**BT-3172: Special Topics in Bioinformatics**

|  |
| --- |
| **Lab 4: Python functions and object-oriented programming concepts for bioinformatics**  **Name: Anushka Udara**  **Index number : s14234**  *Prepared by: Dr. Pasan Fernando* |

In this practical you will learn how to write custom methods and use Object-oriented programming (OOP) concepts in Python to solve biological problems.

For this practical, you will be working with several genes and proteins involved with the DREB pathway. This is an ABA-independent pathway, which is important in plant abiotic stress response.

After using PyCharm to write your scripts, copy the codes to the appropriate space below the questions. Also, submit the Python files separately so they can be tested. Use the following format to name the script: YourIndexNo\_PrimaryQuestionNo\_SecondaryQuestionNo.py.

**Please write the source codes and outputs of your programs in this schedule under the correct question number.**

1. Writing custom Python methods to analyze DNA sequences
   1. Use the UniProt KnowledgeBase to search for the following proteins in rice japonica subspecies: DREB1A, DREB1B, DREB2A, and DREB2B.isoform 1. Write their UniProt identifier numbers in front of the protein names below.

Mention their reviewed status in front of the ID. Access their amino acid sequences in FASTA format. Obtain the RefSeq gene entry record for each protein and write their RefSeq gene IDs in front of the correct record. Then locate their mRNA sequences and access **only the coding sequence** for each entry in FASTA format.

Create an empty FASTA file named “OSDREB\_sequences.FASTA” and copy the above amino acid sequences and coding sequences one after another. Use the following header format to name the FASTA header for each entry.

For proteins: >Gene\_name**\_P**- RefSeq\_gene\_ID-species-subspecies-Uniprot\_ID-reviewed\_status

For coding sequences: >Gene\_name**\_CDS**-RefSeq\_gene\_ID-species-subspecies

# DREB1A- XP\_015610912.1 – Oryza sativa – japonica - Q64MA1- reviewed

# DREB1A- XM\_015755426.2 - Oryza sativa – japonica

# DREB1B XP\_015610928.2 Orysa\_sativa japonica Q3T5N4 reviewed

# DREB1B XM\_015755442.2 Orysa\_sativa japonica

# DREB2A XP\_015621339.1 Orysa\_sativa japonica Q0JQF7 reviewed

# DREB2A XM\_015765853.1 Orysa\_sativa japonica

# DREB2B.isoform\_1 Q5W6R4 Orysa\_sativa japonica Q5W6R4 reviewed

# DREB2B.isoform\_1 [NM\_001402517](https://www.ncbi.nlm.nih.gov/nuccore/NM_001402517). Orysa\_sativa japonica

* 1. Write the equation and a **pseudocode** of an algorithm to calculate the AT content of a given DNA sequence.

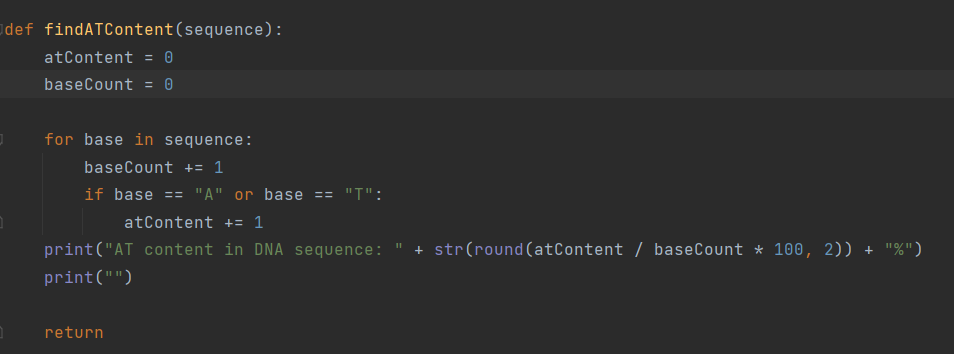
AT content = AT count / base count \* 100

Loop through the given DNA sequence using a for loop.

Increment the baseCount variable for every base read.

