MSc Applied Bioinformatics
alexey.larionov@cranfield.ac.uk

Practical 4: Using Python dictionaries

This practical session is based on Lecture 5. It gives you an opportunity to implement Python dictionaries in bioinformatics context.

Task 1: Counting frequency of amino acids using dictionary

In Practical 3, we counted nucleobases in a DNA sequence using flow control statements. Now we will do the same using dictionaries. However, this time we will count the number of amino acids in a protein sequence (see Table 2 in Practical 2 for amino acid codes).

Given the protein sequence below we will (i) store it in a dictionary, (ii) count how many times each amino acid appears in the protein, and finally (iii) print out the results: how many amino-acids are present, the most frequent, the least frequent and the frequency of all of them in alphabetical order.

Make a new project folder for this Practical session. Start a new script or notebook, call it ***counting_amino_acid.py / ipynb.*** You may use any available Python interpreter without creating a virtual environment because we will not import any external modules.

Add the header (e.g. author name and date, and a sentence describing script purpose), then enter the protein sequence:

```
protein =
"""MGNAAAAKKGSEQESVKEFLAKAKEDFLKKWENPAQNTAHLDQFERIKTLGTGSFGRVML
VKHMETGNHYAMKILDKQKVVKLKQIEHTLNEKRILQAVNFPFLVKLEFSFKDNSNLYMV
MEYVPGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKPENLLIDQQGY
IQVTDFGFAKRVKGRTWTLCGTPEYLAPEIILSKGYNKAVDWWALGVLIYEMAAGYPPFF
ADQPIQIYEKIVSGKVRFPSHFSSDLKDLLRNLLQVDLTKRFGNLKNGVNDIKNHKWFAT
TDWIAIYQRKVEAPFIPKFKGPGDTSNFDDYEEEEIRVSINEKCGKEFSEF"""
```

Q. Can you spot a difference in the way how we define string for protein here and how we defined strings for DNA in Practical 3?

The next step is needed do address one of the formatting features of the source data:

```
protein_oneline = protein.replace("\n", "") # remove new line (EOL)
                                            # characters from the string
```

Explore the result. If you copy-pasted the protein sequence from the PDF file provided to you, it could happen that the original EOL-s had changed to something else. If this happened in your case, remove the other character(s), instead of EOL shown above.

Now, let's define an empty dictionary. Later, this dictionary will be used to contain amino acids as keys and frequencies as values:

```
amino_acids = dict() # you can also use amino_acids = {}
```

Write a loop that will iterate over protein sequence and store the information in the dictionary:

```
for aa in protein_oneline:
     if aa in amino_acids:
          amino_acids[aa] = amino_acids [aa] + 1
     else:
          amino_acids[aa] = 1
```

Q. What does “amino_acids” contain now? Try to print it, what are the keys and what are the values?

Now we have stored the information about amino acids and their frequencies in the dictionary, the length of the dictionary is equal to the total number of amino acids. Let’s print it:

```
num_aminos = len(amino_acids) # number of different amino acids
                              # detected in the sequence

print ("The number of different amino-acids is ", num_aminos)
```

Then we will get all amino acids using the **`<dictionary>.keys()`** function. The keys are not sorted so we will sort them alphabetically:

```
aa_keys = list (amino_acids.keys())
aa_keys.sort()
```

Now two more dictionaries for the most frequent and the least frequent amino acids:

```
mostF = {"frequency" : 0, "aminoacid" : "-"}
leastF = {"frequency" : len(protein_oneline), "aminoacid" : "-"}
```

Iterate through the dictionary to identify most and least frequent amino acids. Also, print the frequency for each amino acid:

```
for aa in aa_keys:

          freq = amino_acids[aa]

          if (mostF["frequency"] < freq):
               mostF ["frequency"] = freq
               mostF ["aminoacid"] = aa

          if (leastF["frequency"] > freq):
               leastF ["frequency"] = freq
               leastF ["aminoacid"] = aa

          print(aa, " is present", freq, "times")
```

Now print the least and most frequent amino acids:

```
print("Amino acid", leastF["aminoacid"], "has the lowest frequency
(",leastF["frequency"],")")

print("Amino acid", mostF["aminoacid"], "has the highest frequency
(",mostF["frequency"],")")
```

Save and run the entire script / notebook.

Exercise 4.1. *Given the following sequence of DNA:*

DNA = "GATTACATATATCAGTACAGATATATACGCGCGGGCTTACTATTAAAAACCCC"

Create a dictionary representing an index of all possible dimers (i.e. 2 bases), 16 dimers in total: AA, AT, AC, AG, TA, TT, TC, TG, ... The keys of the dictionary should be the dimers and the values should be their frequencies.

