Numération

Partie 1 : les entiers positifs

Objectifs :

Écriture d'un entier positif dans une base b ⩾ 2 (base 2, 10 et 16)			
Décrire les tailles courantes des entiers (8, 16, 32 ou 64 bits)			
Passer de la représentation d'une base dans une autre.			
Évaluer le nombre de bits nécessaires à l'écriture en base 2 d'un entier			

Pour écrire un nombre, nous utilisons la base 10, mais dans un ordinateur la base qui est utilisée est la base 2.
Nous utilisons le binaire car, les systèmes technologiques ont souvent deux états stables.

- Un interrupteur est ouvert ou fermé.
- Une diode est allumée ou éteinte.
- Une tension vaut 0V (bit à 0) ou 5V (bit à 1).

Dans le cas d'un ordinateur, on identifie le bit par la tension V_{CE} entre l'émetteur et le collecteur d'un transistor. Cette tension ne peut prendre que 2 valeurs, 0V ou 5V, en fonction de la valeur de la tension d'entrée.

De plus, on sait depuis Boole grâce à son ouvrage publié en 1853 dans lequel il démontre que tout processus logique peut être décomposé en une suite d'opérations logiques (ET, OU, NON) appliquées sur deux états (ZERO-UN, OUI-NON, VRAI-FAUX, OUVERT-FERME). La base 2 est donc suffisante pour représenter des nombres et des données ainsi que pour représenter des processus logiques. C'est ce qu'utilise un ordinateur ou la logique est représentée par le processeur et les données par la mémoire.

La base 2 contient deux symboles : 0 et 1.

Dans un nombre écrit en binaire chaque chiffre s'appelle un **bit** (**B**inary dig**IT** que l'on peut traduire par chiffre binaire et qui est aussi un jeu de mot avec « bit » dans le sens de peu car c'est la plus petite unité d'une information binaire). Un bit c'est donc 0 ou 1.

Un mot binaire de n bits est un ensemble de n bits.

Par exemple (compléter) :

0111 est un mot de 4 bits 0111 1001 est un mot de …… ? bits

100 1001 est un mot de 7 bits 110 est mot de ……. bits

(Remarque: pour faciliter la lecture, on écrira les mots par bloc de 4 bits)

I (Re)découvrons la base 10

La base 10 contient dix symboles, les chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Pour chaque rang, il y a dix chiffres possibles.

Le chiffre le plus à droite correspond aux unités, puis, au fur et à mesure qu'on se décale vers la gauche, le chiffre correspond à celui d'une puissance de 10 supérieure.

Dans 2019, le chiffre 2 est celui des milliers, c'est-à-dire 10^3.

Le nombre 2019 s'écrit :

Rang	5	4	3	2	1	0
Base^{rang}	10^5	10^4	10^3	10^2	10^1	10^0
Valeur	100 000	10 000	1000	100	10	1
Nombre	0	0	2	0	1	9

Soit 2019 =

$= 2\times1\,000+1\times10+9\times1$

L'ordre de grandeur de ce nombre est $2\,000=2\times10^3$. C'est le chiffre le plus à gauche multiplié par la puissance de 10 qui lui est associée.

Indicateur de base

Pour écrire un nombre en base 10 et le distinguer des nombres écrits dans d'autres bases, il suffit de l'écrire comme à l'habitude, il n'y a pas d'indicateur de base: 2019.

II La base 2 : le binaire

1 L'écriture binaire

En base 2, il n'y a que deux chiffres 0 et 1. Pour chaque rang il n'y a donc que deux chiffres possibles. Ainsi le nombre 2019 s'écrit en binaire :

Rang	10	9	8	7	6	5	4	3	2	1	0
Base^{rang}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Valeur	1024	512	256	128	64	32	16	8	4	2	1
Nombre	1	1	1	1	1	1	0	0	0	1	1

Soit 2019 =

$= 1\times1024+1\times512+1\times256+1\times128+1\times64+1\times32+1\times2+1\times1$

Indicateur de base

Pour écrire un nombre en base 2 et le distinguer des nombres écrits dans d'autres bases, on écrit :

$$2019 = 111\ 1110\ 0011_2$$

En Python on écrira : **0b**111 1110 0011 (avec **b** comme **binaire**)