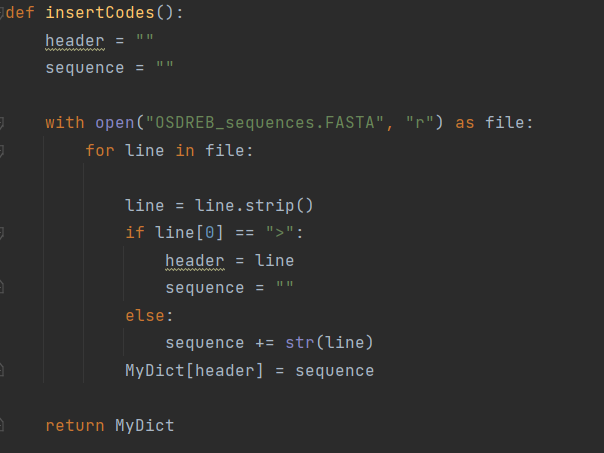
If the base equals to Adenine or Thymine, increment the atCount variable.

Print the AT content.

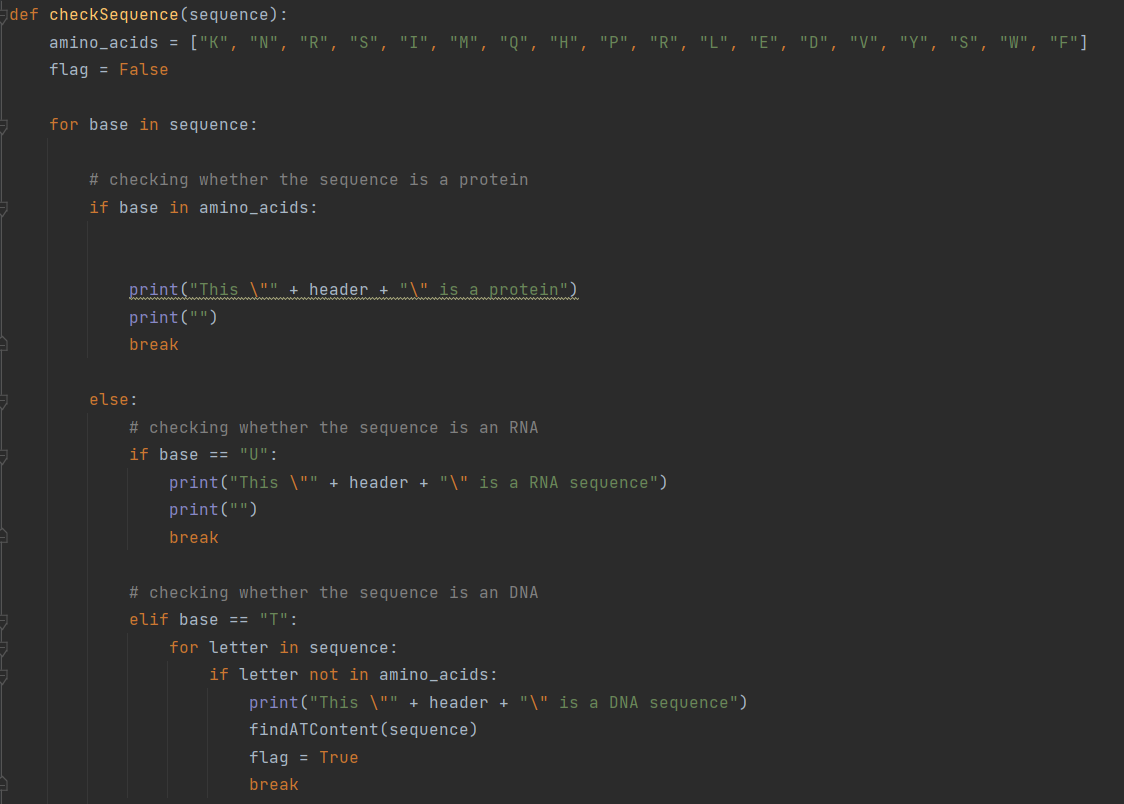


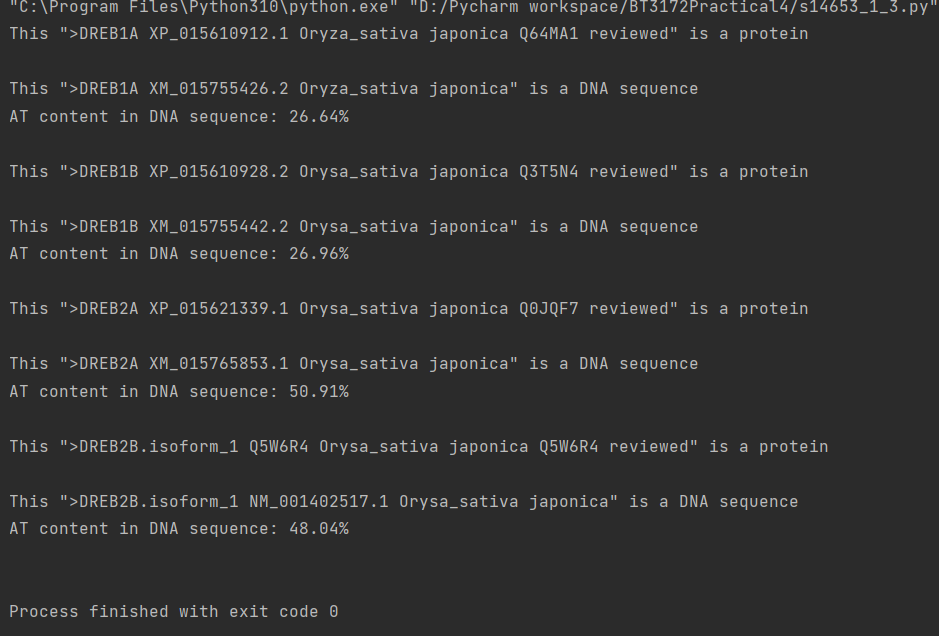


* 1. Implement the above algorithm as a custom method in Python. For the sub questions: III, IV, V and VI, use a single Python script to write the 3 methods and the implementation. You can save it only using the main question number.
  2. Write a custom Python method to split multiple FASTA sequences in a single text file and return a dictionary containing sequence headers as keys and the sequences as values.