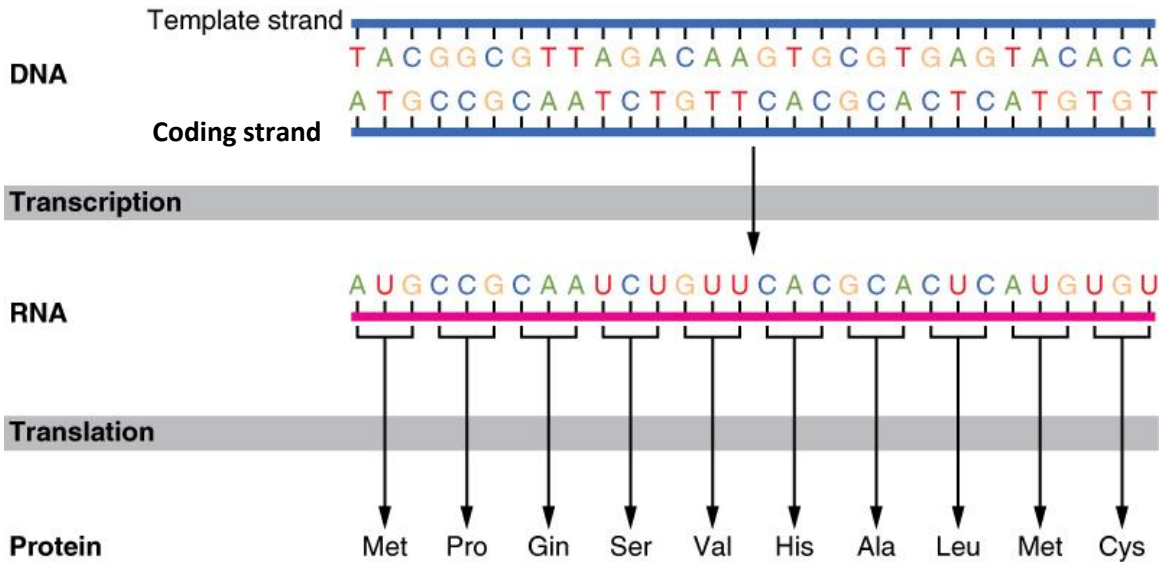
Additionally, print the first position for each dimer:

```
The first position of AA occurred at: 5
...
```

Task 2: From DNA to Protein

We are going to calculate what protein is coded by a DNA sequence following the central dogma of molecular biology:

DNA → RNA → Protein



Central dogma of molecular biology

Add a new script or notebook to your project, give it a meaningful name, add the usual header.

Let's start with the sequence of ***template*** DNA strand:

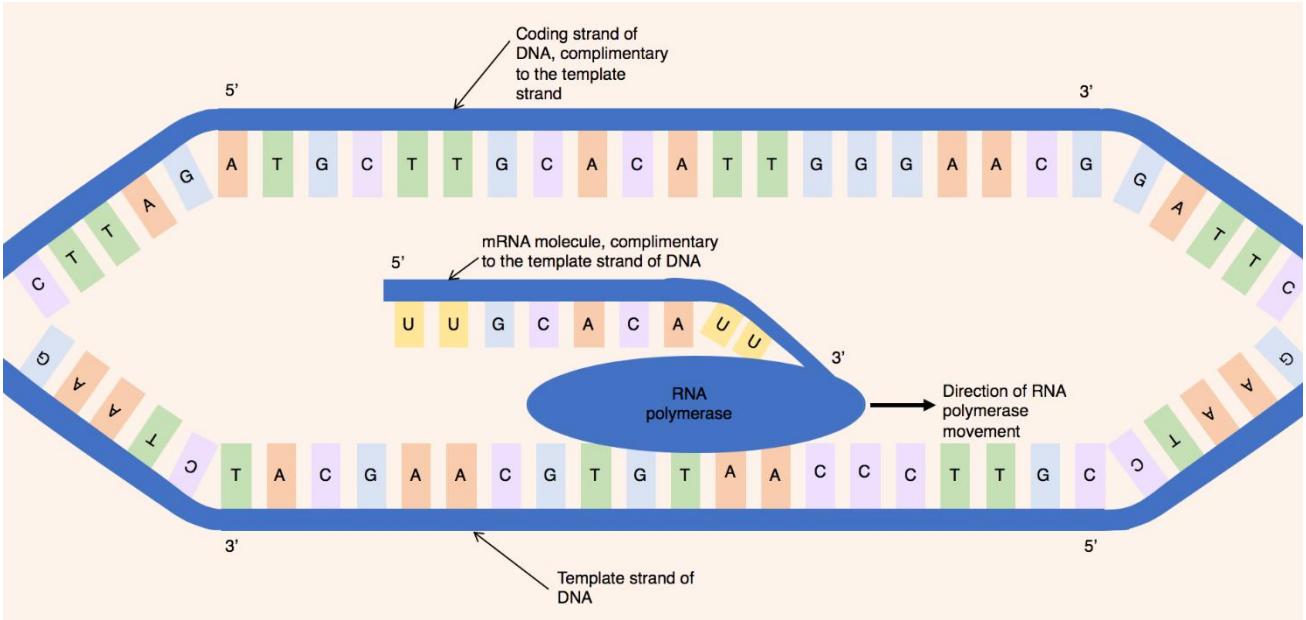
```
dna =
'TCGGGCGTAGTATGGGCGTAGAGGGCGTAGGGCGTAAACCGCGTCTGGGGCGTAAGGGCGTAGGGC
GTATGTCAGTGGGCGTACAGGGCGTAGGGGCGTATCGTATGGGGCGTATGGGCGTAAGGGCGTAAAG
TCGACGGGCGTAGGGCGTAACGGGCGTAGCGGGCGTACAGGGCGTATCGCAACGGGCGTATGGGCG'

print("Template DNA (in 5\'-3\' orientation):", dna)
```

