Exercice 1 : Le chiffre par *ou exclusif* est réversible : on utilise la même fonction pour chiffrer et déchiffrer le message. Pour pouvoir chiffrer le message bit à bit, il est plus aisé de manipuler des chaînes d'octets (`bytes`). Chaque caractère est converti en une suite d'octets (voir le cours de première sur l'Unicode). On transforme une chaîne de caractères en chaîne d'octets avec la méthode `encode`. On utilise la méthode `decode` pour l'opération inverse.

Également si on n'utilise pas des chaînes d'octets, le chiffrage peut créer des caractères non imprimables.

UTF-8 utilise 2 octets pour coder le ë ; \x → les 2 prochains caractères sont hexa.

— ë → \xc3\xab

— c3ab → 1100 0011 1010 1011

— 1100 0011 1010 1011 → 000 1110 1011

— 1110 1011 → eb

— ë → U+00eb

1. Encoder la chaîne de caractères, *noël*, en chaîne d'octets. Remarquer que la chaîne d'octets est enveloppée dans : *b' '*.
2. Décoder la chaîne obtenue pour retrouver le message d'origine.

En Python il est possible d'effectuer des opérations directement sur les bits. L'opérateur ∧permet d'effectuer un *ou exclusif* entre deux bits.

3. Dans la console tester l'opérateur et retrouver la table de vérité du *ou exclusif*.

Alice a envoyé à Bob le message chiffré suivant :

b'\x06Sb\x04a\x0bjQe/A6j_\x81\xe8\xf1\xe0-[3 ?Wc'

La clé est :

J2B

4. Écrire la fonction `chiffrer_xor(message: bytes, cle: bytes)` → `bytes` qui permet de coder et décoder un message par la méthode du *ou exclusif*. Indication : Une utilisation judicieuse du modulo permettra d'étendre la clé sous le message.
5. Retrouver le message envoyé par Alice.

Christophe Viroulaud