* 1. Write a custom Python method to check whether the given sequence type is DNA, mRNA or amino acid sequence.



* 1. Use the above written methods to check each sequence in the OSDREB\_sequences.FASTA file and print the AT content.

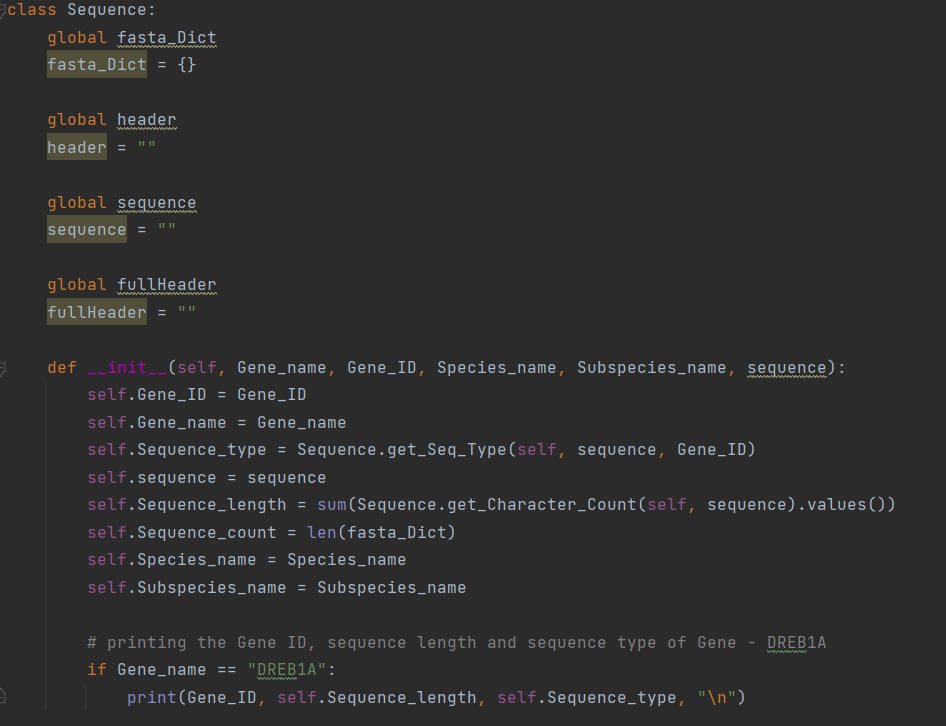
1. Familiarizing with Python OOP techniques. Writing a Sequence class. Use the same Python script to write the class for sub questions I and II.
   1. Write a Python Sequence class to store any biological sequence (DNA, mRNA, and amino acid sequences). It should have the following attributes and methods:

