BT-3172 - Special Topics in Bioinformatics
Computing for Biologists
Lab 2 - Use of Python data structures and control structures in bioinformatics

W.M. Ayesha Sanahari | 2017s16470 | s13722

In this practical you will learn how to use Python data structures and control structures to solve biological problems.

DREB1A (Dehydration-responsive element-binding protein 1A) is an important protein which is expressed in response to abiotic stresses such as drought stress and salinity stress in plants. It is important to study this protein and its involved biological pathways when studying plant abiotic stress responses. During this practical, you will work with DREB1A protein in rice.

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.

1) Using control structures to differentiate between DNA and amino acid sequences

   I. Use the UniProt knowledgebase to search for DREB1A protein in rice subspecies japonica. Write its UniProt identifier number below. Is the protein reviewed by the knowledgebase? What is the importance of this reviewed status in the UniProt knowledgebase?

   UniProt identifier number – Q64MA1
   Yes. It is Reviewed.
   importance of this reviewed status – It includes high quality, manually annotated and non-redundant protein sequences in UniprotKB/Swiss-Prot. So manually annotation consists of a critical review of experimentally proven about each protein including the protein sequences. Data are continuously updated by an expert team of biologists.

   II. Now download the FASTA amino acid sequence for the DREB1A protein in rice and rename it to "OSDREB1A.fasta". Furthermore, obtain the RefSeq gene sequence record for the DREB1A protein. Make sure you obtain the most updated RefSeq record. Write the RefSeq gene ID below. Locate the mRNA sequence record for this gene and write its accession number (with version included). Obtain its FASTA record and copy and paste it below the FASTA record of the amino acid sequence in the previously downloaded OSDREB1A.fasta file. Shorten the FASTA headers to only keep their identifiers.

   RefSeq gene ID – 4347620
   Ac. No. of mRNA sequence – XM_015755426.2

   III. Write an algorithm to differentiate between the two FASTA records in OSDREB1A.fasta file. The algorithm must correctly identify the mRNA record from the amino acid sequence and replace its Thymine ("T") bases with Uracil ("U") bases. Then it should save the resulting transcribed mRNA sequence in a new FASTA file called "OSDREB1A_mRNA.fasta". The FASTA header of this file should keep the header of the original mRNA record with the word: "transcribed" added to the end.

```
Input: A FASTA file with two fasta records
Output: FASTA file containing only transcribed mRNA sequence

Define an empty dictionary to store the fasta sequences
Define an empty string variable to store the sequence
open/read fasta file (with 2 fasta records) & store it in a variable
for line in sequenceFile
    Remove blank lines
    if ('>') in line
        seperate the headers with '>' as the keys in the dictionary
    else
        if 'M' in line
            select the protein sequence as the value
        if 'M' not in line
            select only the mRNA sequence as the value and replace 'T' to 'U' only in
            mRNA sequence

open a file to write the transcribed mRNA sequence
select only the transcribed mRNA sequence from the dictionary
add transcribed to the end of the mRNA header
write the transcribed mRNA sequence and the new header to a new fasta file
print the content to be written in the file (header with transcribed mRNA sequence)
close the files
```

IV. Implement the above algorithm in Python.

```python
"""
Author : Ayesha Sanahari
date : 27/Nov/2020
Differentiate two FASTA records & replace 'T' with 'U' in the mRNA sequence
Input: A FASTA file with two fasta records
Output: FASTA file containing only transcribed mRNA sequence
"""

# Define an empty dictionary to store the fasta sequences
seq_dict = {}
# Define an empty string variable to store the sequence
sequence =''

# open/read fasta file (with 2 fasta records) & store it in a variable
with open('OSDREB1A.fasta','r') as sequenceFile:
    for line in sequenceFile:

        # Remove blank lines
        if line != '\n':
            line = line.strip()

            # seperate the headers with '>' as the keys in the dictionary
            if ('>') in line:
                header = line
                seq_dict[header]=''

            else:
                sequence = sequence + line

                # select the protein sequence as the value
                if 'M' in line:
                    seq_dict[header] = sequence
                # select only the mRNA sequence as the value and replace 'T' to 'U' only in mRNA sequence
                else:
                    seq_dict[header] += line.strip().replace('T','U')

    print(seq_dict)

    # open a file to write the transcribed mRNA sequence
    with open("OSDREB1A_mRNA.fasta",'w') as output:
        # select only the transcribed mRNA sequence
        for item in seq_dict:
            if 'mRNA' in item :
                # add transcribed to the end of the mRNA header
                header1= item +' transcribed'

        # write the transcribed mRNA sequence to a new fasta file
        output = output.write('>'+ header1 + '\n' +seq_dict[item])
        # print the content to be written in the file (header with transcribed mRNA sequence)
        print(('>'+ header1 + '\n' +seq_dict[item]))
```

2) Translating mRNA sequences using a Python script.

