

**Méthode 1 : passer de la base 2 à la base 10**

Que vaut  $(11101)_2$  ?

Chiffre binaire	1	1	1	0	1
Valeur	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

$$\begin{aligned}(11101)_2 &= 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\ &= 16 + 8 + 4 + 1 \\ &= 29\end{aligned}$$

**Exercice 1**

Écrire un programme à la main qui :

- demande à l'utilisateur d'entrer un nombre en binaire sous la forme d'une chaîne de caractères composées uniquement de 0 et de 1;
- affiche l'écriture décimale du nombre binaire que l'utilisateur a entré.

**Comment faire ?**

En reprenant l'exemple de la méthode 1, on entre 11101 dans une variable chaîne et

- on voit que la longueur de cette chaîne est 5;
- donc `chaîne[0]` est le bit de  $2^4$ , `chaîne[1]` est le bit de  $2^3$ , ..., `chaîne[4]` est le bit de  $2^0$ ;
- ainsi on peut créer une variable nombre qui vaut zéro et une boucle for pour parcourir chaîne;
- si `chaîne[i]` vaut 1 on ajoute la valeur correspondante à somme sinon on ne fait rien;
- en sortie de boucle on affiche somme.

## Exercice 2

Écrire le programme précédent sur ordinateur. Il devra s'appeler `methode1.py`

### Méthode 2 : passer de la base 10 à la base 2

$$\begin{aligned} 203 &= 128 + 64 + 8 + 2 + 1 \\ &= 2^7 + 2^6 + 2^3 + 2^1 + 2^0 \\ &= 1 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 \\ &= (11001011)_2 \end{aligned}$$

## Exercice 3

Écrire un programme à la main qui :

- demande à l'utilisateur un entier positif (avec `int(input(...))`);
- affiche l'écriture en binaire de cet entier;

### Comment faire ?

Lors de la méthode 2, j'ai d'abord déterminé que 128 est la plus grande puissance de 2 inférieure à 203.

J'ai commencé à faire une somme commençant à 128. Ensuite j'ai regardé (en faisant des comparaisons) et *dans cet ordre* si 64, 32, 16, 8, 4, 2 et 1 « rentrent aussi » dans cette somme.

Suivant les cas j'obtiens des bits à 1 ou à 0.

Concrètement :

- on suppose que l'entier est dans une variable `nombre` et on définit une variable de type `str` `binaire`;
- d'abord on doit déterminer la plus grande puissance de 2 inférieure à `nombre` (penser à une boucle pour le faire)
- si on note `n` cette puissance, alors on peut créer une boucle pour parcourir les entiers de `n` à 0 en descendant et regarder si les puissances de 2 sont plus grande que `nombre`;
- si c'est le cas on enlève la puissance de 2 à `nombre` et on ajoute un '1' à `binaire`. Sinon on n'enlève rien et on ajoute un '0' à `binaire`;
- en sortie de boucle on affiche `binaire`.

Écrire le programme précédent sur machine, il devra s'appeler `methode2.py`

Voici comment on trouve les chiffres de l'écriture *binaire* de 203 :

$$\begin{array}{r}
 203 \overline{) 2} \\
 \underline{101} \phantom{00} \\
 101 \phantom{00} \\
 \underline{50} \phantom{00} \\
 50 \phantom{00} \\
 \underline{25} \phantom{00} \\
 25 \phantom{00} \\
 \underline{12} \phantom{00} \\
 12 \phantom{00} \\
 \underline{6} \phantom{00} \\
 6 \phantom{00} \\
 \underline{3} \phantom{00} \\
 3 \phantom{00} \\
 \underline{1} \phantom{00} \\
 1 \phantom{00} \\
 \underline{0}
 \end{array}$$

En définitive,  $203 = (11001011)_2$ .

Pour cet exercice il faut se « débrouiller tout·e seul·e » en tirant les leçons des exercices précédents.

1. Écrire un programme à la main qui :
  - demande un entier positif à l'utilisateur;
  - affiche son écriture en binaire en appliquant la méthode précédente.
2. Écrire le programme Python sur l'ordinateur, il devra s'appeler `methode3.py`