Leçon n°3 : Codage binaire et codage hexadécimal, calcul en binaire et codage des entiers relatifs.

1. Codage binaire et codage hexadécimal.

A. Un exemple d'application : le système colorémétrique RVB.

On appelle systéme RVB (ou RGB en anglais) un système de codage numérique des couleurs par addition des 3 couleurs primaires : Rouge, Vert et Bleu. Ce système est utilisé pour la représentation des couleurs pour les écrans.
Pour l'impression on utilise le système CYMK ou CMJN (Cyan, Magenta, Jaune, Noir) qui est basé sur le principe de soustraction des couleurs, plus efficace dans le monde de l'imprimerie.
Le système RVB est codé sur 3 octects, chaque couleur primaire étant codée sur un octet.

Exercice 1

1. Dans le système RVB, quelles sont les valeurs décimales possibles pour chaque canal de couleur ?
2. Combien de couleurs différentes possibles a-t-on avec le système RVB ?
3. En HTML, CSS ou SVG, on peut définir la couleur d'un élément à l'aide de la propriété :

   ```
   color : rgb(100%,80%,60%)
   ```

   soit en écriture décimale :

   ```
   color : rgb( ___ , _____ , ___ )
   ```

4. Déterminer les codes RVB correspondants aux couleurs noire et blanche :

   ```
   color : rgb( ___ , ___ , ___ ) /* codage couleur du noir en % */
   color : rgb( ___ , ___ , ___ ) /* codage couleur du noir en décimal */
   color : rgb( ___ , ___ , ___ ) /* codage couleur du blanc en % */
   color : rgb( ___ , ___ , ___ ) /* codage couleur du blan en décimal */
   ```

B. Passage du codage binaire au codage en héxadécimal.

Considérons le nombre 255, qui se code en binaire sur un octet par : $\underbrace{11111111}_{8\ bits}$

On peut décomposer cette écriture en deux blocs de 4 bits : $\underbrace{11111111}_{4\ bits\ 4\ bits}$

On peut coder facilement chacun de ces deux blocs en hexadécimal :

1111 représente le nombre _ _ en décimal, soit le nombre _ en hexadécimal.

On a alors : $\underbrace{11111111}_{255\ en\ décimal} = \underbrace{ff}_{en\ hexadécimal}$

Exercice 2

1. Donner l'écriture en héxadécimal des nombres 204 et 153.
2. Expliquer pour les 2 codes couleurs suivants sont identiques :

```
color : rgb(255,204,153) /* codage couleur en décimal */
color : #FFCC99 /* codage couleur en hexadécimal */
```

3. Compléter les codes couleurs suivants :

```
color : rgb(105,16,42) /* codage couleur en décimal */
color : #              /* même couleur codée en hexadécimal */
```

```
color : #E0EDDA            /* couleur codée en hexadécimal */
color : rgb(    ,    ,    ) /* même couleur codée en décimal */
```

> Remarques :
>
> - un site pour déterminer les codes couleurs à l'aide d'une palette : https://htmlcolorcodes.com/fr/ ,
> - pour en savoir plus sur le codage RVB : https://fr.wikipedia.org/wiki/Rouge_vert_bleu,
> - sur le système de quadrichromie et la difficultée de passer des systèmes RVB ou CMYK : https://fr.wikipedia.org/wiki/Quadrichromie.

2. Calcul binaire : addition et multiplication.

A. Addition en binaire.

Considérons l'addition en binaire suivante :

$$\begin{array}{r} 01001101 \\ +\ 01101100 \\ \hline = \end{array}$$

Que l'on peut vérifier en décimal :

$$\begin{array}{r} \\ + \\ \hline = \end{array}$$

Les opérations arithmétiques en binaire se font de la même façon qu'en décimal.

B. Multiplication en binaire.

Considérons la multiplication en binaire suivante :

$$\begin{array}{r} 1101 \\ \times\ 1001 \\ \hline = \end{array}$$

La multiplication en binaire s'effectue par de simples décalages de l'écriture.

C. retour sur le passage du binaire en hexadécimal.

Considérons le nombre binaire codé sur 8 bits : $\underbrace{01101101}_{8\ bits}$,

On eut écrire que : $\underbrace{01101101}_{bloc1\ \ bloc2} = \underbrace{01100000}_{bloc1} + \underbrace{00001101}_{bloc2} = \underbrace{0110}_{bloc1} \times \underbrace{10000}_{16\ en\ base\ 2} + \underbrace{00001101}_{bloc2}$

On a alors :

$0110 = 6\ en\ hexadécimal$

$1101 = 13\ en\ décimal = D\ en\ hexadécimal$

et donc $01101101 = 6D\ en\ hexadécimal$

Pour convertir un binaire en hexadécimal, il suffit de regrouper l'écriture par blocs de 4 bits et de traduire chacun de ces blocs en hexadécimal.

Ce que confirment ces commandes python :

```
>>> 0x6D
109
>>> 0b01101101
109
>>> bin(0x6D)
'0b1101101'
>>> hex(0b01101101)
'0x6d'
>>>
```