

Exercice 1

Rappelle-toi ce que nous avons vu en cours :

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

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

Tu vas compléter le programme appelé `methode1.py` qui suit... la méthode 1 :

- il 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 fonctionne ce programme sur un exemple ?

En reprenant l'exemple de l'encadré 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**.

Tu peux déjà commencer par compléter sur papier :

Code Python

```
nnombrebinaire o= n+nbinputp(l+s+s2Entrez un nombre en binaire :p)
nvaleur o= o.....
nn o= n+nbinputp(o.....p)
kfor ni o+owin n+nbrangep(o.....p):
    kif nnombrebinairep[o.....p] o== l+s+s11p:
        nvaleur o+= l+m+mi2o**p(o.....p)
n+nbprintp(l+s+s2Cela vautp,nvaleurp)
```

Ensuite tu peux le programmer avec EDUPYTHON.

Exercice 2

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

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

Tu vas compléter le programme `methode2.py` qui

- demande à l'utilisateur un entier positif;
- affiche l'écriture en binaire de cet entier (au format `n+nbstr`) en suivant la méthode 2;

Comment fonctionne la méthode sur l'exemple ?

- j'ai d'abord déterminé que 128 est la plus grande puissance de 2 inférieure à 203 : je suis parti de 1 puis je l'ai multiplié par 2, par 2 etc, jusqu'à 256. Puisque 256 est strictement plus grand que 203, la plus grande puissance inférieure à 203 est 128.
- j'ai commencé par définir une variable `nbinaire` de type `n+nbstr` valant `l+s+s2`.
- je peux enlever 128 à 203, donc je peux ajouter `l+s+s21` à `nbinaire` : c'est le bit de 128.
- j'enlève 128 à 203, il reste 75 et je regarde si la puissance de 2 « juste avant » 128 est inférieure à 75 : $64 < 75$ donc à ma variable `nbinaire` j'ajoute `l+s+s11` (bit de 64).

- je recommence en enlevant 64 à 75 : il reste 11 et 32 « ne rentre pas dans 11 » donc j'ajoute `l+s+s10` à `nbinaire`.
- et ainsi de suite jusqu'à ce qu'il ne me reste plus rien.

Tu peux commencer par compléter sur papier :

Code Python

```
nentier o= o..... p(n+nbinputp(l+s+s2Entre un entier :p))
n+nbprintp(l+s+s1Je commence par trouver la plus grande puissance
    de 2 inférieure àp, nentierp)
ni o= l+m+mi0
kwhile l+m+mi2 o** ni o= o..... p:

    ni o= ni o+ l+m+mi1
ni o= ni o.....

nbinaire o= o.....
kwhile nentier o l+m+mi0p:

    kif nentier o= o..... p:
        nbinaire o= nbinaire o+ l+s+s21
        nentier o= nentier o l+m+mi2 o** ni
    kelsep:
        nbinaire o+= o.....
    ni o= o.....

n+nbprintp(l+s+s2En définitive jai trouvép, nbinairep)
```

Programme ensuite ton script avec `nEduPython`.

Exercice 3

Tu vas devoir compléter le programme `nmethode3o.npy` qui utilise la méthode des divisions successives par 2 pour obtenir l'écriture binaire (au format `n+nbstr`) d'un entier.

Méthode 3 : les divisions successives

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

$$\begin{array}{r|l}
 203 & 2 \\
 \hline
 1 & 101 \\
 \hline
 & 2 \\
 & \hline
 & 50 \\
 & \hline
 & 25 \\
 & \hline
 & 12 \\
 & \hline
 & 6 \\
 & \hline
 & 3 \\
 & \hline
 & 1 \\
 & \hline
 & 1 \\
 & \hline
 & 0
 \end{array}$$

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`

Tu peux commencer par compléter sur papier :

Code Python

```
nentier o= o..... p(n+nbinputp(l+s+s2Entrez un nombre :p))

nresultat o= o.....
kwhile nentier o!= o..... p:
    nresultat o= n+nbstrp(o.....p) o+ nresultat
    nentier o= nentier o// l+m+mi2
n+nbprintp(nresultatp)
```

Programme ensuite ton script avec nEduPython.