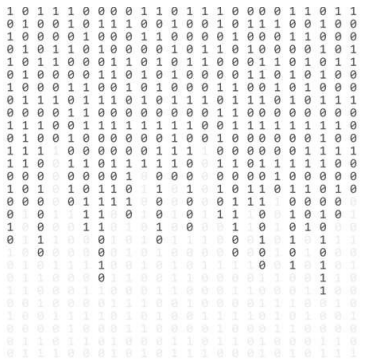
Cours – Le binaire et l'hexadécimal

Objectifs :

- ⇨ Découvrir les écritures binaire et hexadécimale
- ⇨ Savoir écrire un nombre dans une base quelconque
- ⇨ Savoir convertir un nombre d'une base quelconque vers 10 et inversement
- ⇨ Ecrire des programmes de conversion vers ou depuis la base 10

Blague d'informaticien :

« Il y a exactement 10 sortes de personnes : celles qui comprennent le binaire et celles qui ne le comprennent pas. »



I – Introduction – L'élément binaire ou bit

Nous avons déjà vu que les ordinateurs, pour des raisons de conception interne fonctionnaient en binaire. Les informations stockées et manipulées par l'ordinateur seront donc constituées d'éléments binaires (eb en français) couramment appelés bit (Binary digIT en anglais).

1 bit = 1 élément binaire : 2 valeurs possibles 0 ou 1

Le binaire possède donc 2 chiffres : « 0 » et « 1 » pour former des nombres.

Le type booléen (`bool` en python) correspond donc à un bit (0 pour `False` et 1 pour `True`).

II – Bases de numération

1) Définition

Les **nombres** s'écrivent avec des **chiffres** tout comme les mots s'écrivent avec des lettres.
Tout comme il existe des mots qui s'écrivent avec une seule lettre, il existe des nombres qui s'écrivent avec un seul chiffre.
La liste de chiffres 2739 désigne le nombre $2739 = 2\times10^3 + 7\times10^2 + 3\times10^1 + 9\times10^0$

Il s'agit de l'écriture habituelle des nombres basée sur une numération de position. Elle a été mise au point en Inde, développée au Moyen Orient (notamment par Al-Kharizmi) et est arrivée en Europe au moyen-âge (probablement par l'Espagne musulmane)
Le nombre **dix** y joue un rôle particulier, puisqu'on compte des puissances de dix et parce qu'on utilise dix chiffres (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
Cette manière d'écrire les nombres est appelée **écriture décimale** ou **écriture en base 10**

On peut généraliser ce principe pour écrire des nombres entiers en **base b** (où b désigne un nombre entier naturel supérieur ou égal à 2).
Ainsi, $(a_n\ a_{n-1}\ \dots\ a_2\ a_1\ a_0)_b = a_n\times b^n + a_{n-1}\times b^{n-1} + \dots + a_2\times b^2 + a_1\times b^1 + a_0\times b^0$
où les chiffres a_n , a_{n-1} , … , a_2 , a_1 et a_0 sont compris entre 0 et b-1 (inclus)

On écrit ce nombre entre parenthèses en mettant la base b en indice après la parenthèse. Par exemple le nombre 89 en base 10 s'écrit $(89)_{10}$, le nombre 521 en base 8 s'écrit $(521)_8$.

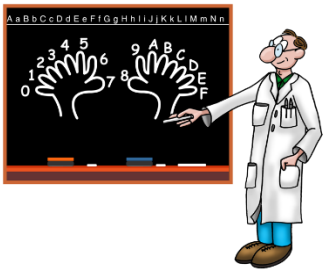
Licence Creative Commons BY-SA

2) Passage d'une base à l'autre

a. De la base n vers la base 10

Prenons l'exemple de la base 8 (octal). Les chiffres sont alors « 0 », « 1 », « 2 », « 3 », « 4 », « 5 », « 6 » et « 7 » (il y a bien 8 symboles).
Le nombre $(45712)_8$ en base 8 peut s'écrire $4\times8^4 + 5\times8^3 + 7\times8^2 + 1\times8^1 + 2\times8^0 = (19402)_{10}$

Application n°1 :
Ecrire le nombre $(1403)_5$ en base 10.