First, we will obtain sequence of RNA synthesised from this ***template*** DNA strand. By convention, nucleic acid sequences are always given in 5'-3' orientation. As illustrated in the figure below, to obtain the RNA sequence we need (i) reverse the template (write it in 3'-5' direction), then (ii) get complementary DNA sequence, and finally (iii) substitute T-s to U-s.

Use slicing in reverse direction to get reverse DNA:

```
reverse_dna = dna[::-1] # [start:stop:step]
                        # -1 step says to go in reverse direction
print("Reverse DNA (in 3\'-5\' orientation):", reverse_dna)
```

Use your code from the last exercise in Practical 2 to find sequence complementary to the reverse DNA, call it `reverse_complement_dna`. If you didn't complete that exercise: use solution provided in **complementary_dna.ipynb/py**. You may note that, reverse-complement to *template* strand represents the *coding* strand, which is identical to RNA sequence, except for RNA having **U**-s instead of **T**-s.



Transcription of DNA to RNA
from https://en.wikipedia.org/wiki/Coding_strand

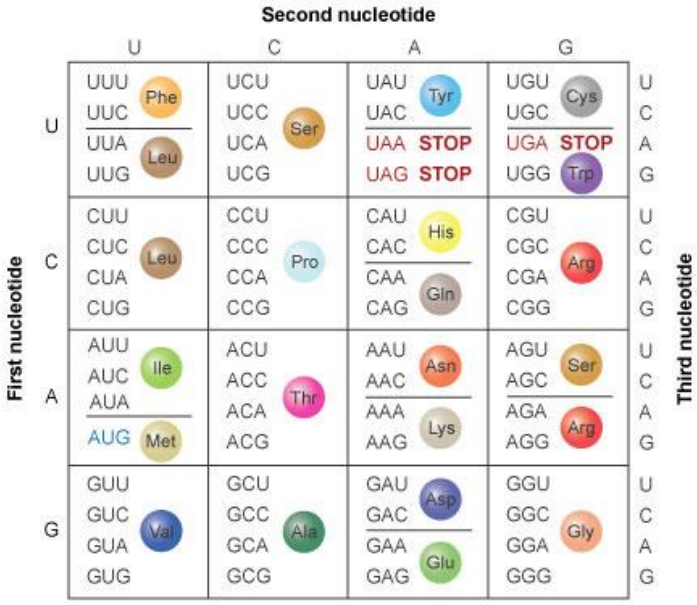
So, to get the RNA sequence ("*transcribe*" DNA into RNA) we need to replace all the **T**s (Thymine bases) for **U**s (Uracil bases) in the `reverse_complement_dna`.

We will use `replace()` function to convert DNA to RNA:

```
rna = reverse_complement_dna.replace('T', 'U')

print("RNA sequence:", rna)
```

Now, let's do the *translation* of RNA to protein.

RNA encodes the amino acids sequence of proteins. However, RNA has just four nucleotides, while the proteins have 20 amino acids. So, the encoding works by the groups of three nucleotides: each group (called triplet or codon) is coding for one amino acid. Some codons are used for the stop signal: when no amino acid is added, and the translation stops. Here's a table that shows the amino acids coded by each possible three-nucleotide combination. The sequence of these codons in RNA encodes a polypeptide chain (it's another fancy name for protein :).



So, we will start our *in silico* translation by converting the RNA sequence to the list of codons.

First, we need to define the initial position, where the translation starts. In this case we start from the beginning:

```
frame_start = 0
```

Then we make an empty list for codons:

```
codons = []
```

Then we find the end of the codons' chain in the given RNA @ selected `frame_start`:

```
frame_end = len(rna[frame_start:]) - len(rna[frame_start:]) % 3
```

Integer reminder (see Lecture 1)

Now we can get the codons from the RNA sequence:

```
for i in range(frame_start, frame_end, 3):
    codons.append(rna[i:i+3])

print(codons)
```

Q. What is the length of the given RNA sequence? How many codons have been identified in the sequence?

Exercise 4.2. *Complete translation of the RNA sequence to polypeptide chain using the file "codon_to_amino_acid_dictionary.txt". (hint: copy/paste the dictionary to your script)*

Well done! You have finished Practical 4.