Coursework 1 - Exercise 1

October 29, 2023

Alice wants to send a couple of secret messages to Bob. To achieve this, they both agreed on OTP key which they will use for encryption and decryption. While one of the messages were being sent you managed to obtain both the plaintext message m_1 and the corresponding ciphertext c_1.

a) Can you compute the OTP key from m_1 and c_1, when:

$$m_1 = \text{LIFEISLIKEABOXOFCHOCOLATES}$$

$$c_1 = \text{CXGDXNIPWXYXTONWQTCVCFXKCY}$$

If it is possible, describe the process of how to achieve the key.

b) Alice and Bob continue to use the same OTP key for multiple messages. Please recover the new message m_2 using all previously known information.

$$c_2 = \text{PDVMTQBYWGMSBYZKMAIPWFIXCZ}$$

---

a) To compute the OTP key we have just have to calculate the distance between each letter of the ciphertext and plaintext.

We can implement the following function:

[1]:
```
def get_key(ciphertext, plaintext):
    assert len(ciphertext) == len(plaintext), "Ciphertext and plaintext must be␣
↪of the same length"
    return [ord(ciphertext[i]) - ord(plaintext[i]) for i in␣
↪range(len(ciphertext))]
```

And then we get the key:

[2]:
```
ciphertext = "CXGDXNIPWXYXTONWQTCVCFXKCY"
plaintext = "LIFEISLIKEABOXOFCHOCOLATES"

key = get_key(ciphertext, plaintext)
print(key)
```

```
[-9, 15, 1, -1, 15, -5, -3, 7, 12, 19, 24, 22, 5, -9, -1, 17, 14, 12, -12, 19,
-12, -6, 23, -9, -2, 6]
```

b) To obtain the second message, we have to reverse the operation, by substracting the key to each character of the ciphertext, making sure we perform the right modulus operations to assert the characters remain within the letter range

[3]:
```
def decrypt(ciphertext, key):
    return "".join([chr((ord(ciphertext[i]) - key[i] - 65) % 26 + 65) for i in␣
 ↪range(len(ciphertext))])
```

This way we get the following message:

[4]:
```
ciphertext2 = "PDVMTQBYWGMSBYZKMAIPWFIXCZ"
print(decrypt(ciphertext2, key))
```

```
YOUNEVERKNOWWHATYOUWILLGET
```