

# TD : Conversions et encodages

---

## I. Conversions binaire/décimal

L'objectif de cette première partie est de vous familiariser avec l'écriture binaire et les conversions entre la base 10 (écriture décimale) et la base 2 (écriture binaire).

Dans toute cette partie, un nombre écrit en binaire sera représenté par une liste d'entiers contenant des 0 et des 1 et commençant par le bit de poids fort.

Par exemple, le nombre 42, dont l'écriture en base 2 est 101010<sub>2</sub>, sera représenté dans Python :

- En base 10, par l'entier 42
- En base 2, par la liste d'entiers [1, 0, 1, 0, 1, 0]

On ne manipulera en outre que des entiers positifs ou nuls.

### a. De la base 2 à la base 10

**Question 1.** Pour vous réapproprier la notion de liste en Python, commencer par écrire une fonction `base2_valide` telle que `base2_valide l` renvoie `True` si la liste `l` ne contient que des 0 et des 1, et `False` sinon.

**Question 2.** Ecrire une fonction `base2_vers_base10` telle que `base2_vers_base10 l` renvoie l'entier dont la liste `l` est la représentation en base 2.

**Remarque :** Si la liste `l` n'est pas une écriture en base 2 valide, utiliser la syntaxe `raise Exception("Message a afficher")` pour renvoyer une erreur.

### b. De la base 10 à la base 2

**Question 3.** Ecrire une fonction `taille_ecriture_binaire` telle que `taille_ecriture_binaire n` renvoie le nombre de bits nécessaires pour écrire l'entier `n` en base 2.

**Question 4.** Ecrire une fonction `base10_vers_base2` telle que `base10_vers_base2 n` renvoie la liste de 0 et de 1 correspondant à l'écriture en binaire de l'entier `n`.

**Question 5.** Ecrire une fonction `base10_vers_octet` telle que `base10_vers_octet n` renvoie la liste de 0 et de 1 correspondant à l'écriture en binaire sur 8 bits de l'entier `n`.

**Indice :** Dans certains cas, le bit de poids fort pourra valoir 0.

**Remarque :** Si l'entier `n` n'est pas compris entre 0 et 255, utiliser la syntaxe `raise Exception("Message a afficher")` pour provoquer une erreur.

## II. Encodages

Cette seconde partie est consacrée à la représentation de caractères en binaire.

### a. Les chaînes de caractères en Python

Parmi les types standards en Python, on trouve le type `str`, qui correspond aux chaînes de caractères (généralement notées entre guillemets, ex : `"hello"`).

Lorsqu'une telle chaîne ne contient qu'un seul caractère, on parle simplement de "caractère" (dans certains langages, il existe un dédié qui se nomme `char`).

Comme vous pourrez le constater avec les fonctions ci-dessous, le type `str` est proche par certains aspects du type `list` (une chaîne de caractère est en quelque sorte une liste de caractères) :

- Pour obtenir la longueur d'une chaîne de caractères `s`, on utilise `len(s)`
- Pour créer une chaîne de caractères `s` de longueur `n` contenant uniquement le caractère `c`, on écrit `s = c * n`
- Pour accéder au caractère situé à la position `i` d'une chaîne `s`, on utilise `s[i]`

**Remarque :** Contrairement à d'autres langages, Python n'autorise pas l'édition des chaînes de caractères : il est par exemple impossible de modifier le  $i^{\text{ème}}$  caractère d'une chaîne.

Python met en outre à votre disposition deux fonctions liées au code ASCII :

- La fonction `ord` renvoie l'entier correspondant en ASCII au caractère passé en argument.
- La fonction `chr` renvoie le caractère correspondant en ASCII à l'entier passé en argument.

### b. De l'ASCII au binaire

**Question 6.** Ecrire une fonction `char_vers_octet` telle que `char_vers_octet c` renvoie la liste de 0 et de 1 correspondant à la représentation binaire basée sur le code ASCII du caractère `c`.

**Question 7.** Ecrire une fonction `str_vers_octets` telle que `str_vers_octets s` renvoie la liste de 0 et de 1 correspondant à la représentation binaire basée sur le code ASCII de la chaîne de caractère `s`.

### c. Du binaire à l'ASCII

**Question 8.** Ecrire une fonction `octet_vers_char l` telle que `octet_vers_char l` renvoie le caractère dont la liste `l` est la représentation binaire basée sur le code ASCII.

**Question 9.** Ecrire une fonction `octets_vers_str l` telle que `octets_vers_str l` renvoie la chaîne de caractères dont la liste `l` est la représentation binaire basée sur le code ASCII.

**Indice :** Pour ajouter un caractère `c` à une chaîne `s`, on peut écrire `s = s + c`.

### d. Un code « artisanal »

Dans cette dernière partie, nous allons travailler sur un code basé sur le principe suivant :

- Les seuls caractères autorisés sont les lettres majuscules et le symbole espace.
- Chaque lettre est codée par sa position dans l'alphabet (la lettre A étant représentée par 1)
- Le symbole espace est codé par 0

★ **Question 10.** Ecrire une fonction `char_vers_code c` telle que `char_vers_code c` renvoie le code du caractère passé en argument.

**Indice :** La fonction `ord` renvoie un entier, et on peut soustraire des entiers.

★ **Question 11.** Ecrire une fonction `str_vers_codes s` telle que `str_vers_codes s` renvoie la liste des codes des caractères composants la chaîne `s`.

★ **Question 12.** Ecrire une fonction `code_vers_char n` telle que `code_vers_char n` renvoie le caractère associé à l'entier `n`.

★ **Question 13.** Ecrire une fonction `codes_vers_str l` telle que `codes_vers_str l` renvoie la chaîne de caractères correspondant à la liste de codes `l`.