1. Numeriser des informations

ACTIVITÉ :

1°)

QUI ?	Genre ?	Look EMO?		Moins de 15 ans ?	Code
Pierre	garçon	Oh non !	0	oui	‘001’
Nadia	fille	Good	1	oui	‘111’
David	garçon	Enorme	1	non	‘010’
Hamdi	garçon	Certes	1	oui	‘011’
Marie	fille	Surtout pas	0	non	‘100’
Adèle	fille	Evidemment	1	non	‘110’
Ming Fu	fille	Beurk	0	oui	‘101’
Igor	garçon	Horreur	0	non	‘000’

2°) En trois questions, « Est-ce une fille ? » « Aime-t-elle le look Emo ? » « A-t-elle moins de 15 ans? », nous allons forcément deviner parmi les huit personnes celle correspondant au portrait, car toute l’information est utile ici. D’ailleurs, nous avons symbolisé ce fait en utilisant un bit pour chacune des questions binaires, reporté dans la colonne de droite.
Deux questions binaires, c’est-à-dire deux bits d’information, ne suffisent pas ici. Par exemple, si on ne sait pas si la personne recherchée a moins de 15 ans, alors Adèle et Nadia ou Igor et Pierre, par exemple, ne peuvent pas être distingués, puisqu’ils pensent la même chose du look Emo. On constate bien ici que le nombre de bits correspond au nombre de questions binaires à poser pour deviner toute l’information. Cela correspond donc à la taille en information.
3°) On voit ici que la réciproque est fausse : ce n’est pas parce qu’il y a trois bits d’information que nous pouvons distinguer huit éléments : si tous ou simplement plus ou moins de la moitié des garçons avaient eu moins de quinze ans, par exemple, les deux informations auraient été redondantes et le codage de l’information n’aurait pas été suffisant.
4°) On mesure aussi qu’il faut bien s’entendre sur le codage choisi : dans le cas du « Look Emo » les réponses contiennent bien plus d’information que oui ou non, mais des appréciations de valeur ou des nuances de ton qui peuvent, ou non, être elles-mêmes codées lors de la numérisation de l’information.
5º) Il faudrait au minimum 4 questions binaires : on code sur 4 bits : $2^4=16$

voir EXERCICE 1

2. Numériser des nombres

2.1. Coder des nombres en ….

2.1.1 « Rappel » : la numération de position en base décimale

Il y a 10 SYMBOLES ici 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
Avec ces derniers, on peut compter jusqu'à 9. Et si l'on veut aller au-delà de 9, il faut changer de rang (base DÉCIMALE)
Exemple : $(135)_{10}=100+30+5=1\times100+3\times10+5\times1=1\times10^2+3\times10^1+5\times10^0$ (on retrouve les unités, dizaines, centaines...de droite à gauche)

Cette écriture est différente de $(351)_{10}=3\times10^2+5\times10^1+1\times10^0$ bien que composée des mêmes chiffres (la POSITION est donc importante)

voir EXERCICE 2

2.1.2 La base 2

Il y a 2 SYMBOLES ici 2 chiffres : 0,1 (chiffres binaires ou *Binary Digits*, plus simplement ***bits***)

Avec ces derniers, on peut compter jusqu'à 1. Et si l'on veut aller au-delà de 1, il faut changer de rang (base **BINAIRE**)

Exemple : $(1011)_2=1\times2^3+0\times2^2+1\times2^1+1\times2^0=(64)_{10}\neq(1110)_2$ *encore la position qui n'est pas la même...*

2.1.3 La base 16

Il faut 16 SYMBOLES ici 10 chiffres 0,1,2,3,4,5,6,7,8,9 (...il n'y a pas d'autres chiffres!) et 6 lettres : A,B,C,D,E,F (qui correspondent en décimal respectivement à 10,11,12,13,14,15)

Avec ces derniers, on peut compter jusqu'à 15. Et si l'on veut aller au-delà, il faut changer de rang (base HEXADECIMALE)

CONVERSIONS DE BASE EN PYTHON

1°) Fonction Python qui réalise une conversion d'un nombre n dans une base b :

```
def convert(n, b):
    res = []
    while n > 0:
        res.append(n % b)
        n = n // b
    res.reverse()
    return res
```

2°) CORRECTION AIDE MÉMOIRE

Pour ...	*Je tape dans la console*
Pour convertir un nombre binaire en base 10	
Obtenir l'écriture décimale du nombre binaire 01001101	>>> 0b01001101 OU >>>int('01001101',2)
Pour convertir un nombre décimal en base 2	
Obtenir l'écriture binaire de 43 (écrit en décimal)	>>> bin(43)
Pour convertir un nombre hexadécimal en base 10	
Obtenir l'écriture décimale de DF40E (écrit en hexadécimal)	>>> 0xDF40E OU >>>int('DF40E',16)
Pour convertir un nombre décimal en base 16	
Obtenir l'écriture hexadécimale de 1759 (ecrit en décimal)	>>> hex(1759)
Pour convertir un nombre binaire en base 16	
Obtenir l'écriture hexadécimale de 1001011(écrit en binaire)	>>> hex(0b1001011)
Pour convertir un nombre hexadécimal en base 2	
Obtenir l'écriture binaire de A48E (écrit en hexadécimal)	>>>bin(0xA48E) OU >>>bin(int('A48E',16))

2.2 Encodage

	Calcul des possibles	**Les entiers de...à…**
UN OCTET	2^8	0 à 255
16 BITS	2^{16}	0 à 65535
32 BITS	2^{32}	0 à 2^{32}-1
64 BITS	2^{64}	0 à 2^{64}-1

2.2.1 Entier positif en binaire sur n bits

Entier	Sur 4 bits	Sur 8 bits	Sur 16 bits	Sur 32 bits
135	overflow	10000111	0000000010000111	00000000000000000000000010000111

voir **EXERCICE** 3

2.2.2 Somme et produits

Voici la table d'addition binaire :

+	0	1
0	0	1
1	1	10

$(135)_{10}+(100)_{10}=(10000111)_2+(1100100)_2$

donc
```
  10000111
+  1100100
= 11101011
```
et $(11101011)_2=(235)_{10}$

Voici la table de multiplication binaire :

+	0	1
0	0	0
1	0	1

$(6)_{10}\times(3)_{10}=(110)_2\times(11)_2$

donc
```
  00110
x 00011
→ 00110
   +0110
= 10010
```
et $(10010)_2=(18)_{10}$

voir EXERCICE 4