Caesar Cipher

Criptography
Carmen Azorín Martí

March 22th 2024

1 Introduction

The code written in Python will use two external .txt files. These files contain an original message (*message.txt*) and a *modified_message.txt*, after the desired encryption or decryption operation.
In addition, the Python code will use the *Unicode* library that allows us to read a file containing letters of the Polish alphabet. In addition, it will also use the *codecs* library, used to open and read a file saving the original letters.
Suppose our *message.txt* file contains the message

XKZBKKGN OVELSO333UBIZDYQBKP
SO

which we obviously want to decrypt.

2 Implemented functions

The first function implemented in Pyhton is called *removeUnnecesaryChars()* and receives a string as parameter. This line of text could contain characters other than letters, such as spaces, line breaks, etc. What the function returns is the string removing all non-alphabetic characters.

```python
def removeUnnecesaryChars(text):
    return ''.join(letter for letter in text if letter.isalpha())
```

This method would take care of debugging our specified message, updating it to

XKZBKKKGNOVELSOUBIZDYQBKPSO

The following functions are actually the most important of the whole project. They are the functions that encrypt and decrypt depending on a key.
The *encryptText()* function receives two parameters:

- Text. the message to be encrypted
- Key. is the number between 0 and 25 to be used

Since the encrypted message has to be in uppercase letters independently of the original message, we put everything in uppercase before encrypting it.
Then we go character by character from the original message and encrypt it. For this we need the number assigned to each letter in the ASCII table with the *ord()* function. For the number to be between 0 and 25, we have to subtract *ord('A')*. If character is a letter between A and Z, then

$$ord(character) - ord('A') \in Z_{26}$$

Once we have the original character with a number assigned between 0 and 25, we add the key and do modulo 26. Finally, to get the new character in the ASCII table, we have to add *ord('A')* and pass the new number to the *chr()* function.

```
def encryptText(text, key):
    text = text.upper()
    print(text)
    encrypted_text =''
    for character in text:
        character_to_num = ord('A') + (ord(character) + key - ord('A'))%26
        encrypted_text += chr(character_to_num)
    return encrypted_text
```

The *decryptText()* function receives two parameters:

- Text. the message to be decrypted
- Key. is the number between 0 and 25 to be used

This time the decrypted message must be in lower case.
We will go character by character of the message calculating the original character. To do this, we pass the character to a number from the ASCII table with the *ord()* function, and pass it to a number between 0 and 25 by subtracting *ord('a')*. If chaarcter is a letter between a and z, then

$$ord(character) - ord('A') \in Z_{26}$$

To calculate the original number, we subtract the key and calculate the modulus 26. And to pass the new number to the ascii table, we add the number from *ord('a')*. This number is assigned to the letter indicated by *chr*.

```
    def decryptText(text, key):
    text = text.lower()
    print(text)
    decrypted_text = ''
    for character in text:
        character_to_num = ord('a') + (ord(character) - key - ord('a'))%26
        decrypted_text += chr(character_to_num)
    return decrypted_text
```

In our example, we would call the *decryptText()* function by passing the string "XKZBKKKGNOVELSOUBIZDYQBKPSO" and the key 10. The algorithm would transform the string into lowercase and start with character = x, we have that:

$$ord(character) - ord('a') = 120 - 97 = 23 \in Z_{26}$$

Then, substracting the value of the key 10 and applying modulo 26:

$$\begin{aligned} ord('A') + (ord(character) - key - ord('a')) \mod 26 &= 97 + (23 - 10) \mod 26 \\ &= 97 + 13 \\ &= 110 \end{aligned}$$

The value of 110 in the ASCII table corresponds to letter n, so the program appends this letter in the variable decrypted_text. Then, the algorithm would continue with character = k, and so on. Finally, the value of decrypted_text would be "naprawdelubiekryptografie".

3 Get the message

To get the message we want to encrypt or decrypt we have to read the *message.txt* file. This file can contain national characters, so to read it without problems, we use the *codecs open()* function. We save the message read from the file in the variable message and close the file reader.

```
    file = codecs.open('message.txt','r','utf-8')
    message = file.read()
    file.close()
```

To be able to write to the *modified_message.txt* file, we open it with the 'w' option. This file should be empty.

```
    file_2 = open('modified_message.txt', 'w')
```

The text stored in the message variable can have national characters, as mentioned above. To change these national characters to characters of the Latin alphabet, we use the *unicode()* function. Finally, we use the function we created earlier, *removeUnnecessaryCharacters()*, to remove non-letter characters.

```
message = unidecode(message)
message = removeUnnecesaryChars(message)
```

4 Menu implementation

The menu will be the part of the code in charge of interacting with the user. The first thing we will do is ask the user to enter a number indicating whether they want to encrypt (1) or decrypt (0) a message. We will store the number entered in the encrypt_option variable.
Afterwards, we ask the user for the key he wants to use and store it in the variable key.
Depending on the options chosen, the *encryptText()* or *decryptText()* function will be called and the result will be saved in *modified_message.txt*. In case the key is not between 1 and 25, or the encryption option is not 0 or 1, a message will be launched indicating that no valid values have been entered.

```
encrypt_option = input('Press 1 for encrypting a message and press 0 for decrypting a
                                              message')
key = input('Write the key you want to use from 1 to 25')
if encrypt_option == '1' and 1 <= int(key) <= 25:
    file_2.write(encryptText(message,int(key)))

elif encrypt_option == '0' and 1 <= int(key) <= 25:
    file_2.write(decryptText(message, int(key)))

else:
    print('The values introduced are not valid')

file_2.close()
```