
- manipulating biological sequences, such as transcription, translation, and calculating GC content,
- working with biological file formats, e.g., FASTA and GenBank,
- analyzing motifs and protein structures.

One of the core components of Biopython is the **Seq** object, which represents a biological sequence. It is an enhanced object compared to plain strings in Python, tailored to meet the needs of bioinformatics analysis. The **Seq** object supports many operations, such as:

- **creating sequences**: the Seq object allows storing and managing DNA, RNA, and protein sequences,
- **sequence operations**: easily compute GC content, transcribe DNA to RNA, translate RNA to proteins, or generate the reverse complement strand,
- **sequence manipulation**: the Seq object enables slicing, concatenating, and analyzing sequences.

Example of creating a **Seq** object:

```
from Bio.Seq import Seq
```

```
dna_sequence = Seq("AAGAAATTCCAAGTCCAGGGATACACAAACAGGTGTACAGC")
print(dna_sequence)
```

The **SeqIO** module in Biopython enables working with biological files such as FASTA and GenBank. It is particularly useful for loading and analyzing large collections of sequences. SeqIO allows:

- reading and writing data in various biological formats,
- iterating through multiple sequences in a file,
- easy integration with other bioinformatics tools.

Example of reading a FASTA file:

```
from Bio import SeqIO

for record in SeqIO.parse("example.fasta", "fasta"):
    print(f"Identifier: {record.id}")
    print(f"Sequence: {record.seq}")
```

Basic sequence analysis functions:

- **counting nucleotides**: built-in Python functions or methods can be used to count occurrences of A, T, C, and G,
- **calculating GC content**: Biopython provides the GC tool from the **SeqUtils** module,
- **transcription and translation**: these operations are available as methods of the Seq object.

Example of calculating GC content in a sequence:

```
# function available in version 1.80+
from Bio.SeqUtils import gc_fraction

gc_content = gc_fraction(dna_sequence)
print(f"GC content: {gc_content * 100:.2f}%")
```

FASTA is one of the most commonly used file formats for storing biological sequences such as DNA, RNA, or proteins. It is simple and consists of two parts:

1. **Header line**: starts with the **>** symbol and contains the sequence identifier and an optional description;
2. **Sequence**: represented in subsequent lines as a string of characters (A, T, C, G for DNA).

Example of a FASTA file:

```
>Seq1 Sequence description
AAGAAATTCCAAGTCCAGGGATACACAAACAGGTGTACAGC
AAATCATGTAGGTGGTACTTTTCCCCTAAGTTATAATATT
```

```
>Seq2 Second sequence
GGTGACACTTGATGCTGATGCCCTGAGTGAGTGGTGGATTA
```

Translation is the process by which a nucleotide sequence in RNA is translated into an amino acid sequence, forming a protein. This process occurs in triplet units called **codons**. Each codon encodes one amino acid or a stop signal for translation (stop codon). A DNA or RNA sequence can be read in three different configurations, depending on the starting point, known as **reading frames**:

- **frame 1**: reading starts from the first nucleotide;
- **frame 2**: reading starts from the second nucleotide;
- **frame 3**: reading starts from the third nucleotide.

DNA is double-stranded, and the strands are complementary and antiparallel, resulting in a total of **six reading frames**:

1. **frames 1-3**: reading on the leading strand in the 5' → 3' direction;
2. **frames 4-6**: reading on the complementary strand in the 5' → 3' direction.



Example of translation in Biopython:

```python
from Bio.Seq import Seq

dna_seq = Seq("ATGTACTAA")

protein = dna_seq.translate()
print(f"Frame 1: {protein}")
```

General Guidelines for Practical Tasks

The tasks are based on practical problems. Your goal is to find a solution using Python and the Biopython library, making use of available information resources. Remember that your instructor may ask you to explain a selected part of your code, so it is important to understand every element of it.

Software Requirements

To complete the tasks, you will need Python 3.x with the Biopython library. You can work in a local environment (Anaconda) or use Google Colab, which allows you to run Python code directly in your web browser.

Option 1 - Anaconda, local

Download and install Anaconda Distribution. Installing it for the current user (*Just Me*) does not require administrator privileges, provided that the computer's security settings allow it.

