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

**Lab 5: Advanced object-oriented programming concepts for bioinformatics.**

**Name: Anushka Udara**

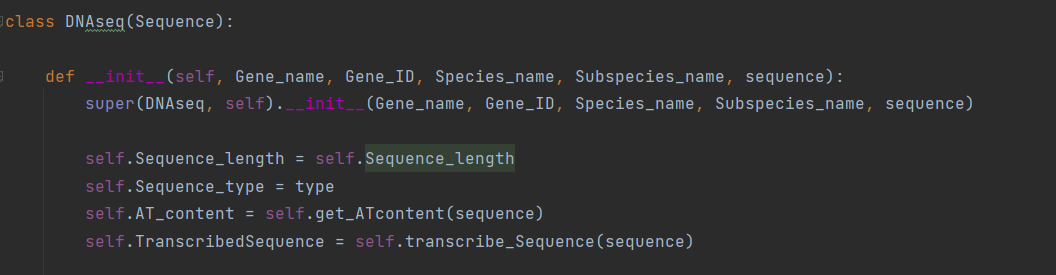
**Index number: s14234**

In this practical you will extend the Sequence class that you wrote in the previous lab to include subclasses. You will learn how to use inheritance and polymorphism when writing these subclasses.

Use PyCharm to write the scripts in **the same Species class** you have written in the previous lab. Write the implementation in the same python file or in a separate file. Save the file including your index number in the python file and upload it to the LMS with the answers. When writing the answers in this sheet, copy and paste only the necessary codes under the question number. No need to include the parent Sequence class in this sheet; it should be only included in the Python file.

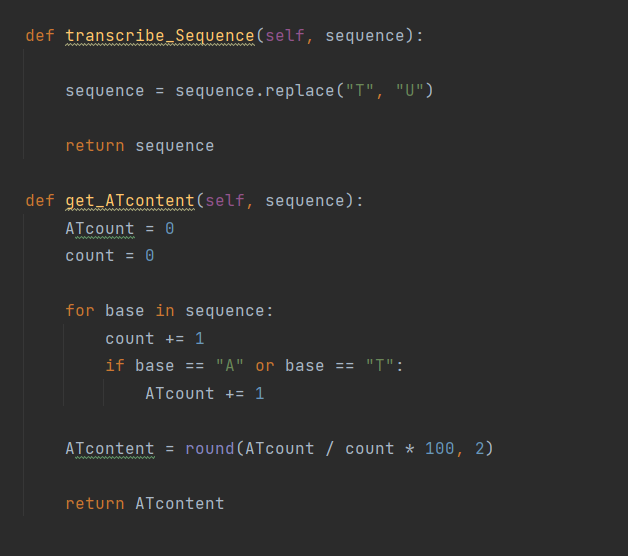
1. In the previous lab, you wrote a general Sequence class to represent all sequences. In this exercise, you will write subclasses for DNA, mRNA, and amino acid sequences and use these subclasses when creating new objects.
   1. Write a subclass of the Sequence class for DNA sequences named “DNAseq”. It should have the following unique/additional attributes and methods.

Attributes

* + 1. AT\_content.
    2. Transcribed sequence

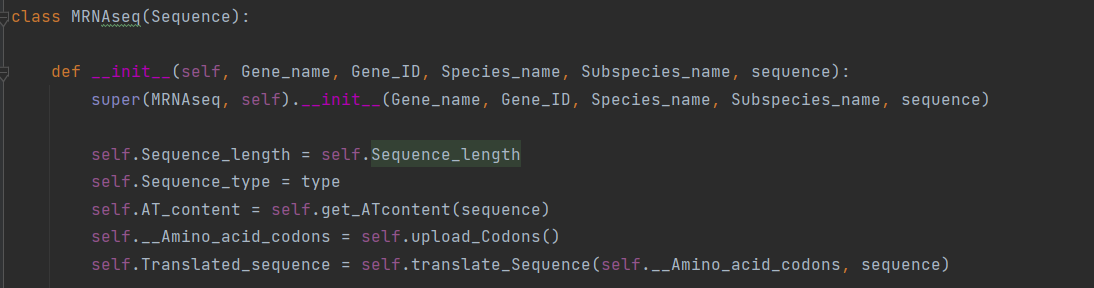
Methods

* + 1. Constructer method to create DNA sequence objects
    2. transcribe\_Sequence(): transcribe the given DNA sequence into its mRNA sequence and store it in the Transcribed sequence instance variable.
    3. get\_ATcontent(): this should return the AT content of the given sequence and update the AT\_content instance variable.

