Info 6 - Binaire et Hexadécimal

On se fixe ici l'objectif de créer des scripts qui permettent de transformer des nombres dans les bases 2, 10 et 16.

On demande de rédiger un compte-rendu au format *.doc* ou *.odt* à transférer en fin d'activité par l'intermédiaire de l'onglet ***Mon Compte*** du site https://nsibranly.fr en utilisant le code : **tp6** . Ce compte-rendu contiendra :

- les réponses aux différentes questions posées,
- les captures d'écran **des morceaux de codes** écrits **et** celles **des résultats des exécutions**.

Attention à repérer correctement les titres de paragraphe.

1- Code qui fait une conversion base 2 → base 10 :

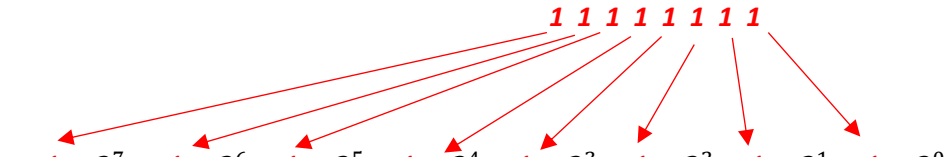
Ecrire le code d'une fonction nommée *base2Vers10()* qui prend en argument un **string** composé uniquement de caractères ''1'' ou ''0''. Ce string représente l'écriture en base 2 d'un nombre.
La fonction *base2Vers10()* doit renvoyer un **nombre entier** qui sera l'écriture en base 10 du nombre mis en argument.
Par exemple, l'exécution du programme principal ci-contre donnera dans la console :

```
# programme principal
nBase2 = "1101"
nBase10 = base2vers10(nBase2)
print(f"({nBase2})2 = ({nBase10})10")

nBase2 = "11111111"
nBase10 = base2vers10(nBase2)
print(f"({nBase2})2 = ({nBase10})10")

nBase2 = "1111111111111111"
nBase10 = base2vers10(nBase2)
print(f"({nBase2})2 = ({nBase10})10")
```

```
>>> (executing file "tpBinaire.py")
(1101)2 = (11)10
(11111111)2 = (255)10
(1111111111111111)2 = (65535)10
```

Aides :

- Pour réaliser la conversion sur le nombre binaire `"11111111"` on réalise l'opération suivante :



$$1\times2^7+1\times2^6+1\times2^5+1\times2^4+1\times2^3+1\times2^2+1\times2^1+1\times2^0$$
$$=128+64+32+16+8+4+2+1$$
$$=\mathbf{255}$$

- Avec la boucle for i in range(… , … , …) , il est possible de parcourir les caractères d'un string en partant de la droite et en allant vers la gauche.

2- Amelioration du code precedent :

⇨ Ecrire le code d'une fonction nommée *enleveEspace()* qui prend en argument un **string** et retourne le même string sans les caractères « *espace* ». Par exemple :

```
>>> enleveEspace("1111 0010")
'11110010'
```

⇨ Modifier le code de la fonction *base2Vers10()* déjà écrite, afin qu'elle puisse traiter des nombres binaires qui comprennent des espaces. L'exécution de :

```
nBase2 = "1111 1111 1111 1111 1111 1111"
nBase10 = base2vers10(nBase2)
print(f"({nBase2})2 = ({nBase10})10")
```

Devra donner :

```
(1111 1111 1111 1111 1111 1111)2 = (16777215)10
```

3- CONVERSION DES CHIFFRES DE BASE 16 :

Une fonction nommée *valeur()* prend en argument un chiffre de la base 16 et renvoie sa valeur en base 10.
Son code Python est donné ci-contre, mais mal indenté.

```
def valeur(c) :
if not c in "0123456789abcdef" :
val = None
elif c == 'f' : val = 15
elif c == 'e' : val = 14
elif c == 'd' : val = 13
elif c == 'c' : val = 12
elif c == 'b' : val = 11
elif c == 'a' : val = 10
else : val = int(c)
return val
```

⇨ Copier ce code, corriger les indentations et le tester dans la console pour : `valeur('a')`, `valeur('3')` et `valeur('t')`

4- CODE QUI FAIT UNE CONVERSION BASE 16 → BASE 10 :

Ecrire le code d'une fonction nommée *base16Vers10()* qui prend en argument un **string** qui représente l'écriture en base 16 d'un nombre.
La fonction *base16Vers10()* doit renvoyer un **nombre entier** qui sera l'écriture en base 10 du nombre mis en argument.
Par exemple, l'exécution du programme principal ci-contre donnera dans la console :

```
# programme principal
nBase16 = "ab50"
nBase10 = base16vers10(nBase16)
print(f"({nBase16})16 = ({nBase10})10")

nBase16 = "ff ff ff"
nBase10 = base16vers10(nBase16)
print(f"({nBase16})16 = ({nBase10})10")
```

```
>>> (executing file "tpBinaire.py")
(ab50)16 = (1466)10
(ff ff ff)16 = (16777215)10
```

5- CODE QUI CONVERTI UNE NOTATION DE COULEUR HEXA EN NOTATION RGB :

En informatique, une couleur est obtenue en superposant sur un pixel une lumière rouge, verte et bleue. L'intensité de chacune de ces lumières est donnée sur 1 octet. On a donc 3 octets pour définir la couleur de chaque pixel. Par exemple :

- si l'octet de la lumière rouge est à 1111 1111 en binaire, ce qui fait 255 en décimal, l'intensité de cette lumière est maximale,
- si l'octet de la lumière verte est à 0000 0000 en binaire, ce qui fait 0 en décimal, l'intensité de cette lumière est minimale,
- si l'octet de la lumière bleue est à 0111 1111 en binaire, ce qui fait 127 en décimal, l'intensité de cette lumière est à moitié de sa valeur max,

on obtient alors la couleur suivante :

La notation « *rgb* » pour définir cette couleur dans le web est : rgb(255,0,127)

Un développeur travail souvent avec une notation dite « *hexadécimale* ». Dans cet exemple, la conversion de ''FF'' d'hexa en décimal donne 255, celle de ''00'' donne 0 et celle de ''7F'' donne 127. La notation hexa de cette même couleur sera ainsi ''#FF007F'' . Par convention, pour définir une couleur en hexa, on préfixe les 6 chiffres hexa par ''#''.

⇨ Jeter un coup d'œil sur ce lien : https://htmlcolorcodes.com/fr/selecteur-de-couleur/ dans lequel on retrouve ces notations.

⇨ Ecrire le code d'une fonction nommée couleur*()* qui prend en argument un **string** qui représente l'écriture hexa d'une couleur et qui retourne un **string** qui représente son écriture rgb.

Par exemple, on peut exécuter dans la console cette fonction :

```
>>> couleur("#ff007f")
'(255,0,247)'
```