I. Write an algorithm to translate a given mRNA sequence (with Uracil bases instead of Thymine) to its amino acid sequence. For this algorithm, ignore the different open reading frames and begin the translation from the first letter. You have to use a while loop when writing the algorithm. Use the amino acid codons table text file to create a dictionary to store codon to amino acid mappings. End the translation when you hit a stop codon.

```
Input: transcribed mRNA sequence fasta file and codon_table
Output: Amino acid sequence

Define an empty dictionary to store codons & amino acid letters
Open/read codon_table and store it in a variable
for line in Codon_table:
    Remove header and empty lines in Codon table
    make a dictionary from codons as key and amino acid letters as values

Open /read mRNA and store in a variable
for line in sequence:
    Remove empty lines
    remove header and select only the sequence
    get header to write in protein file
```

```
Define the position of the sequence
iterate through the sequence
while position  <= len(seq):
    divide the sequence into codons by three bases assuming 1st reading frame
    go through the dictionary and find relevant amino acid letters
    increment the position by 3
get the amino acid sequence

Define start codon & end codons
Find the position of aa relevant to start codon and end codon
Assuming translation is starting with Methionine, get the translated protein seq. and length
Assuming translation is starting with first letter, get the translated protein seq. and length

write the translated protein into a fasta file
```

II. Implement the above algorithm in python using OSDREB1A_mRNA.fasta file as the example. Write the translated sequence in an output file in FASTA format

```
"""
Author : Ayesha Sanahari
date : 27/Nov/2020
Translate mRNA sequence into an Amino acid sequence
Input: transcribed mRNA sequence fasta file and codon_table
Output: Amino acid sequence
"""
#Define an empty dictionary to store codons & amino acid letters
Codon_map = {}
# Open/read codon_table and store it in a variable
with open('codon_table.txt', 'r') as Codon_table:
    for line in Codon_table:

        # Remove header and empty lines in Codon table
        if '#' not in line and line != '\n':
            (codon, amino_acid, Letter, FullName) = line.strip().split('\t')
            # make a dictionary from codons as key and amino acid letters as values
            Codon_map[codon] = Letter
print(Codon_map)
protein = ''
seq = ''
Header = ''

# Open/ read mRNA and store in a variable
with open('OSDREB1A_mRNA.fasta', 'r') as sequence:
    for line in sequence:
        # Remove empty lines
        if line != '\n':
            line = line.strip()
            # remove header
            if '>' not in line:
```

```
            seq = seq + line
        # get header to write in protein file
        if '>' in line:
            Header = Header + line
            Header = Header.replace('mRNA transcribed','translated amino acid sequence')

print(seq)
print("Length of transcribed mRNA seq: ", len(seq),"bp")

# Define the position of the sequence
position = 0
# iterate through the sequence
while position  <= len(seq):
    #divide the sequence into codons by three bases assuming 1st reading frame
    codon1 = seq[position:position+3]

    # go through the dictionary and find relevant amino acid letters
    if codon1 in Codon_map.keys():
        protein += Codon_map[codon1]

    position = position + 3

print(protein)
print("Length of amino acid seq: ",len(protein),'aa')

# Find the position of aa relevant to start codon and end codon
AA_No_BeforeMeth= protein.find('M')
AA_No_Before_O= protein.find('O')
print("No. of amino acids before Methionine: ", AA_No_BeforeMeth)
print("No. of amino acids before the first stop codon: ", AA_No_Before_O)

# Assuming translation is starting with Methionine, get the translated protein seq. and length
protein1 = protein[AA_No_BeforeMeth:AA_No_Before_O]
print(protein1)
print("Assuming translation is starting with Methionine",'\n' ,"Length of translated amino acid sequence
is: ", len(protein1), "aa")


# Assuming translation is starting with first letter, get the translated protein seq. and length
print(Header)
print(protein[:AA_No_Before_O])
print("Assuming translation is starting with first letter",'\n' ,"Length of translated amino acid sequence
is: ", len(protein[:AA_No_Before_O]), "aa")

# write the translated protein into a fasta file
with open("Translated_mRNA.fasta", 'w') as output:
    output = output.write(Header + '\n' + protein[:AA_No_Before_O])
```

III. What is the length of the translated amino acid sequence?

```
Assuming translation is starting with first letter, Length of translated amino acid sequence is:  230 aa
Assuming translation is starting with Methionine, Length of translated amino acid sequence is:  89 aa
```

3) Using the graph data structure in Python to calculate the degree of a given protein. For this, you will need the NetworkX Python package installed.

   I. First find protein-protein interaction data using the rice DREB1A Uniprot record. Write the STRING database identifier for this protein below.

      STRING database identifier - 4530.OS09T0522200-01

   II. Go to the STRING record of the rice DREB1A protein. How many other proteins does it interact with according to the visualization? Now increase the maximum number of interactions to show to 100.

      With 10 proteins

   III. Now, write a Python program to calculate the degree of the rice DREB1A protein. You have to use the NetworkX module for the calculation. First, download the interaction data in tabular format. Use the downloaded TSV file for the calculation. Write the resulting degree of the program.

```
"""
Author : Ayesha Sanahari
date : 01/Dec/2020
Find the degree of DREB1A protein using networkx package
Input: string_interactions.tsv file containing information about protein interactions
Output: degree of DREB1A protein
"""

import networkx as nx;

# Define an empty dictionary to store interactions
dict={}
# define the key of dictionary
num=1
#open/read file and store in a variable
with open('string_interactions.tsv', 'r') as file:
    for line in file:
        if '\n' != line and 'node1' not in line:
            interaction = line.strip().split('\t')
            dict[num]= interaction[0:2]
            num +=1
print(dict)

G = nx.Graph()
for key, values in dict.items():
    G.add_edge(values[0], values[1])

print("number_of_edges: " ,G.number_of_edges())
print("degree of the rice DREB1A protein: ", G.degree['ERF24'])
```

```
degree of the rice DREB1A protein:  62
```