Exercice 1 :

1. "noël".encode() → b'no\xc3\xabl'
2. b'no\xc3\xabl'.decode() → "noël"
3. Table de vérité

```
0 ^ 0 # affiche 0
0 ^ 1 # affiche 1
1 ^ 0 # affiche 1
1 ^ 1 # affiche 0
```

4. Chiffrer

```
def chiffrer_xor(message: bytes, cle: bytes) -> bytes:
    res = [message[i] ^ cle[i % len(cle)] for i in range(len(message))]
    return bytes(res)
```

5. Message en clair

```
m_dechiffre = chiffrer_xor(m_chiffre, cle.encode())
print(m_dechiffre)
print(m_dechiffre.decode())
```

Exercice 2 :

```
def compare_fin(message: bytes, fin: bytes) -> bool:
    """
    compare la fin de 'message' à 'fin'
    NB: on compare ici des octets; len(bytes) renvoie un
    nombre d'octets (cela reste équivalent à comparer des
    lettres ASCII)
    """
    return message[-len(fin):] == fin


def compare_fin2(message: bytes, fin: bytes) -> bool:
    i_message = len(message)-1
    i_fin = len(fin)-1
    while i_fin >= 0 and message[i_message] == fin[i_fin]:
        i_message -= 1
        i_fin -= 1
    return i_fin < 0
```

```
def bruteforce(secret: bytes) -> tuple:
    """
    Teste toutes les combinaisons de 3 lettres minuscules
    pour trouver la clé
    """
    for l1 in range(97, 123):
        for l2 in range(97, 123):
            for l3 in range(97, 123):
```


Christophe Viroulaud

```
                # création de la clé
                cle = chr(l1)+chr(l2)+chr(l3)
                # tentative de décryptage avec cette clé
                message = chiffrer_xor(secret, cle.encode())
                # comparaison de la fin du message
                if compare_fin(message, "Tof!".encode()):
                    return (message.decode(), cle)
    return "Message non décrypté!"
```

```
debut = time.time()
message_decrypte = bruteforce(message_secret)
fin = time.time()
print(message_decrypte)
print(fin-debut)
```

On obtient un durée de 0,15 secondes pour décrypter le message. À noter qu'avec une clé de quatre caractères on monte à 3,5 secondes.

Christophe Viroulaud