Attributes

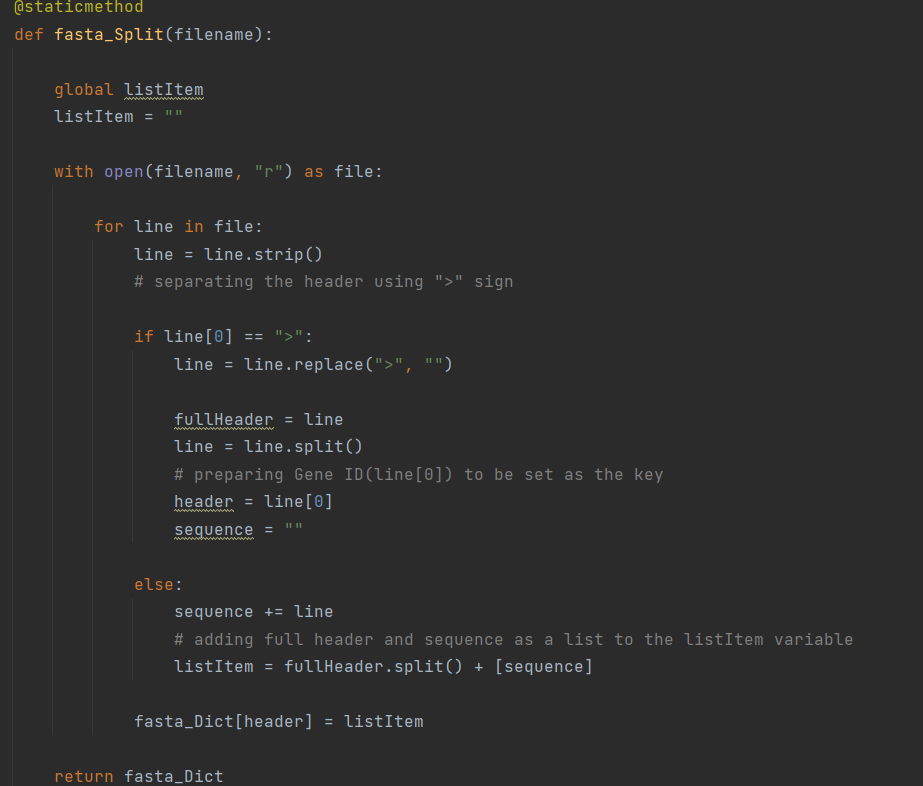
* + 1. Gene ID.
    2. Gene name.
    3. Sequence type
    4. Sequence length.
    5. Sequence count (to count the number of sequences created by the sequence class).
    6. Species name.
    7. Subspecies name

Methods

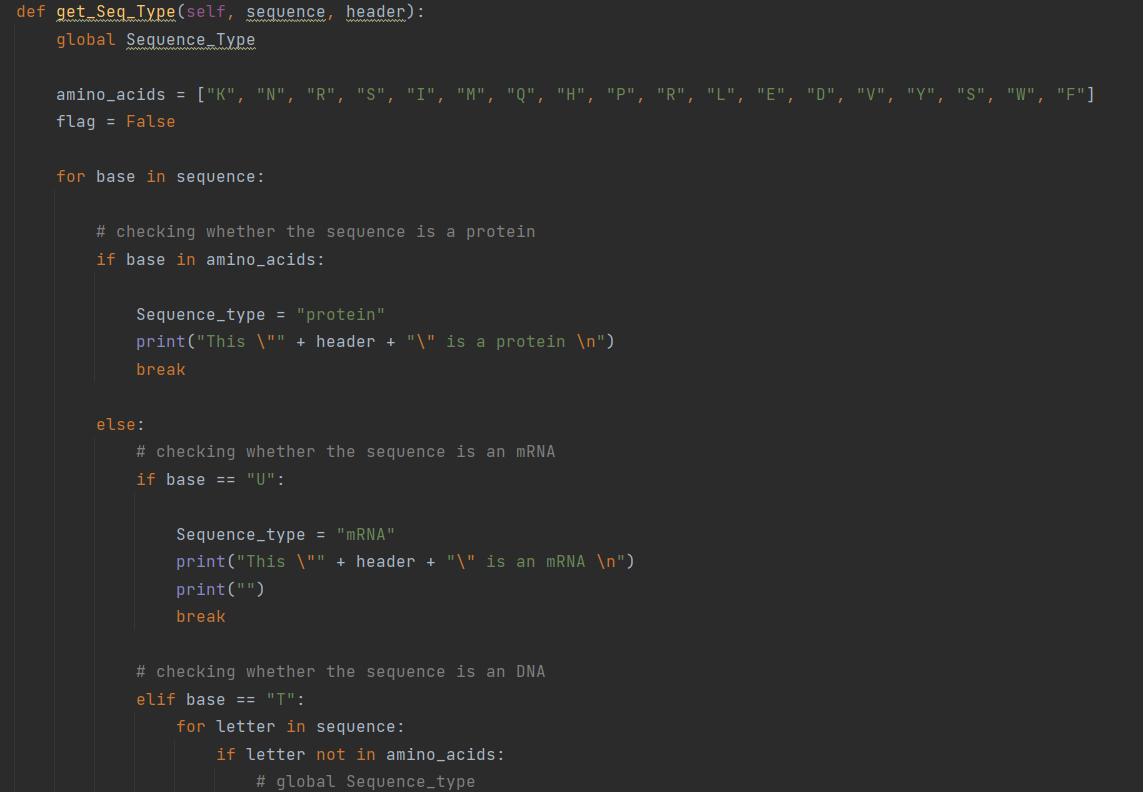
* + 1. Constructer method to create sequence objects