- **Ecriture hexadécimale** ou écriture en base 16 (utilisée en informatique)

On considère **l'adresse MAC d'une carte réseau** : 10 : 93 : E9 : 0A : 42 : AC

C'est une liste de 6 nombres de 2 chiffres écrits en base 16.
Les seize chiffres autorisés sont 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A (pour 10), B (pour 11), C (pour 12), D (pour 13), E (pour 14) et F (pour 15).

Exemples :
$(10)_{16} = 1\times16 + 0\times1 = 16$
$(E9)_{16} = 14\times16 + 9\times1 = 233$
$(42)_{16} = 4\times16 + 2\times1 = 66$
$(93)_{16} = 9\times16 + 3\times1 = 147$
$(0A)_{16} = 0\times16 + 10\times1 = 10$
$(AC)_{16} = 10\times16 + 12\times1 = 172$

Ainsi, l'écriture en base 10 de l'adresse MAC précédente serait 16 : 147 : 233 : 10 : 66 : 172

La notation hexadécimale est assez utilisée en informatique, par exemple pour désigner une couleur en HTML ou dans le module `turtle` en python. Par exemple, la couleur `'sienna'` a pour valeur `#A0522D` (le signe « # »indique que la valeur qui suit est en hexadécimal. Les deux premiers chiffres représentent la valeur de la composante rouge, les deux suivant celle de la composante verte et enfin les deux derniers la composante bleue.

Application n°2 :
1) Saisir « cmd » dans le menu démarrer. Une fois l'invite de commandes ouverte, saisir la commande `ipconfig /all`. Lire l'adresse MAC (aussi appelée « adresse physique ») de la carte réseau de l'ordinateur (il y en a parfois plusieurs), puis en déterminer à la main l'écriture décimale (la calculatrice de Windows en mode « programmeur » permet de vérifier les résultats).

2) Calculer le nombre d'adresses MAC possibles. L'affirmation « Chaque carte réseau possède une adresse MAC unique au monde » paraît-elle réaliste ?

- **Ecriture binaire** ou écriture en base 2 (fondamentale en informatique)

Les deux chiffres autorisés sont 0 et 1.
La capacité de mémoire d'un composant électronique pouvant adopter 2 états, elle correspond à un bit (voir plus haut)

Licence Creative Commons BY-SA

En regroupant plusieurs circuits, on obtient des **mots** écrits avec plusieurs bits qui correspondent donc à des nombres écrits en base 2.

L'**octet** (ou Byte en anglais, à ne pas confondre avec le bit) est un mot de ..... .
L'octet $(10001101)_2$ désigne le nombre 141 car $(10001101)_2 = 1\times2^7 + 0 + 0 + 0 + 1\times2^3 + 1\times2^2 + 0 + 1\times2^0 = 141$
Un octet peut désigner tous les nombres entiers compris $0 = (00000000)_2 = (00)_{16}$ et $255 = (11111111)_2 = (FF)_{16}$, il y a donc 2^8=**256 valeurs possibles**.