Exemple

```
>>> 0b110
6
```

Donc $110_2 = 6$

En binaire, 2019 s'écrit sur 11 bits. Si le nombre est stocké en mémoire sur 16 bits, on le complète par des 0 à gauche.
Sur 16 bits : $2019 = 0000\ 0111\ 1110\ 0011_2$

2 Nombres de possibilités et valeur maximale.

Les calculateurs électroniques n'utilisent pas tous le même nombre de bits. Cela dépend du processeur utilisé (actuellement, 64 bits).
Pour connaître le nombre de combinaisons qu'il est possible d'écrire (de coder) avec n bits, on utilise cette formule : `nombre de combinaison = 2^n.`
Pour connaître la valeur maximum que l'on peut écrire avec n bits **pour un nombre entier positif**, on utilise cette formule : `valeur maximum = 2^n-1.`
Sur n bits on peut représenter les nombres de 0 à 2^n-1.

Avec … bits,		1	2	4	8
Il y a … combinaisons possible					
La valeur maximum est	En binaire				
	En décimal				

3 Les préfixes

Lorsqu'on manipule des grands nombres, on utilise des préfixes. Les tableaux ci-dessous récapitulent ceux des puissances positives :

- En base 10 :

Puissances	10^3	10^6	10^9	10^{12}	10^{15}	10^{18}
Nom	Kilo	Méga	Giga	Tera	Peta	Exa
Abréviation	k	M	G	T	P	E

Remarque : le k de kilo est un k minuscule.

- En base 2 :

Puissances	2^{10}	2^{20}	2^{30}	2^{40}	2^{50}	2^{60}
Nom	Kibi	Mébi	Gibi	Tébi	Pébi	Exbi
Abréviation	Ki	Mi	Gi	Ti	Pi	Ei

Remarque : le K de Kibi est un K Majuscule.

/!\ Attention à ne pas confondre les préfixes décimaux avec les préfixes binaires. /!\
Un abus de langage et d'utilisation a longtemps fait correspondre 1 Kilobits = 1024 bits. Or, ils ne sont pas égaux car $10^3 \neq 2^{10}$. De fait, 1 Kilobits ≠ 1 Kibibits
Les préfixes de la base 10 **ne peuvent pas** être utilisés pour la base 2. Cela est une norme du **B**ureau **I**nternational des **P**oids et **M**esures, décrit dans les documents du **S**ytème **I**nternational d'unité, chapitre 3 : https://www.bipm.org/utils/common/pdf/si-brochure/SI-Brochure-9-FR.pdf

III Une autre base : la base 16 ou hexadécimale

La base 16 ou hexadécimale utilise 16 symboles : Les chiffres de 0 à 9 et les lettres A, B, C, D, E et F.

Elle est très utilisée par exemple pour coder les couleurs dans une page web en html-CSS ou encore la représentation d'une adresse IP pour la norme V6. Elle a l'avantage d'être « compacte » à l'écriture contrairement au binaire et $16 = 2^4$ la conversion entre les bases est donc facilitée.

Voir le tableau ci-contre.

Base 10	Base 2	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Exemple :
Le nombre 2019 en base 16 s'écrit :

Rang	4	3	2	1	0
Base^{rang}	16^4	16^3	16^2	16^1	16^0
Valeur	65 536	4096	256	16	1
Nombre	0	0	7	E	3

Soit 2019 =

= 7·256 + E·16 + 3·1

=

Indicateur de base

Pour écrire un nombre en base 16 et le distinguer des nombres écrits dans d'autres bases, on écrit : $2019 = 7E3_{16}$
En Python on écrit : **0x**7E3 (avec **x** comme he**x**adécimal)
Exemple

```
>>>0x1F
31
```

On a donc : $1F_{16} = 31$

IV Conversions

1 Conversions d'un nombre entier de la base 2 à la base 10

Un nombre en binaire sur 8 bits peut s'écrire :

128	64	32	16	8	4	2	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
b7	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**

Poids décimal de chaque bit

Nombre binaire

On peut écrire la conversion :

$$Y_{(10)} = b_7\ b_6\ b_5\ b_4\ b_3\ b_2\ b_1\ b_{0\,(2)} = b_7 * 2^7 + b_6 * 2^6 + b_5 * 2^5 + \ldots\ldots + b_0 * 2^0$$

Exemple :

