1 Découvrir les bases 2, 10 et 16

1.1 Compter en base décimale - base 10

La base décimale est la base qui nous est habituelle et quotidienne. Prenons le temps de réfléchir ce que signifie **compter** !

En décimal, il existe 10 chiffres (caractères) pour compter : 0 1 3 4 5 6 7 8 9.
En comptant, après 9, il vient 1 dizaine (10^1) et 0 unité (10^0), qui s'écrit 10.
Après 99, qui représente 9 dizaines et 9 unités, il vient 1 centaine (10^2) et 0 dizaine (10^1) et 0 unité (10^0), qui s'écrit 100.

Le nombre qui s'écrit 3256 s'interprète ainsi :
3 milliers (10^3), 2 centaines (10^2), 5 dizaines (10^1) et 6 unités (10^0)
$3256 = 3 \times 10^3 + 2 \times 10^2 + 5 \times 10^1 + 6 \times 10^0$

1.2 Compter en base binaire - base 2

La base binaire est la base utilisée en dans le monde numérique car les ordinateurs, fondamentalement, ne peuvent travailler qu'avec deux informations (soit le courant passe, soit il ne passe pas !).

En binaire, il n'existe que 2 caractères : 0 et 1.

En comptant, après 1, il vient donc 10 ! c'est à dire 1 "*deuzaine*" (2^1), et 0 unité (2^0).
En poursuivant avec la même logique, après 11 (1 "*deuzaine*" et 1 unité), il vient 100 : 1 "*quatraine*" (2^2), 0 "*deuzaine*" (2^1) et 0 unité (2^0).

Le tableau suivant compte en binaire jusqu'à l'équivalent de 15 en décimal. (on écrira 15_{10}).

décimal	binaire
0	0
1	1
2	10
3	11
4	100
5	101
6	110
7	111

décimal	binaire
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

On peut par exemple écrire les conversions suivantes :
- $(101)_2 = (5)_{10}$
- $(1000)_2 = (8)_{10}$
- $(1111)_2 = (15)_{10}$

Le nombre qui s'écrit 10010110 s'interprète ainsi :
$(10010110)_2 = (1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0)_{10}$
$(10010110)_2 = (128 + 16 + 4 + 2)_{10} = (150)_{10}$

Il sera utile d'apprendre par cœur les 8 premières puissance de 2 :

puissance de 2	décimal	binaire
0	$2^0 = 1$	0000 0001
1	$2^1 = 2$	0000 0010
2	$2^2 = 4$	0000 0100
3	$2^3 = 8$	0000 1000
4	$2^4 = 16$	0001 0000
5	$2^5 = 32$	0010 0000
6	$2^6 = 64$	0100 0000
7	$2^7 = 128$	1000 0000
8	$2^8 = 256$	1 0000 0000

Vocabulaire important :

- un chiffre binaire s'appelle un bit (abrégé anglais de ***bi**nary digi**t***).
- un nombre binaire de 8 bits s'appelle un octet.

1.3 Compter en base hexadécimale - base 16

La base hexadécimale est également très fréquente en numérique car elle permet d'écrire rapidement (en moins de caractères) un équivalent de nombres binaires.

En hexadécimal, il existe 16 caractères : 0 1 3 4 5 6 7 8 9 A B C D E F.

En comptant, après 9, il vient simplement A puis B, puis C, puis D, puis E, puis F et enfin 10 : c'est à dire 1 "*seizaine*" (16^1), et 0 unité (16^0).

Le nombre qui s'écrit 2AD3 s'interprète ainsi :
$(2AD3)_{16} = (2 \times 16^3 + 10 \times 16^2 + 13 \times 16^1 + 3 \times 16^0)_{10}$
$(2AD3)_{16} = (2 \times 4096 + 10 \times 256 + 13 \times 16 + 3 \times 1)_{10} = (10963)_{10}$

L'ensemble des 16 chiffres hexadécimaux peuvent se coder avec 4 bits, et donc un octet s'écrit simplement avec 2 chiffres hexadécimaux, d'où le gain de place lorsqu'on convertit des nombres binaires en hexadécimal.

Exemples :

décimal	binaire	hexadécimal
0	0000 0000	0
1	0000 0001	1
2	0000 0010	2
3	0000 0011	3
4	0000 0100	4
5	0000 0101	5
6	0000 0110	6
7	0000 0111	7
8	0000 1000	8
9	0000 1001	9

décimal	binaire	hexadécimal
10	0000 1010	A
11	0000 1011	B
12	0000 1100	C
13	0000 1101	D
14	0000 1110	E
15	0000 1111	F
16	0001 0000	10
104	0110 1000	68
162	1010 0010	A2
255	1111 1111	FF

1.4 Écriture d'un nombre dans une base b quelconque

Une base b contient b caractères différents.
Un nombre écrit $a_n a_{n-1} \cdots a_1 a_0$ s'interprète ainsi :
$a_n a_{n-1} \cdots a_0 = a_n \times b^n + a_{n-1} \times b^{n-1} + \cdots + a_1 \times b^1 + a_0 \times b^0$

2 Conversions d'une base dans une autre

2.1 Conversion décimal - binaire

On souhaite convertir un nombre N écrit en décimal vers sa représentation binaire.

