# Codage des nombres et des objets

Laurent Signac – CC-BY-SA – 05-09-23 1100 d54ee684a37bea77215f

## 1 Généralités

L'information est la matière première de l'informatique.

Un **programme**, constitué d'**informations**, stocke, manipule, transforme, d'autres informations, à l'aide de **machines**

Toutes ces informations sont à un moment représentées par des nombres (séquence de 0 et 1).

Le binaire est la base la plus simple utilisable et elle est adaptée à l'électronique.

L'unité d'information est le *bit* : 0 ou 1. Un groupe de 8 bits constitue un octet (*byte* en anglais). Il y a deux systèmes de multiples en vigueur ($\times 10^3$ et $\times 2^{10}$) utilisant des préfixes ridicules.

Multiples du système international :

- kilo-octet (ko) : $10^3$ octets
- méga-octet (Mo) : $10^6$ octets
- giga-octet (Go) : $10^9$ octets
- tera-octet (To) : $10^{12}$ octets
- péta-octet (Po) : $10^{15}$ octets

Norme historique[1] :

- kibi-octet (Kio) : $2^{10}$ octets
- mébi-octet (Mio) : $2^{20}$ octets
- gibi-octet (Gio) : $2^{30}$ octets

L'unité de débit est le bits / s (et pas octets / s).

## 2 Valeur d'un nombre écrit en base $b$

En base $b$, on utilise exactement $b$ symboles (les chiffres) : - base 10 : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 - base 2 : 0, 1 - base 16 : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

L'écriture en base $b$ : $c_n c_{n-1} ... c_1 c_0$ représente un nombre qui a pour valeur :

$$v(c_n) \times b^n + v(c_{n-1}) \times b^{n-1} + ... + v(c_1) \times b^1 + v(c_0) \times b^0$$

avec $v(c_k)$ la valeur du chiffre $c_k$.

Exemple : le nombre $5A_{16}$ est ici donné en base 16. Quelle sont sa valeur et son écriture en base 10 ?

$$v(5) \times 16^1 + v(A) \times 16^0 = 5 \times 16 + 10 = 90$$

## 3 Écriture d'un nombre dont on connaît la valeur

En effectuant des divisions successives de $n$ par la base $b$, la séquence des restes donne la valeur de chaque chiffre de l'écriture de $n$ en base $b$.

Exemple : Comment s'écrit 90 en base 2 ?

[1] https://physics.nist.gov/cuu/Units/binary.html

- 90 divisé par 2 donne 45, reste 0
- 45 divisé par 2 donne 22, reste 1
- 22 divisé par 2 donne 11, reste 0
- 11 divisé par 2 donne 5, reste 1
- 5 divisé par 2 donne 2, reste 1
- 2 divisé par 2 donne 1, reste 0
- 1 divisé par 2 donne 0, reste 1

L'écriture est donnée par la séquence des restes, le premier reste obtenu étant le chiffre de poids faible (le plus à droite) :

$$90_{10} = 1011010_2$$

# 4 Nombres à virgule

## 4.1 Base 2 → base 10

$$\begin{aligned} 100,011_2 &= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + \\ &\quad 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} \\ &= 4,375 \end{aligned}$$

## 4.2 Base 10 → base 2

$$\begin{aligned} 0,6875 &\times 2 = 1,375 \\ 0,375 &\times 2 = 0,75 \\ 0,75 &\times 2 = 1,5 \\ 0,5 &\times 2 = 1 \end{aligned}$$

$$0,6875_{10} = 0,1011_2$$

# 5 Développements binaires infinis

En base $b$, un nombre $\frac{p}{q}$ (fraction irréductible) a une écriture finie si tous les facteurs premiers de $q$ sont aussi des facteurs premiers de $b$. En binaire, seuls les nombres de la forme $\frac{p}{2^k}$ ont une écriture finie.

Certains nombres ont une écriture infinie en binaire. Par exemple :

$$0,2_{10} = 0,001100110011..._2 = 0,\overline{0011}_2$$

# 6 Nombres négatifs

Une fois choisi le nombre de bits du codage ($n$), on représente les entiers compris entre 0 et $2^{n-1}-1$ comme d'habitude, et on choisit comme code d'un entiers $-k$ compris entre $-2^{n-1}$ et $-1$ : l'écriture binaire de $2^n - k$ (qui s'appelle le complément à 2 ou complément à $2^n$).

Le complément à 2 est involutif.

Par exemple, si on code en complément à 2 sur 8 chiffres :

- 33 est codé tel quel puisqu'il est positif : 00100001
- $-33$ est codé par la représentation binaire de $2^8 - 33 = 223$, c'est à dire : 11011111

Il y a d'autres moyens de trouver le complément à 2 (remplacer les 0 par des 1 et ajouter 1, par exemple).