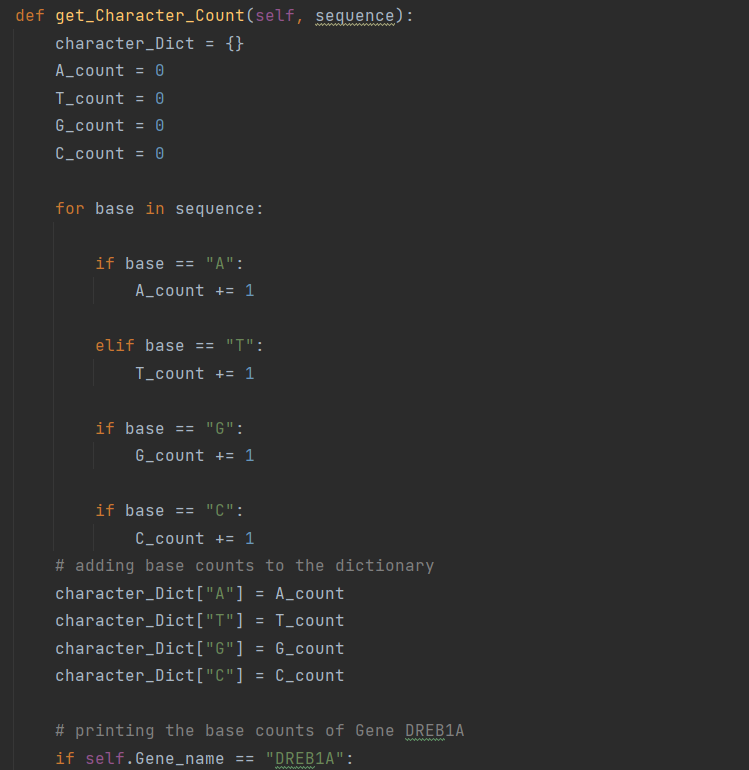


* + 1. fasta\_Split(): a static method to split multiple FASTA sequences in a single text file and return a dictionary containing the Gene name (first item in the hyphen-separated list) as the key and a **list** containing hyphen-separated fields in the header plus the sequence as value. Make sure the **gene name (key) is also included in the value list** (refer to 1.IV and modify the code accordingly).



* + 1. get\_Seq\_Type() A method to check the sequence type (refer to 1.V), but this time, distinguish between all 3 sequence types: DNA, mRNA, amino acid.

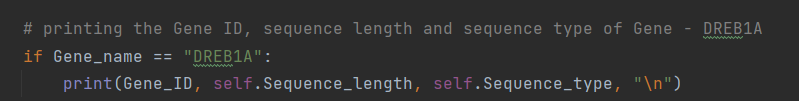


* + 1. get\_Character\_Count(): this should return a dictionary of character counts with each character as the key and count as the value. A character can be a nucleobase or an amino acid.

**Create a copy of the “OSDREB\_sequences.FASTA” file you created during question 1(I) but remove all the amino acid sequences and keep only the coding sequences.** Then, use this file to write a Python program to read the sequences in and create objects for each FASTA record. Moreover, perform the tasks mentioned in question II and III using the Sequence class. You can write this script in a separate file and import the Sequence class or use the same Python file with the Sequence class. When creating objects, you can manually type each parameter for the object necessary for running the following commands or you can pass a list of elements as parameters using the following command. Use sequence name as the object name.

Object\_name = Class\_name(\*[a list of parameters to be passed in the correct order])

* 1. Print the following details for the DREB1A DNA sequence: Gene ID, sequence length, and sequence type.





* 1. Using the Sequence class output the base count of the four bases of the DREB1A coding sequence.

