REPRÉSENTATION DES NOMBRES ENTIERS - CORRECTION

Exercice 1

1. $5279 = 5\times10^3 + 2\times10^2 + 7\times10^1 + 9\times10^0$

9 est le chiffre des unités, 7 celui des dizaines, 2 celui des centaines et 5 celui des milliers.

2. Avec 5,2,7 et 9, on a 4 possibilité pour le chiffre des unités. Une fois le chiffre des unités fixé, il nous reste 3 possibilités pour les dizaines. Une fois les deux chiffres des unité et des dizaines fixés, il nous reste 2 possibilités pour celui des centaines. Et une fois les 3 premiers fixés, nous n'avons plus le choix pour le dernier.

Ainsi, il y a $4\times3\times2\times1=24$ nombres différents possibles qui s'écrivent avec les chiffres 5,2,7 et 9.

Exemple : $2957 = 2\times10^3 + 9\times10^2 + 5\times10^1 + 7\times10^0$

Exercice 2

La phrase devrait plus précisément s'écrire « *Il y a $(10)_2$ sortes de gens au monde : ceux qui connaissent le binaire et les autres* », ce qui revient à écrire « *Il y a $(2)_{10}$ sortes de gens au monde : ceux qui connaissent le binaire et les autres* »

Exercice 3 : la base 2

1. a) $(0101)_2 = (5)_{10}$
   b) $(110)_2 = (6)_{10}$
   c) $(11101001)_2 = (233)_{10}$

2. Ci-dessous :

```
1  def Convert2_10(nb):
2      nb10=0 #variable dans laquelle on va stocker le nombre en base 10
3      for i in range(8): #on parcourt le nombre en binaire
4          if nb[i]=="1":
5              nb10+=2**(7-i)
6      return nb10
7
8  ########### Corps du programme ####################
9  nb2=input("Donner un nombre binaire codé sur un octet : ")
10 print("Le nombre binaire ", nb2 , " correspond à ",Convert2_10(nb2), " en base 10.")
```

3. a) $(14C)_{16} = (332)_{10}$
   b) $(7FA)_{16} = (2042)_{10}$
   c) $(BEEF)_{16} = (48879)_{10}$

Exercice 4 : écrire en base 2

1) $(14)_{10} = (1110)_2$: il faudra 4 bits pour coder ce nombre
2) $(218)_{10} = (11011010)_2$: il faudra 8 bits pour coder ce nombre
3) $(42)_{10} = (101010)_2$: il faudra 6 bits pour coder ce nombre
4) $(57)_{10} = (111001)_2$: il faudra 6 bits pour coder ce nombre

Exercice 5 : écrire en base 16

1) $(14)_{10} = (E)_{16}$ 2) $(218)_{10} = (DA)_{16}$ 3) $(42)_{10} = (2A)_{16}$ 4) $(57)_{10} = (39)_{16}$

Exercice 6 : bases 2 et 16

1. On se sert du tableau du cours (partie 1.2.3)
   Question 1) : $(1110)_2 = (E)_{16}$
   Question 2) : $(11011010)_2 = (1101\ 1010)_2 = (DA)_{16}$
   Question 3) : $(2A)_{16} = (0010\ 1010)_2 = (101010)_2$
   Question 4) : $(39)_{16} = (0011\ 1001)_2 = (111001)_2$
   On trouve les bons nombres.

2. 1) $(1001010)_2 = (0100\ 1010)_2 = (4A)_{16}$
   2) $(100010001)_2 = (0001\ 0001\ 0001)_2 = (111)_{16}$
   3) $(10100100111110010)_2 = (0001\ 0100\ 1001\ 1111\ 0010)_2 = (149F2)_{16}$

3. 1) $(5A92E3)_{16} = (0101\ 1010\ 1001\ 0010\ 1110\ 0011)_2$
   2) $(BAD)_{16} = (1011\ 1010\ 1101)_2$
   3) $(FACADE)_{16} = (1111\ 1010\ 1100\ 1010\ 1101\ 1110)_2$

Exercice 7 : un peu d'arithmétique...

1. a) $(1101)_2+(111)_2=(10100)_2$: à coder sur 5 bits
   b) $(1101)_2\times(111)_2=(1011011)_2$: à coder sur 7 bits
   c) $(1111)_2+(10)_2=(10001)_2$: à coder sur 5 bits

2. a)

```
1  def somme(nb1,nb2):
2      r=0
3      nbresul=""
4      for i in range(-1,-9,-1):
5          c1=int(nb1[i])
6          c2=int(nb2[i])
7          c=(c1+c2+r)%2
8          r=(c1+c2+r)//2
9          nbresul=str(c)+nbresul
10     if r==1:
11         nbresul=str(r)+nbresul
12     return nbresul
13
14 ######### CORPS DU PROGRAMME ##########
15 nb1=input("Donner un nombre en binaire sur 8 bits : ")
16 nb2=input("Donner un deuxième nombre en binaire sur 8 bits : ")
17 S=somme(nb1,nb2)
18 print("La somme en binaire de ",nb1," et ",nb2," est : ",S)
```

   b) A faire vous-même.

Exercice 8 : un peu de programmation...

1. a)

```
def conversion(n,b):
    result=''
    q=n
    while q!=0:
        result=str(q%b)+result
        q=q//b
    return result

############## CORPS DU PROGRAMME #############
nb=int(input("Choisir un nombre exprimé en base 10 : "))
base=int(input("Choisir une base : "))
nbconvert=conversion(nb,base)
print(nb," converti en base ",base, "est : ",nbconvert)
```

b) En base 16, un « 11 » sera écrit « 11 » et non « B ». Donc le programme ne fonctionne que pour les bases inférieures à 10.

2.

```
def conversion(nb):
    nb10=0
    p=len(nb)-1
    for i in nb:
        if i in '0123456789':
            nb10=nb10+int(i)*16**p
        elif i=='A':
            nb10=nb10+10*16**p
        elif i=='B':
            nb10=nb10+11*16**p
        elif i=='C':
            nb10=nb10+12*16**p
        elif i=='D':
            nb10=nb10+13*16**p
        elif i=='E':
            nb10=nb10+14*16**p
        else :
            nb10=nb10+15*16**p
        p=p-1
    return nb10

########## CORPS DU MESSAGE ###############
nb16=input("Donner un nombre hexadécimal : ")
nb10=conversion(nb16)
print("La conversion en base 10 du nombre hexadécimal ",nb16, "est : ",nb10)
```

Un peu plus fastidieux… puisqu'il faut considérer les 6 cas où le reste de la division euclidienne par 16 est 10 ou 11 ou 12 ou 13 ou 14 ou 15 ...