Méthode :
- Diviser le nombre par 2 et noter le reste de la division (0 ou 1).
- Recommencer avec le quotient de la division précédente.
- Poursuivre ainsi jusqu'à ce que le quotient soit nul.

Le nombre binaire est alors la suite des restes (0 ou 1) obtenus en partant de la dernière division vers la première.

Algorithme :

```
quotient ← N
listeBinaire ← [] # (liste vide)
tant que quotient ≠ 0 faire
    reste ← quotient % 2 # (reste de la division euclidienne)
    insérer reste au début de listeBinaire
    quotient ← quotient // 2 # (quotient de la division euclidienne)
fin tq
renvoyer listeBinaire
```

Exemple : Convertir 652 en binaire :
$652 = 0 + 326 \times 2$
$326 = 0 + 163 \times 2$
$163 = 1 + 81 \times 2$
$81 = 1 + 40 \times 2$
$40 = 0 + 20 \times 2$
$20 = 0 + 10 \times 2$
$10 = 0 + 5 \times 2$
$5 = 1 + 2 \times 2$
$2 = 0 + 1 \times 2$
$1 = 1 + 0 \times 2$

$(652)_{10} = (1010001100)_2$

Algorithme légèrement modifié : On désire obtenir la conversion dans une chaîne de caractères plutôt qu'une liste de bits.

$quotient \leftarrow N$
$strBinaire \leftarrow$ `""` # (chaine vide)
tant que $quotient \neq 0$ **faire**
  $reste \leftarrow quotient \ \% \ 2$ # (reste de la division euclidienne)
  $strBinaire \leftarrow$ **chaine**$(reste) + strBinaire$
  $quotient \leftarrow quotient \ // \ 2$ # (quotient de la division euclidienne)
fin tq
renvoyer $strBinaire$

2.2 Conversion binaire - décimal

Il suffit dans ce cas d'ajouter toutes les puissances de 2 qui sont affectées du bit 1 dans le nombre binaire.

Exemple : Convertir 1010001100 en décimal :

1	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**	**0**
9	8	7	6	5	4	3	2	1	0
2^9	0	2^7	0	0	0	2^3	2^2	0	0
512	0	128	0	0	0	8	4	0	0

$(1010001100)_2 = 2^9 + 2^7 + 2^3 + 2^2 = 512 + 128 + 8 + 4 = (652)_{10}$

2.3 Conversion décimal - hexadécimal

On retrouve le même principe que pour la conversion décimal/binaire (en adaptant évidemment le 2 en 16 !) :

Méthode :
- Diviser le nombre par 16 et noter le reste de la division (de 0 à F)
- Recommencer avec le quotient de la division précédente
- Poursuivre ainsi jusqu'à ce que le quotient soit nul

Le nombre hexadécimal est alors la suite des restes (de 0 à F) obtenus en partant de la dernière division vers la première.

Exemple : Convertir 652 en hexadécimal :
$652 = 12\ (C) + 40 \times 16$
$40 = 8 + 2 \times 16$
$2 = 2 + 0 \times 16$

$(652)_{10} = (28C)_{16}$

2.4 Conversion hexadécimal - décimal

Il suffit d'ajouter toutes les puissances de 16 multipliées par le chiffre hexadécimal qui leur est affecté dans le nombre hexadécimal.

Exemple : Convertir $28C$ en décimal :

2	**8**	**C**
2	1	0
2×16^2	8×16^1	12×16^0
2×256	8×16	12×1
512	128	12

$(28C)_{16} = 512 + 128 + 12 = (652)_{10}$

2.5 Conversion en Python

Python propose de façon native les fonction **bin** et **hex** pour convertir un nombre décimal en binaire ou hexadécimal.

```
>>> bin(652)
'0b1010001100'
>>> hex(652)
'0x28c'
```

Python renvoie une chaîne de caractères préfixée par '0b' pour le binaire et '0x' pour l'hexadécimal.

On peut aussi directement calculer en Python avec des nombres binaires ou hexadécimaux avec les mêmes préfixes (sans chaîne de caractères !). Le résultat s'affiche en décimal.

Exemple (tordu mais contient l'essentiel !) :

```
>>> 0x2a + 0b10 + 5
49
>>> bin(49)
'0b110001'
>>> hex(49)
'0x31'
```