Repeat Exam BIO134: Programming in Biology HS2022

29.08.2023; 13:30-16:00

Question 1

```
seq = 'TAATTTCTGACNATGGCGTCAATGGTACTCGCGNNGAG'
```

Write a program that, based on the string *seq*, prints all nucleotides before the first occurrence of letter *N* (stands for unidentified) as one nucleotide per output line.
It should thus print exactly:

```
T
A
A
T
T
T
C
T
G
A
C
```

The program should be general enough such that it would still work according to the same principle if *seq* was replaced by another sequence with a different length.

Question 2

The following list contains words that may occur more than once:

```
original = ['my', 'care', 'is', 'loss', 'of', 'care', 'by', 'old',
'care', 'done']
```

Make a program that creates, based on this list, a new list, which contains each word only once, in the order of their first occurrence.
Then print this list:

```
['my', 'care', 'is', 'loss', 'of', 'by', 'old', 'done']
```

The program should be general enough such that it would still work according to the same principle, if *original* contained different words and a different number of words.

Question 3

```
import numpy as np
np.random.seed(1)
```

Start your code with the lines above. Then, write a function *guessing()* that simulates a dice game, where two players each guess a number from 1-6 and the player whose number is rolled first, wins.
So, the function should take two different integer numbers from 1 to 6 as input and simulate rolling a dice by using *np.random.randint(1,7)*. The function should stop as soon as one of the two input numbers matches the actual roll. It should return 3 numbers, the winner (1 or 2), the guessed number that was rolled first, and the number of required attempts.

Call the function as follows:

```
w, n, a = guessing(3,4) # winner, number, attempt

print('Player {} won! {} was rolled with the {}. roll.'.format(w, n, a))
```

The program should now print the following, indicating who has won plus the number of tosses that were required.

```
Player 2 won! 4 was rolled with the 2. roll.
```

The function should be general enough such that it would work according to the same principle when called with different input arguments. You may assume that the two players never submit the same number as their guess.

Question 4

```
items = [['fly', 'bat', 'eagle'], ['hut', 'barn', 'villa', 'castle']]
```

Create a program that, based on the list *items*, creates a new list of the same dimensions but with the elements of each sublist in inverted order.

The program should then print the new list:

```
[['eagle', 'bat', 'fly'], ['castle', 'villa', 'barn', 'hut']]
```

You are not allowed to use *reversed()* or any other existing function that reverses the order of elements. The program should still work if the list *items* was of a different length, contained lists of different items, and/or different numbers of items.

Question 5

```
protein = 'MALWRLLPALALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAED'
protein += 'LQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSYQLENYCN'
```

Write a program that analyses the amino acid composition of the protein sequence in the above string *protein* by creating a dictionary that contains the amino acids as keys and their position in the sequence (starting with 0) in a list as values.
In the end, print the alphabetically sorted amino acids followed by their frequency and position in the exact following format:

```
A: 10 x at 1, 8, 10, 17, 18, 19, 33, 53, 69, 76
C:  6 x at 26, 38, 90, 91, 95, 103
D:  2 x at 15, 55
E:  8 x at 32, 40, 52, 54, 62, 78, 88, 100
F:  3 x at 20, 43, 44
G: 12 x at 13, 27, 39, 42, 59, 64, 65, 66, 68, 70, 79, 85
H:  2 x at 24, 29
I:  2 x at 86, 94
K:  2 x at 48, 83
L: 16 x at 2, 5, 6, 9, 11, 25, 30, 34, 36, 56, 63, 72, 75, 77, 81, 99
M:  1 x at 0
N:  3 x at 22, 101, 104
P:  6 x at 7, 14, 16, 47, 67, 74
Q:  7 x at 23, 57, 60, 73, 82, 89, 98
R:  5 x at 4, 41, 50, 51, 84
S:  5 x at 28, 71, 80, 93, 96
T:  3 x at 46, 49, 92
V:  6 x at 21, 31, 37, 58, 61, 87
W:  2 x at 3, 12
Y:  4 x at 35, 45, 97, 102
```

Note that the frequency is right-aligned to the third position after the colon and that the reported indices are separated by a comma and a space.