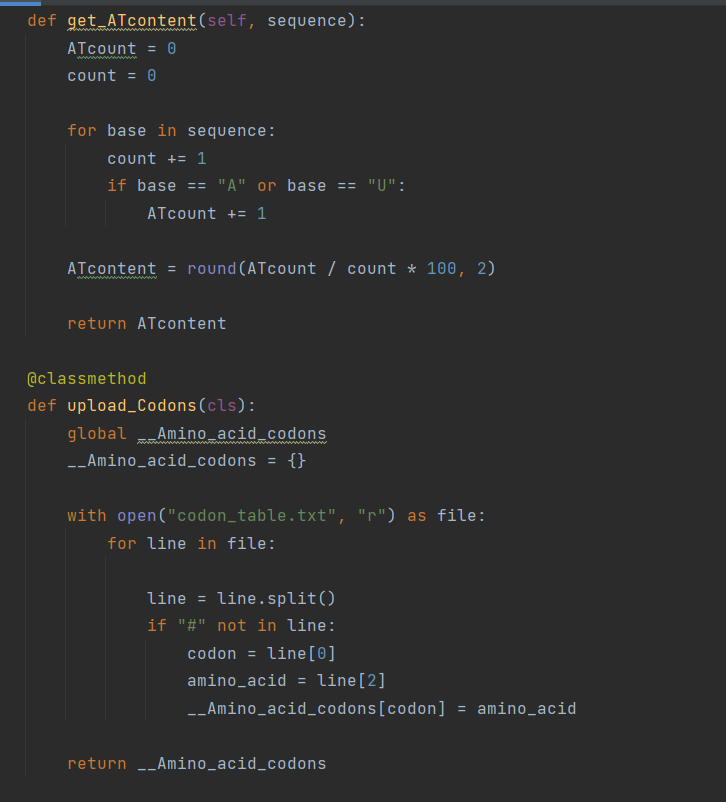


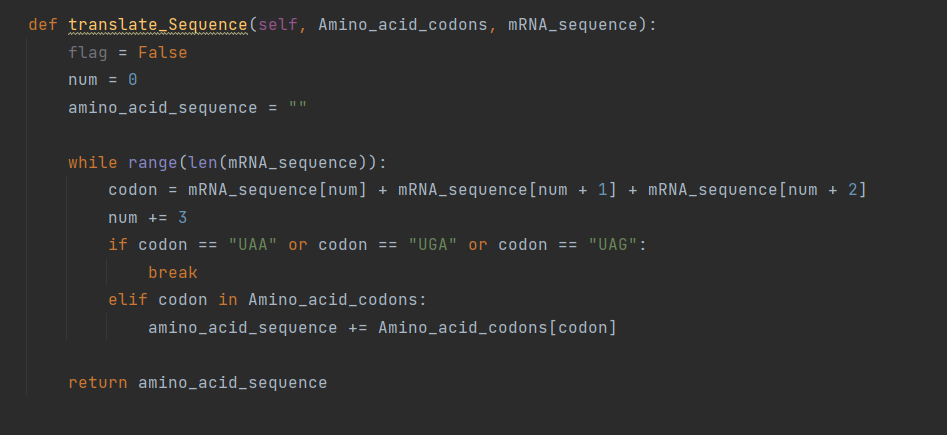
* 1. Write a subclass of the Sequence class for mRNA sequences named “MRNAseq”. It should have the following unique/additional attributes and methods.

Attributes

* + 1. AT\_content.
    2. Amino\_acid\_codons (This should be a dictionary and should be hidden/private)
    3. Translated\_sequence

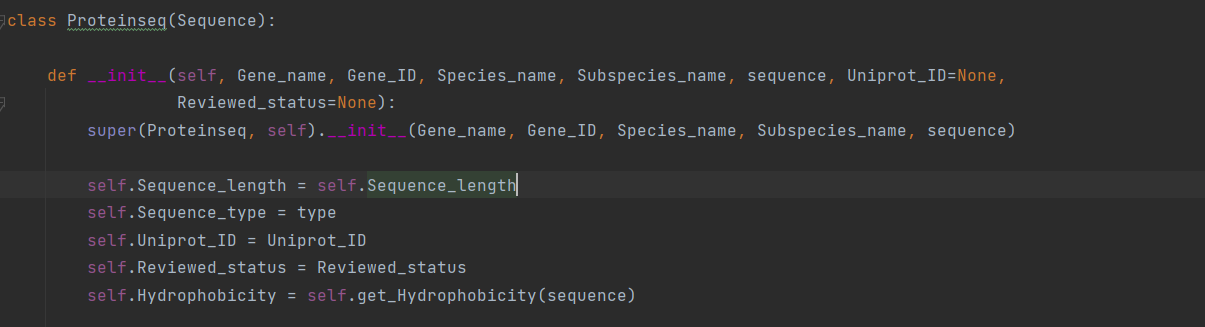
Methods

* + 1. Constructer method to create DNA sequence objects
    2. get\_ATcontent(): this should return the AT content of the given sequence and update the AT\_content object variable. Because this is for mRNA sequences, rethink the way you should write this method.
    3. upload\_Codons(): This class method should store codon-amino acid pairs from a text file into the Amino\_acid\_codons variable.
    4. translate\_Sequence(): translate a given mRNA sequence into its amino acid sequence and return the translated sequence.