Open Anaconda Prompt and install the Biopython library using one of the following commands:

```
conda install -c conda-forge biopython
# or
pip install biopython
```

You can work in the **base** environment or create a separate environment for bioinformatics — you can find instructions at this link.

Option 2 - Google Colab, online

Alternatively, you can use Google Colab, which allows you to write and run Python code directly in your web browser without installing any software on your computer. A Google account is required to use Colab.

1. Open Google Colab and create a new notebook (*New notebook*).
2. In the first cell, enter and run the following command:

```
%pip install biopython
```

3. In the following cells, you can write and run Python code to solve the tasks.

Note: Files created in Google Colab are stored in a temporary environment by default. To download the generated FASTA file to your computer, use:

```
from google.colab import files
files.download("nazwa_pliku.fasta")
```

In both environments, you can use the same Biopython modules and install additional libraries needed to complete the tasks.

Task 1: Basics of Working with Biopython

Familiarize yourself with the Biopython documentation. In particular, the modules Seq, SeqIO, SeqUtils, Motifs may be helpful. Review the FASTA format looks like, as it is crucial in bioinformatics.

Part one

Create a nucleotide sequence generator in Python in the FASTA format with the following minimum requirements:

1. The script should generate a random DNA sequence.
2. Sequence length, ID and description (in this order) should be provided by the user using input() function.
3. The result should be written to a FASTA file named according to the inputted sequence ID. Additionally, ID and sequence description should be included in the FASTQ file header in the following format:
   `>{ID} {sequence description inputted by the user}`
   `{sequence generated by the script}`
4. The script should also calculate the sequence statistics: the percentage of each nucleotide occurrence and the ratio of C+G nucleotides to A+T nucleotides.
5. At a random position within the sequence, the script should insert the inputted name. However, make sure that the letters forming your name do not affect the DNA sequence statistics and are not counted toward its length.

Script interaction example:

```
Input sequence length: 20
Input sequence ID: A123
Input sequence description: Random test sequence
Input name: Mike

Sequence was saved to A123.fasta file
Sequence statistics:
A: 23.2%
C: 26.4%
G: 24.8%
```

```
T: 25.6%
CG/AT: 1.05
Inserted name position: 11
```

Example of what the file content might look like (in this case, the file will be named A123.fasta):

```
"""
>A123 Random test sequence
ACTGCCTGAAMikeACGACTGCCT
"""
```

Save the first version of the script and continue working on it in the same *.py file.

Part two- ADD FUNCTIONALITIES (at least 3):

You can continue working on your own or with the help of the AI. Add **at least three features** from the list below to the script**:**

Option A - "GC percent":
Add the option to generate a sequence with controlled GC content
- The user provides an additional input: **GC content (%)** in the range of 40–60
- The program generates a sequence close to the specified value (±5%)
- Display how closely the generated sequence matches the target

Option B - "Batch generator":
Extend the program to generate multiple sequences at once
- Add a prompt asking for the **number of sequences to generate**
- Save all sequences to a single file (a so-called multi-sequence FASTA)
- Wyświetl zbiorczą statystykę dla wszystkich sekwencji

Option C - "Quality Control":
Add sequence **validation and warnings**:
  - If the GC% is extremely low or high (outside the 30–70% range)
  - If there are nucleotide repeats longer than 4 (e.g., "AAAA")
- Add the option to regenerate the sequence if any warnings appear

Option D - "Colorful Output":
Add a **colourful output** in the terminal
- Colour: A=green, T=red, G=yellow, C=blue, imię=purple
- Add scripts ASCII art logo
- Add a progress bar during long sequence generation

Option E - "Additional data":
In addition to the name, hide the current date within the sequence
- Add a **decode** function that reads the generated FASTA file and finds all additional elements in the sequence (the name and the date)

Option F - "Format Pro":
Add line **wrapping for long sequences** at 60/80 characters (FASTA standard)
- Add metadata: `**##Generated: {date} | Tool: Python**`
- Add output file saving with an option to choose the export format: FASTA or JSON with full metadata

Prepare the final version of the script as a Python file (.py or .ipynb) that serves as the solution for this task.