You may assume that no amino acid occurs more than 99 times. The program should still work according to the same principle if the string *protein* contained a different protein sequence of different length.

Question 6

The file *sequence.txt* contains the DNA sequence of human insulin in the FASTA format. The first line (FASTA definition line) starts with a greater-than (>) and contains the gene annotation; the second to last lines contain the DNA sequence, 60 nucleotides per line.

```
>NG_007114.1:4986-6416 Homo sapiens insulin (INS)
AGGACAGGCTGTATCAGAAGAGGCCATCAAGCAGGTCTGTTCCAAGGGCCTTTGCGTCAG
GTGGGCTCAGGATTCCAGGGTGGCTGGACCCCAGGCCCCAGCTCTGCAGCAGGGAGGACG
…
```

Write a program that finds potential transcription factor binding sites in the given DNA sequence based on the transcription factor binding motif *motif*.

```
motif = 'ACAGTCAGT'
```

Start by creating a string that only contains the full DNA sequence of the FASTA file. Next, slide the binding motif over the DNA sequence and calculate for each step a match score, which is defined as the number of identical nucleotides in the same position in the motif as well as in the corresponding DNA fragment. In the illustration below the nucleotides per sequence fragment that match the corresponding nucleotides in the motif are indicated in bold capital letters.

																		match score	start pos
sequence	**a**	**g**	**g**	**a**	**c**	**a**	**g**	**g**	**c**	**t**	**g**	**t**	**a**	**t**	**c**	**a**	**g**	**a**	…		
position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	…		
motif	A	C	A	G	T	C	A	G	T												
fragment	**A**	g	g	a	c	a	g	**G**	c											2	0
motif		A	C	A	G	T	C	A	G	T												
fragment		g	g	**A**	c	a	g	g	c	**T**										2	1
motif			A	C	A	G	T	C	A	G	T										
fragment			g	a	c	a	g	g	c	t	g									0	2
motif				A	C	A	G	T	C	A	G	T									
fragment				**A**	**C**	**A**	**G**	g	**C**	t	**G**	**T**								7	3

Find all sequence fragments in the insulin gene that have a match score larger than 6 for the given binding motif. Print them, together with their match score and the starting position, as follows:

```
ACAGgCtGT 7 3
cCAGTCAGa 7 593
AtAGTCAGT 8 891
ACAccCAGT 7 1029
```

Note that, as in the example above, the matching nucleotides in the fragments should be indicated as capital letters.
The program should still work according to the same principle with another binding motif of different length and if *sequence.txt* contained another DNA sequence of different length in the same FASTA format.
For opening the file, your program should simply use the file name without path information (i.e., relative path).

Question 7

```
persons = {}
person['darwin'] = ['Charles Darwin','12 February 1809', '19 April
1882']
persons['shakespeare'] = ['William Shakespeare', '26 April 1564','23
April 1616']
persons['cervantes'] = ['Miguel de Cervantes', '29 September
1547','23 April 1616']
persons['lincoln'] = ['Abraham Lincoln', '12 February 1809','15
April 1865']
```

Use the dictionary above to calculate the age of each individual person in years and days, considering that a year has 365 days and each month has a specific number of days (January: 31, February: 28, March: 31, April: 30, May: 31, June: 30, July: 31, August: 31, September: 30, October: 31, November: 30, December: 31). Ignore the leap years (*Schaltjahre*). Add the obtained age to the dictionary as the fourth and fifth elements of each individual's list, representing years and days, respectively.

The program should then print the dictionary:

```
{'darwin': ['Charles Darwin', '12 February 1809', '19 April 1882',
73, 66], 'shakespeare': ['William Shakespeare', '26 April 1564', '23
April 1616', 51, 362], 'cervantes': ['Miguel de Cervantes', '29
September 1547', '23 April 1616', 68, 206], 'lincoln': ['Abraham
Lincoln', '12 February 1809', '15 April 1865', 56, 62]}
```

- You are not allowed to use any pre-existing modules/functions that help to handle dates.
- Note that a dictionary does not have a specified order, so that a correct program may print the key-value pairs in a different order.
- The program should be general enough that the dictionary *persons* can be replaced by another dictionary that may contain data of a different number of persons.
- If you do not manage to find all the correct ages, please print your best intermediate result.