➢ Bits, octets, mots

Un ordinateur travaille en binaire, un chiffre (0 ou 1) est alors appelé un bit (**bi**nary digi**t**)
Un regroupement de 8 bits est appelé un octet (ou byte) : de 00000000 à 11111111 (c-à-d : de 0 à 255)
Le bit le plus à droite est appelé bit de poids faible (lsb), et celui le plus à gauche, bit de poids fort (msb).
Suivant la puissance de l'ordinateur, celui-ci travaille sur des mots plus ou moins grands…
(32 bits : mots de 4 octets, 64 bits : mots de 8 octets)

Unités de octets v · d · m

Ordre de grandeur	Système international (SI)			Préfixes binaires		
	Unité	Notation	Valeur	Unité	Notation	Valeur
1	**octet**	o	1 octet	**octet**	o	1 octet
10^3	kilooctet	ko	10^3 octets	kibioctet	Kio	2^{10} octets
10^6	mégaoctet	Mo	10^6 octets	mébioctet	Mio	2^{20} octets
10^9	gigaoctet	Go	10^9 octets	gibioctet	Gio	2^{30} octets
10^{12}	téraoctet	To	10^{12} octets	tébioctet	Tio	2^{40} octets
10^{15}	pétaoctet	Po	10^{15} octets	pébioctet	Pio	2^{50} octets

➢ Multiples normalisés (Source : Wikipédia)

Remarque : $2^{10} = 1024 \approx 10^3$.

➢ Entiers : courts ou longs, signés ou non signés

Dans beaucoup de langages informatiques il existe différentes sortes d'entiers
Les entiers courts sont codés sur 16 bits : de 0 à 65 535 s'ils ne sont pas signés (entiers naturels)
de – 32 768 à 32 767 s'ils sont signés (entiers relatifs)
Les entiers longs sont codés sur 32 bits : de 0 à 4 294 967 295 s'ils ne sont pas signés
de – 2 147 483 648 à 2 147 483 648 s'ils sont signés
ou sur 64 bits : de 0 à 18 446 744 073 709 551 615 s'ils ne sont pas signés
de – 9 223 372 036 854 775 808 à 9 223 372 036 854 775 808 sinon
Remarque : Cette distinction n'existe pas en Python, seule la capacité mémoire limite la taille des entiers.

➢ Représentation des entiers négatifs en binaire

Pour représenter un entier négatif, on aurait pu simplement utiliser le premier bit pour indiquer si c'est un nombre positif ou un nombre négatif, mais cela n'est pas pratique dans les calculs !
On utilise alors le « complément à 2 » : Pour les entiers courts par exemple (sur 16 bits), les entiers positifs (de 0 à 32 767) sont codés normalement (le 1er bit est alors 0), pour les entiers négatifs, on remplace chaque bit par son complément à 2 (0 devient 1, et 1 devient 0), puis on ajoute 1 (le 1er bit est alors 1). Remarque : La somme d'un entier et de son opposé donne bien 0 dans ce cas.
Attention : Si n est négatif, il est alors codé par le nombre $2^{16} - n$.
Exemple : $00000111\ 10011001_{b2} = 1\ 945_{b10}$ et $11111000\ 01100111_{b2} = -\ 1\ 945_{b10}$ en binaire signé.

➢ Représentation des nombres à virgule flottante en binaire (Norme IEEE754)

Pour représenter un nombre à virgule, il existe plusieurs formats : simple précision (32 bits), double précision (64 bits), double précision étendue (80 bits). On utilise le modèle suivant :

1 bit de signe	e bits d'exposant					m bits de mantisse				
			…					…		

Le nombre codé vaut alors : signe × (1 + mantisse) × $2^{\text{exposant} - \text{décalage}}$, où décalage = $2^{e-1} - 1$.
Remarque : en 32 bits : $e = 8$ et $m = 23$ (décalage = 127) , en 64 bits : $e = 11$ et $m = 52$.
Exemple : Sur 32 bits l'expression 0 10000000 10010001111010111000010 correspond à

- bit de signe « 0 » donc un nombre positif, signe = 1.
- exposant « 10000000 » donc $2^7 = 128$ soit après décalage : $2^{\text{exposant} - \text{décalage}} = 2^1$
- mantisse « 10010001111010111000010 » donc

$1 + 1\times2^{-1} + 0\times2^{-2} + 0\times2^{-3} + 1\times2^{-4} + 1\times2^{-5} + \ldots + 1\times2^{-22} + 0\times2^{-23} \approx 1{,}5699999332427979$
D'où $N = 2 \times 1{,}5699999332427979 \approx 3{,}14$ à 10^{-6} près