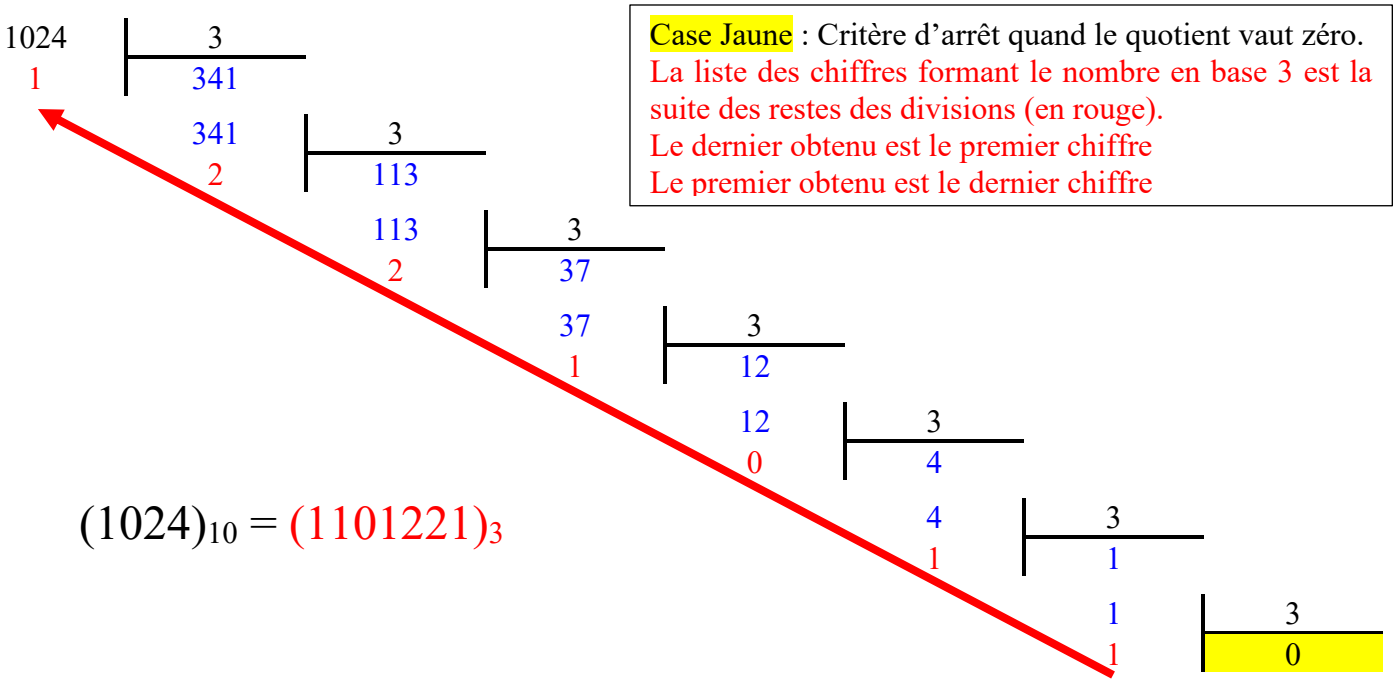
En prenant des mots suffisamment longs, on peut représenter directement tous les nombres entiers naturels, mais aussi (de manière indirecte grâce à des codages adaptés) des nombres entiers relatifs, des nombres décimaux, des lettres, des phrases, des textes, des images, des sons et des films.

Application n°3 :
1) Ecrire le nombre $(101101)_2$ en base 10.

2) Ecrire un programme en python capable de convertir un nombre binaire entré par l'utilisateur en base 10.
Aide : L'opérateur puissance se note `**` en python (ainsi a^b s'écrit `a**b`).

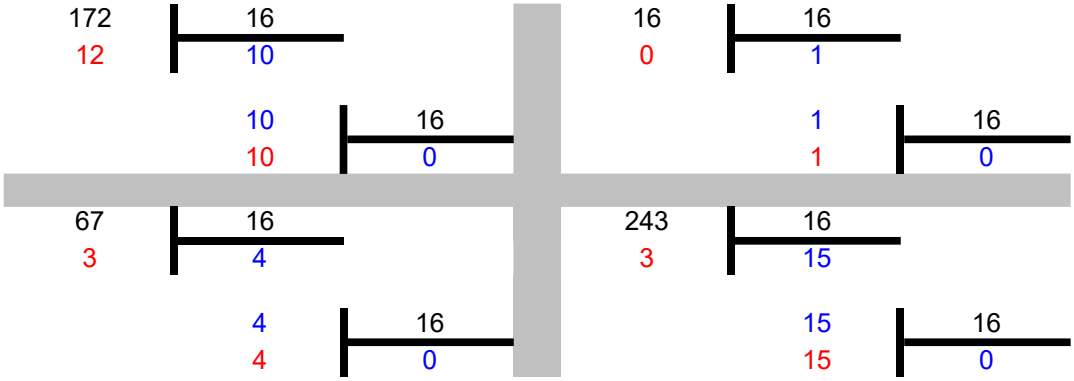
b. De la base 10 vers la base n

Si on souhaite écrire $(1024)_{10}$ en base b=3, il suffit d'effectuer une série de divisions euclidiennes (avec quotient et reste). Le diviseur est toujours b (ici 3). La liste des dividendes est formé par le nombre à convertir et les quotients successifs, jusqu'à obtenir un quotient nul.
La liste des restes obtenus (qui sont bien compris entre 0 et b-1=2) donne les chiffres de l'écriture du nombre en base b.