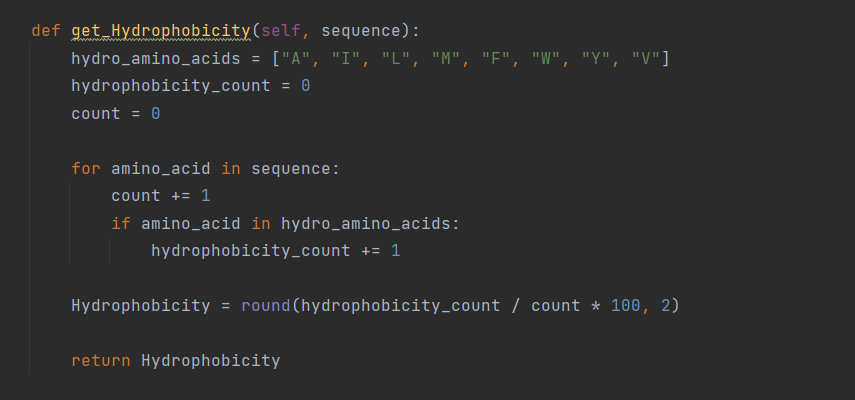


* 1. Write a subclass of the Sequence class for protein sequences named “Proteinseq”. It should have the following unique/additional attributes and methods.

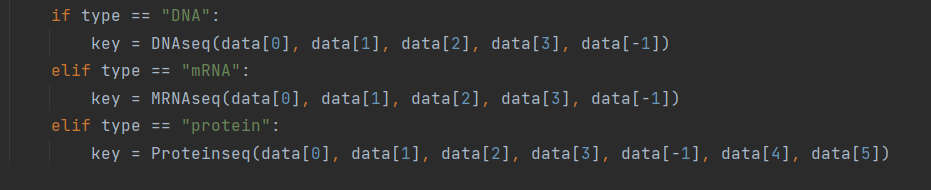
Attributes

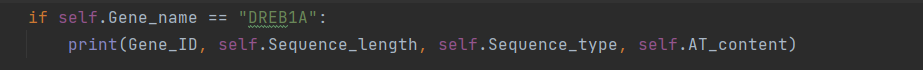
* + 1. Uniprot\_ID
    2. Reviewed\_status
    3. Hydrophobicity

Methods

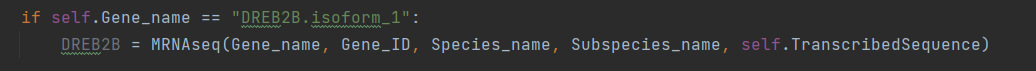
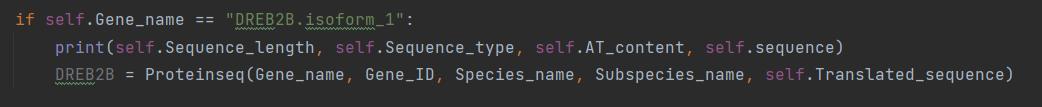
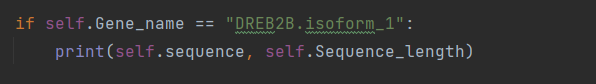
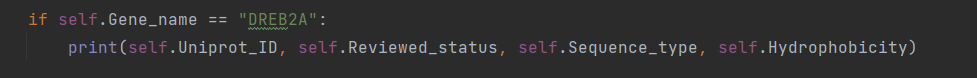
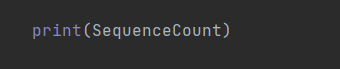
* + 1. get\_Hydrophobicity(): this should return the percentage of the total hydrophobic amino acid residues (A, I, L, M, F, W, Y, V) in the sequence.

1. Write a Python program to read the sequences in OSDREB\_sequences.FASTA file you generated during previous lab question 1(I) and create objects for each FASTA record. Moreover, perform the following tasks using the Sequence class and the subclasses. You can write this script in a separate file and import the Sequence class to perform the tasks or implement in the same file. When creating objects, you can manually type each parameter/argument for the object 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 OSDREB1A DNA sequence: Gene ID, sequence length, sequence type and AT content. Pay special attention to the class that you use to create the object.



1. Transcribe the OSDREB2B coding sequence and create a new object for the resulting mRNA sequence. Print the length and sequence type, AT content, and the sequence of the resulting mRNA sequence.
2. Translate the OSDREB2B mRNA sequence created above into its amino acid sequence and print the result and also print its length.
3. Print the Uniprot ID, reviewed status, type, amino acid composition and the Hydrophobicity of DREB2A protein.
4. Output the number of sequences created using the Sequence class variable.
5. Explain how encapsulation and method polymorphism are used in the subclasses that you wrote above.

Encapsulation is the hiding of unnecessary data from the user and other classes. They can only be accessed by only through the methods of their current class. Encapsulation is achieved in MRNAseq class by using a private dictionary variable. It is declared by adding two underscores in front of variable names. In this case amino\_acid\_codons is a private dictionary.

Method polymorphism is the ability of methods named by the same name, to do different things based on the particular object that they are working on. In this exercise method polymorphism is achieved through having two get\_ATcontent() methods in both MRNAseq and DNAseq sub classes. In both methods, they calculate the AT content of mRNA and DNA sequences. But, based on the object by which they are called, they act on two different type of sequences, calculate the AT content differently.