Submit the file via MS Teams platform under the name **s11111_task1.py or s11111_task1.ipynb**, replacing s11111 with your student ID number.

Task 2: Sequence analysis and visualization with Biopython

For this task, use the provided **ls_orchid.fasta** file, which contains multiple DNA sequences.

Part one

Using the selected AI tool, create a nucleotide sequence generator in Python in the FASTA format with the following minimum requirements:

1. The program should read DNA sequences from the provided file (the file should be located in the same directory as the script — use a relative path).
2. The program should identify the starting positions of the following motifs: ATG, TATA, GAATTC. The output should list the positions of each motif in every sequence (in other words, the starting positions of each motif in each sequence, using Python's indexing convention).
3. Perform sequence translation. Translate each DNA sequence into a protein using all six reading frames.

Script output example:
```
Loaded sequence with id Seq_1:
ACATGCCTGAAACGAATTCTATATGGCCT
Motifs found:
ATG:[2,22]
TATA:[19]
GAATTC:[13]
Translacja:
Frame 1:TCLKRILYG
Frame 2:HA*NEFYMA
Frame 3:MPETNSIWP
Frame 4:RPYRIRFRH
Frame 5:GHIEFVSGM
Frame 6:AI*NSFQAC
```

Part two- ADD FUNCTIONALITIES (all 3):

You can continue working on your own or with the help of the same AI. Add **all three features** from the list below to the script**:**

Option A - "Sequence charts":
Create charts showing specific information for the first 30 sequences in the input file:
- Bar chart or area plot: nucleotides distribution
- Heatmap: nucleotide type by position
- Line plot: % GC at each position across all analyzed sequences
Save each chart to file (JPEG or PNG)

Option B - "CSV report":
Prepare **a data frame** with subsequent information:
- sequence ID,
- count of each nucleotide,
- GC content (%),
- motifs positions,
- first 10 nucleotides of the reverse complement sequence,
- length of the translated proteins.
Save the data to the CSV file.

Option C - "Sequence rating"
Create your **own scoring system** to evaluate the sequences. Criteria:
a. Most balanced (the smallest difference between nucleotide contents)
b. Most interesting (your own definition + a custom scoring() function). Describe the criterion you chose in a comment within the script.
For each criterion, display the **TOP 3 sequences and their score** and save them to a FASTA file (let the score be included within the FASTA description, use the filenames: criterion_a.fasta and criterion_b.fasta).

Prepare the final version of the script as a Python file (.py or .ipynb) that serves as the solution for this task.

Submit the file via MS Teams platform under the name **s11111_task1.py or s11111_task1.ipynb**, replacing s11111 with your student ID number.

Task 3: Working with LLM

Evaluate your experience working with AI tools while completing Tasks 1 and 2.
Prepare a short report by answering the following questions (it's best to copy the questions below into a new Google document and fill in your answers there):

Part one: General experience

How often did you use AI during your work?

- ☐ At every step
- ☐ For generating the initial (skeleton) code, then only when you encountered a problem
- ☐ For generating the initial (skeleton) code, then worked completely independently
- ☐ Other. describe how _____

Part two: Specific applications

What exactly did you use AI for? (select all that apply)

- ☐ Generating the entire code from scratch
- ☐ Explaining Biopython syntax
- ☐ Debugging errors
- ☐ Code optimization
- ☐ Creating visualizations
- ☐ Helping to understand biological concepts
- ☐ Other. Describe _____

Provide 1–2 specific examples where AI was most helpful.

- Describe the problem and how AI helped to solve it.
- Did the solution work immediately, or did it require modifications?

Provide 1–2 examples where AI failed or misled you.

- What went wrong?
- How did you solve the problem?

Część 3: Pros and cons

Main ADVANTAGES of working with AI in this context:

- List at least 3 advantages
- Justify each (1–2 sentences)

Main DISADVANTAGES or LIMITATIONS of AI:

- List at least 3 disadvantages/problems
- Justify each (1–2 sentences)

Submit your **report** via the MS Teams platform in **DOCX or PDF format**, using the filename **s11111_task3**, replacing s11111 with your student ID number.