Basic Text Encryption / Decryption

In this mini-project, you will create a Python program that implements functions for encryption and decryption of a text using the "Caesar Cipher".

You should work on the project **individually**, at home or during the Lab sessions. You can ask the teaching assistants for help if you have any questions (during the exercises sessions or lab sessions). When you finish, please upload your source code to Blackboard. NOTE: Please be aware that any tentative of plagiarism or copying/modifying the source code from another student will be detected and will result in the project **not** being accepted from both students.

Introduction

In cryptography, Caesar's cipher is one of the simplest and most widely known encryption techniques. The action of a Caesar cipher is to replace each letter within a text with a different letter, using a fixed number of places down the alphabet **[1]**.

For example, to encrypt a given text, we can use a **left shift of three characters**, so that (for example) each occurrence of the letter **E** in the text becomes **B** in the encrypted text.

This transformation can be represented by aligning two alphabets : the original plain alphabet, and the cipher alphabet. The cipher alphabet is the plain alphabet shifted left by some number of positions **n**. This number of positions (**n=3** in the previous example) plays the role of an encryption/decryption key.

Plain Alphabet :	ABCD**E**FGHIJKLMNOPQRSTUVWXYZ
Cipher Alphabet :	XYZA**B**CDEFGHIJKLMNOPQRSTUVW

To encrypt a message using a given key (e.g. **n=3**), a person looks up each letter of the message in the *Plain Alphabet* and writes down the corresponding letter in the *Cipher Alphabet*. Example:

Clear Message:	« THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG »
Encrypted message:	« QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD »

To decrypt the encrypted message, we repeat the same process in reverse (i.e., with a **right** shift of 3 positions).

NOTE: some built-in Python functions that may be useful for this mini-project are `ord(…)` and `chr(…)`. Check out, for example, the following links (or other links) to learn about these built-in functions:

- https://www.w3schools.com/python/ref_func_ord.asp

- https://www.w3schools.com/python/ref_func_chr.asp
- https://www.geeksforgeeks.org/ord-function-python/

Project Description

1. Encryption function

First, you are asked to define a function `encrypt(msg, n)` that takes as arguments a text `msg` (as a string) and a key `n` (as an integer number). This function is supposed to shift all the characters in the text message **to the left** by `n` positions, and return a new encrypted message.

If `msg` contains any characters that are not part of the alphabet (such as spaces etc.), then you can keep them unchanged (i.e. no need to encrypt them).

```
def encrypt(msg, n):
    # Implement you code here
    # ...
    return encrypted_msg
```

Once your function `encrypt(message, n)` is defined, you can test it on some text (e.g. "`THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG`") and check if it returns the expected encrypted text (e.g. "`QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV ALD`" if you chose `n = 3`).

2. Decryption function

Second, you are asked to define a function `decrypt(encrypted_msg, n)` that takes as arguments an encrypted text `encrypted_msg` (as a string) and a key `n` (as an integer number). This function is supposed to decrypt the text message by shifting all the characters of the encrypted message **to the right** by `n` positions, and return the decrypted message.

If `encrypted_msg` contains any characters that are not part of the alphabet (such as spaces etc), then you can keep them unchanged (i.e. no need to decrypt them).

```
def decrypt(encrypted_msg, n):
    # Implement you code here
    # ...
    return clear_msg
```

Once your function `decrypt(encrypted_msg, n)` is defined, you can test it on some text (e.g. "`QEB NRFZH YOLTK CLU GRJMP LSBO QEB IXWV AL`") and check if it

returns the correct decrypted text (e.g. "`THE QUICK BROWN FOX JUMPS OVER THE LAZY DOG`" if you chose `n = 3`).

3. Reading / Writing to text a file

In this part, you are asked to write a code to read each line from a text file named `clearText.txt`, encrypt the line by calling the `encrypt` function that you defined previously, then write (append) the encrypted line to another file `encryptedText.txt`.

To read from an existing file in Python you first need to open the file in *read mode* using the function `open("clearText.txt", "r")` (note the second argument `"r"` which stands for "*read mode*") and then read the lines from the file by calling the function `readlines()` as follows:

```
f1 = open("clearText.txt", "r")
lines = f1.readlines()
```

The variable `lines` consists of a list of strings (corresponding to lines). So, you can iterate over each `line in lines` using a `for` loop.

To append an encrypted line to a file (i.e. write the line without overwriting the previous content), you can open the file in *append mode* and then write to the file by calling the function `write(..)` as follows:

```
f2 = open("encryptedText.txt", "a")
lines = f2.write(encrypted_line)
```

4. Letter frequency analysis

The encryption method that you implemented previously is not very secure. It is possible to find the original text from the encrypted text by performing a process called frequency analysis.

The reason why this kind of encryption is not secure, is because each letter in the original text is always encrypted the same way. For example, the most common letter in the English language is "**E**" and it might always be encrypted as "**B**" for example, so if a hacker finds that "**B**" is the most common letter in the encrypted text then he can assume it represents "**E**". From this, the hacker can deduce that the original text has been encrypted by shifting letters to the left by 3 positions. Now, that he knows the key (i.e. 3), he can easily decrypt the text. This process can be carried out for all letters, and it is called frequency analysis.

A first step to perform frequency analysis is to count the frequency (or number of occurrence) of each letter in the encrypted text.

In this part of the project, you are asked to write a function that computes the number of occurrences of each letter of the alphabet in the encrypted text. The output should correspond to something like the following (of course, the numbers will depend on the text that you use) :

```
A occurs 29 times in the encrypted text
B occurs 15 times in the encrypted text
C occurs 18 times in the encrypted text
... etc ...
Z occurs 7 times in the encrypted text
```

[1] Caesar_cipher, https://en.wikipedia.org/wiki/Caesar_cipher