Licence Creative Commons BY-SA

On considère l'adresse IPv4 172.16.67.243
Elle est constituée de 4 nombres compris entre 0 et 255 (donc codés chacun par 1 octet).
Elle permet d'identifier un ordinateur sur un réseau pour une période donnée (contrairement à l'adresse MAC, l'adresse IP est temporaire)
Un octet se codant par 2 chiffres en écriture hexadécimale, une adresse IPv4 peut être donnée sous la forme de 4 nombres de 2 chiffres.



Ainsi, $(172)_{10} = (AC)_{16}$ avec A qui correspond à 10 et C qui correspond à 12
$(16)_{10} = (10)_{16}$ et plus généralement le nombre b s'écrit $(10)_b$ en base b
$(67)_{10} = (43)_{16}$
$(243)_{10} = (F3)_{16}$ avec F qui correspond à 15

L'adresse IP sous forme hexadécimale est donc AC.10.43.F3

Application n°4 :
1) Ecrire le nombre $(77)_{10}$ en hexadécimal et en binaire.

2) La couleur `'pink'` de `turtle` correspond à des valeurs R/G/B de 255 / 192 / 203. Donner la valeur hexadécimale de cette couleur.

3) Intérêt de la base 16

Dans un ordinateur, la mémoire est organisée en octets qui contiennent 8 éléments binaires (8 bits). Les processeurs qui équipaient les premiers ordinateurs personnels étaient des processeurs « 8 bits » car ils traitaient les données et faisaient les opérations arithmétiques sur des nombres de 8 bits. De nos jours les processeurs des ordinateurs personnels sont généralement 64 bits. Ils traitent donc 8 octets simultanément.

Application n°5 :
1) Combien de bits sont nécessaires pour coder un chiffre hexadécimal ? Quelle est la valeur en binaire du chiffre hexadécimal F ?

2) Combien de bits y a-t-il dans un octet ? En déduire combien de chiffres hexadécimaux sont nécessaires pour représenter un octet. Quel est alors l'intérêt de la notation hexadécimale ?

Licence Creative Commons BY-SA

III – Travail de programmation

1) Ecrire une fonction `base10_vers_binaire(n)` qui renvoie la chaîne de caractère correspondant à la conversion de l'entier `n` en binaire. Par exemple `base10_vers_binaire(19)` doit renvoyer la chaîne `'10011'`. Si besoin, s'inspirer de l'algorithme proposé plus bas.
2) Ecrire une fonction `base10_vers_n(nombre, n=8)` qui renvoie la chaîne correspondant à la conversion du nombre `nombre` en un nombre en base `n` (`n` étant un entier de 2 à 9).
   Exemple : `base10_vers_n(169, n=5)` renverra `'1134'`.
3) Même question que précédemment mais pour un nombre jusqu'à la base 20 (voir aide plus bas).

Tester vos fonctions avec les exemples du cours et/ou la calculatrice de Windows en mode programmeur.

Aides :

- Pour inverser facilement une chaîne de caractères `chaine` en python, on peut faire : `chaine = chaine[::-1]`
  Exemple : si `test` vaut `'azerty'`, alors `test[::-1]` vaudra `'ytreza'`.

- Si on veut le caractère (str) `c` correspondant au chiffre (int) `n`, on peut écrire :
  `caractere = str(n)` si $0 \leq n < 10$ ou `str(chr`[1]`(ord('A')+ (n-10)))` pour $n > 9$
  Exemple : si `n` vaut `3`, alors `caractere` vaudra `'3'` (le caractère « 3 »).

Algorithme pour la question 1 :

FONCTION decimalVersBase2(nbDecimal : *entier*)
    nombre ← chaîne vide
    dividende ← nbDecimal
    TANT QUE dividende < > 0 FAIRE
        a_concatener ← reste de la division de dividende par 2
        nombre ← concaténation de nombre et de la chaîne correspondant au nombre a_concatener
        dividende ← division entière de dividende par 2
    FIN TANT QUE
    nombre ← chaine_renversée(nombre)
    RENVOYER nombre
FIN FONCTION

[1] Nous reverrons les fonctions `chr` et `ord` dans un prochain chapitre.

Licence Creative Commons BY-SA