SOFTWARE 1 PRACTICAL

FUNCTIONS

Week 4 – Additional Exercises

Exercise 1: *The King and the Wise man*

When the creator of the game of chess showed his invention to the ruler of the country, the ruler was so pleased that he gave the inventor the right to name his prize for the invention. The man, who was very wise, asked the king this: that for the first square of the chess board, he would receive one grain of wheat (in some telling, rice), two for the second one, four on the third one, and so forth, doubling the amount each time. The ruler, arithmetically unaware, quickly accepted the inventor's offer, even getting offended by his perceived notion that the inventor was asking for such a low price and ordered the treasurer to count and hand over the wheat to the inventor. Given that the chessboard is a 8×8 board and given the weight of a single grain of rice is about 30 mg, calculate the total weight of rice the king must give to the wise man. The program should print the weight of rice for each chessboard square.

Exercise 2: *from a resit paper.*

You must use the file `textanalysis.py` to answer this question. Write a function `get_words_starting_with(text, letter)` that returns the list of words starting with `letter` in the string `text`. The result should not be case sensitive, e.g. ’about’ should be returned if the letter ’a’ or ’A’ is given as parameter. For simplicity, we assume for exercises 2,3, and 4 that the text does not have any punctuations, and words are separated by at least one blank space.

For example, using the variable `sample_text` we should obtain:

```
>>> get_words_starting_with (sample_text, ’a’)
[’As’, ’a’, ’about’, ’adding’, ’a’, ’ago’, ’a’,
’around’, ’Amsterdam’, ’a’, ’and’, ’an’, ’about’,
’a’, ’ABC’, ’appeal’, ’as’, ’a’, ’a’, ’a’]
>>> get_words_starting_with(sample_text, ’z’)
[]
```

Hint: You may want to use the method `split()` from the `str` type.

Exercise 3: *from a resit paper.*

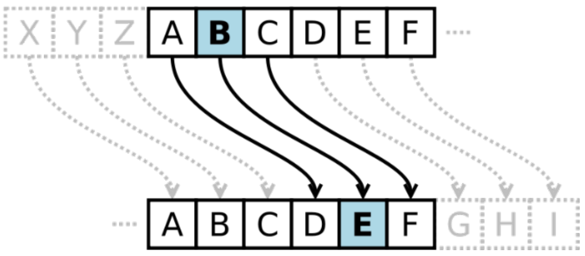
As you can see in question 2, there are many repetitions of the word ’a’ in the list. Improve your solution so no repetition of the same word occurs in the list.

```
>>> get_words_starting_with(sample_text, ’a’)
[’As’, ’a’, ’about’, ’adding’, ’ago’, ’around’,
’Amsterdam’, ’and’, ’an’, ’ABC’, ’appeal’, ’as’]
```

Exercise 4: *extension of exercise 3 Practical 2*

In cryptography, a **Caesar cipher**, also known as the shift cipher, is one of the simplest and most widely known encryption techniques. It is a type of substitution cipher in which each letter in the plain text is replaced by a letter some fixed number of positions down the alphabet.

For example, with a shift of 3, A would be replaced by D, B would become E, and so on. The method is named after Julius Caesar, who used it to communicate with his generals.



Mathematically, a Caesar cipher can be expressed by the following equation:

$$c = (p + a) \bmod 26$$

Here, 'mod 26' means that you use clock arithmetic for values greater than 26, i.e., 0=26 mod 26, 1=27 mod 26, 2=28 mod 26, …, 0=52 mod 26, 1=53 mod 26, …, 10=62 mod 26, and so on.

1. Write a function `caesar_encrypt` that encrypts a plain text into a cypher text using the Caesar Cipher algorithm. What parameters are needed? Should the function return something? For simplicity, assume that only alphabet letters are encrypted, other symbols remain the same.
2. Write a function `caesar_decrypt` that decrypts a cipher text into a plain text using the Caesar Cipher algorithm. What parameters are needed?
3. Given the cipher text below, and knowing it has been encrypted using a Caesar Cipher algorithm, could you decrypt it?

```
"bpm owwl vmea ijwcb kwuxcbmza qa bpib bpmg lw epib gwc
bmtt bpmu bw lw. bpm jil vmea qa bpib bpmg lw epib gwc
bmtt bpmu bw lw."
```

Exercise 5: *Vectors*

A vector of dimension n can be represented by a list in Python. For example, a vector of dimension 3 could represent a point in space, and a vector of dimension 4 could represent a point in space and time (the fourth dimension being the time). In mathematical notation, a vector of dimension 3 is represented as follow:

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

The vector could be stored in a Python list `[a, b, c]`. There are two simple operations that can be done on vector, and the result of the two operation is also a vector. The two operations are:

Scalar product: $$\lambda \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \lambda \cdot a \\ \lambda \cdot b \\ \lambda \cdot c \end{bmatrix}$$

Addition: $$\begin{bmatrix} a \\ b \\ c \end{bmatrix} + \begin{bmatrix} d \\ e \\ f \end{bmatrix} = \begin{bmatrix} a + d \\ b + e \\ c + f \end{bmatrix}$$

Implement two functions:

1. `scalar_product(scalar, vector)` where scalar is a float and vector is a list of float. The function returns the scalar product of the two parameters.
2. `vector_addition(vector1, vector2)` where `vector1` and `vector2` are lists of float. The function returns the vector addition of the two parameters. If `vector1` and `vector2` don't have the same dimension, you should print an error message and return `None`.