TPOP PRACTICAL

ITERATION

PRACTICAL 03

Question 1: *Palindrome*

A palindrome is a word, phrase, number, or other sequence of units that may be read the same way in either direction, `'radar'` and `'Delia saw I was ailed'` are palindromes, whereas `'reader'` is not.

1. Write a program that take a sentence or a word like `'radar'` as an input and print if it is a palindrome or not.
2. Modify your program (if needed) so it can take a sentence like `'Delia saw I was ailed'` as an input and print if it is a palindrome or not.

The advanced bit

`'Dammit, I'm mad!'` is also considered a palindrome when neither punctuation nor spaces are taken into account. Change your program so it can recognise these palindromes too.

During the Lecture in week 2, we have briefly introduce the notion of function. A function is defined using the keyword `def`, followed by the name of the function, and then the list of parameters between round brackets. The body of the function, i.e. all the statements that need to be executed, are listed below the function header/signature and must be indented. For example:

```
def printHelloWorld():
   print("hello world!")
```

The function above has no parameter (note the round brackets must still be there) and its only function is to print the text "hello world!". The body consists of a single print statement.

```
def introduceYourself(name):
   sentence = "Hi, my name is " + name.upper()
   return sentence
```

The second function above takes one parameter `name` and the body consists of two statements. We can also notice a new keyword, `return`. This means that the function returns a value, that could be stored in a variable. In this case it returns the value stored in the variable sentence. We can call the function and store the result of this function into a variable as shown below:

```
>>> intro = introduceYourself("lilian")
>>> intro
"Hi, my name is LILIAN"
```

In summary, when we declare a function we need to do several things:

- Give a name to the function
- Define what parameters are needed to do the computation, zero, one or more. The parameters can be seen as information that we need to provide in order to do the computation.
- Define if the function needs to return a value or not.

Let's consider the following problem. In order to know what truck's size we need to move house, and assuming that we have only boxes, we need to create a function that calculate the volume occupied by a box. Let's call the function `boxVolume`. What information do we need to do the computation? We need three, the length, the width, and the depth. That means the function needs three parameters. The next step is to decide if we need to return a value or not. In our case we want to return the total volume of the box. Now we can declare the function:

```
Def boxVolume(length, width, depth):
   totalVolume= length * width * depth
   return totalVolume
```

Note that I gave meaningful names to my parameters and variable. This is important, to ensure that other developer can read and understand my code more easily.

Exercise: modify your answer to question 1 to create a function named `palindrome`.

What are the parameters if any, should the function return a value?

Question 2: *Vectors*

A vector of dimension n can be represented by a list in Python. We would like to write two programs to evaluate two basic operations on vectors:

1. Print the scalar product of a vector (both inputs from the user).
2. Print the addition of two vectors (both inputs from the user). Note that the addition is possible only if the two vectors have the same dimension.
3. If you haven't done so in the first place, create two functions:
   a. `scalarProduct(scalar, vector)` where scalar is a float and vector is a list of float. The function returns the scalar product of the two parameters.
   b. `vectorAddition(vector1, vector2)` where vector1 and vector2 are lists of float. The function returns the vector addition of the two parameters.

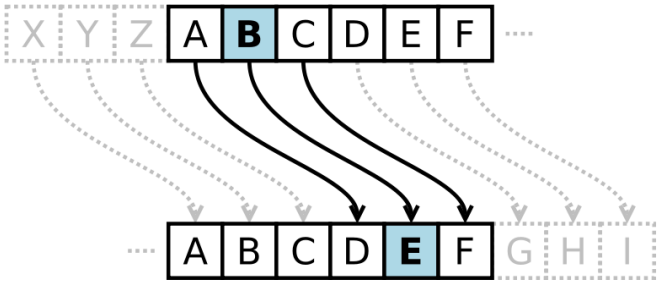
Scalar product: $$\lambda \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \lambda \cdot a \\ \lambda \cdot b \\ \lambda \cdot c \end{bmatrix}$$

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

Question 3: *Cryptography, Caesar Cipher*

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 arithmetics 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."
```