128	64	32	16	8	4	2	1
0	0	1	0	1	0	0	1

$$0010\ 1001_{\ 2} = 32 + 8 + 1 = 41$$

Exercice : Donner les valeurs décimales des nombres binaires suivants :

128	64	32	16	8	4	2	1
1	0	0	1	1	1	0	0

128	64	32	16	8	4	2	1
0	1	0	1	0	0	1	1

128	64	32	16	8	4	2	1
1	0	1	0	0	1	0	0

128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1

2 Conversions de la base 10 vers la base 2

Dans un mot binaire on repère deux bits importants :

Le bit de poids fort → 1011 1100 ← le bit de poids faible
MSB : Most Significant Bit **LSB :** less Significant Bit

Méthode par poids

Le principe est d'additionner les poids (en commençant par le plus fort possible) afin d'obtenir le nombre décimal voulu.
Le nombre *n* de chiffres nécessaire pour écrire le nombre *a* est tel que $2^n \geq a$. Par exemple $2^6 = 64 \leq 67 \leq 128 = 2^7$ donc il faut au minimum 7 chiffres pour écrire 67 en binaire.

Remarque : On peut démontrer qu'en suivant cette méthode il n'y a qu'un seul résultat possible.

Exemple : Convertir convertir 67 en binaire.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

128> 67 est trop grand
64 : ok je mets « 1 » sous le 64. Il manque 3 pour obtenir 67
64 + 32 = 96 → trop grand
64 + 4 = 68 → trop grand
64 + 2 = 66 → ok, je mets « 1 » sous le 2. Il manque 1 pour obtenir 67.
66 + 1 = 67 → ok, je mets « 1 » sous le poids 1
Je complète par des 0.

J'obtiens donc : $67 = 0100\ 0011_2$

Exercice :
S'entraîner à convertir par cette méthode :

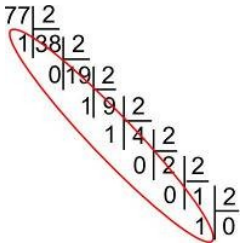
25, 36, 155, 128, 200, 840

Éléments de corrections :

25 : 0001 1001
36 : 0010 0100

Méthode par divisions successives par la base

Cette méthode consiste à diviser autant de fois que nécessaire le nombre décimal par 2 jusqu'à ce que le nombre restant à diviser soit inférieur à 2. Puis, en remontant depuis le résultat de la dernière division jusqu'au reste de la première, on prend les restes de chaque division, pour obtenir le nombre. Dans cette méthode, on divise successivement le nombre à convertir par 2 (car base 2) .
/ ! \ Attention au sens de lecture (du bas (MSB) vers le haut (LSB)).



Exemple

On a donc : $77 = 100\ 1101_2$

Exercice :
Convertir 56 et 107 en base 2 par la méthode de la division par la base
vérifier que vous trouvez bien : $0011\ 1000_2$ (pour 56) et 01101011_2 (pour 107)

3 Conversions de la base 10 à la base 16

La meilleure méthode est celle par **divisions successives**, on procède comme pour passer de la base 10 à la base 2 sauf que l'on divise par 16.

Exemple :
Pour convertir 429 en hexadécimal on effectue les divisions comme ci-contre.
Puis on lit en remontant les restes.
Comme 10 = A_{16} et 13 = D_{16}, on obtient :
429 = AD_{16}

429	16	
13	26	16
	10	1

4 Conversions de l'hexadécimal vers la base 10

La meilleure méthode est celle de la **méthode par poids.**

Exemple
Soit le nombre 0xBB06. On établit le tableau des poids et l'on y place chaque chiffre au rang correspondant. Puis, on effectue les additions pondérées.
0xBB06=11·4096+11·256+6·1=47878

Rang	3	2	1	0
Base^{rang}	16^3	16^2	16^1	16^0
Valeur	4096	256	16	1
Nombre	B	B	0	6

5 Conversions entre binaire et hexadécimal

La conversion est directe en utilisant le tableau ci-contre et en regroupant les bits par quatre ce que l'on appelle un quartet.

Base 10	Base 2	Base 16
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Exemples

- `0b100010100010 = 0b 1000 1010 0010 = 0x8A2`
- `0xFADE= 0b1111 1010 1101 1110= 0b1111101011011110`

S'entraîner

Convertir en binaire

`0xA4, 0xF1`

Convertir en hexadécimal

`0b11100110 0b10101111`