# 7 Codage des nombres à virgule

## 7.1 Codage en virgule fixe

- Retenir un nombre fixe de chiffres.
- Garder simplement un nombre fixe de bits avant et après la virgule.
- L'espace entre 2 nombres qui se succède est toujours le même.
- **Problème** : l'erreur relative peut être grande

## 7.2 Codage en virgule flottante

- Retenir un nombre fixe de **chiffres significatifs**
- Les petits nombres seront plus serrés que les grands
- L'erreur relative est **tolérable**

Pour en savoir plus sur les erreurs de calculs en flottants : `https://0.30000000000000004.com`[2]

## 7.3 Calculs faux

Les calculs en virgule flottante (ou virgule fixe) donnent le plus souvent un résultat approché. Par exemple :

```
>>> 0.3 - 0.2 - 0.1
-2.7755575615628914e-17
```

Ces erreurs de calculs depuis les débuts de l'informatique ont parfois eu des conséquences graves.

# 8 Codage des couleurs

La couleur sur un écran utilise la synthèse additive, et les couleurs de base rouge, vert, bleu. Une couleur est codée comme la suite des intensités des 3 couleurs de base.


Figure 1: Les filtres rouge, verts, bleus placés devant les pixels d'un écran sont très visibles avec un microscope

Avec 3 leds de couleurs (rouge, verte, bleue) dont l'intensité ne varie pas (on a juste allumé ou éteint), on a 8 combinaisons qui vont de tout éteint (noir) à tout allumé (blanc). Ces 8 combinaisons peuvent êtres codées exactement par 3 bits qui indiquent si chacune des leds est allumée ou non. Ainsi, 000 serait le code du noir et 111 celui du blanc.

# 9 Codage du texte

Le code ASCII ne contient que 128 caractères.

Il y a donc des extensions à l'ASCII qui ajoutent des bits d'information pour avoir des caractères en plus. Exemples : latin-1 (iso-8859-1). Il y a des extensions pour le chinois, japonais, cyrillique, arabe, etc…

[2] https://0.30000000000000004.com/

La norme Unicode recense *tous* les caractères dans une table qui en compte plusieurs dizaines de milliers. Ces codes sont ensuite représentés, en utilisant un encodage comme UTF-8 ou UTF-16

Python utilise Unicode nativement. L'encodage à utiliser pour l'avenir est UTF-8.

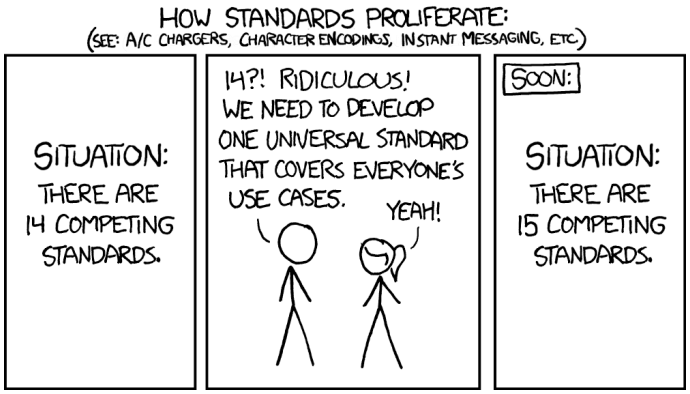


Figure 2: https://xkcd.com/927/

## 10 Codage des sons

Un son stéréo est composé de deux courbes. Chaque courbe est échantillonnée dans le temps à une certaine fréquence et quantifiée en valeur. Le son est donc la donnée de deux séries de nombres codés comme des entiers ou des flottants.

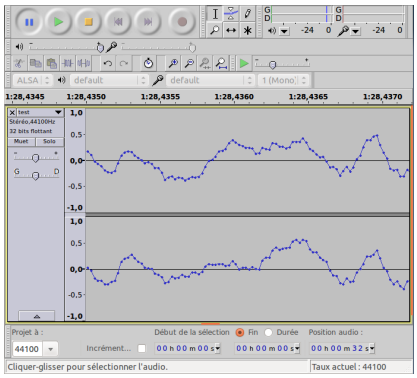

Figure 3: Échantillons sonores dans *Audacity*

C'est de cette manière qu'est numérisé la musique sur un CD Audio, avec une fréquence d'échantillonnage fixe de 44100Hz.

## 11 Image Bitmap

Une image bitmap est un tableau bidimensionnels de nombres (on dit pixel). Chaque pixel est composé de composantes (généralement 3 (RVB) ou 4 (RVBA)). Chaque composante est représentée par un nombre, souvent sur 1 octet. Une image est donc une suite de nombres.

## 12 Codage symbolique

Plutôt que d'échantillonner et quantifier, les codages symboliques cherchent à représenter l'information utile :

- les formes sur l'image plutôt que les pixels
- les hauteurs des notes jouées plutôt que des échantillons sonores

Le codage symbolique est situé à *un niveau supérieur d'abstraction.*

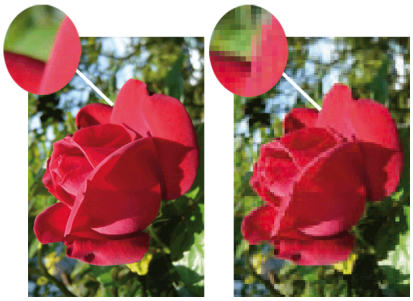

Figure 4: Image bitmap d'une rose dans deux définitions différentes