

REPRÉSENTATION DES NOMBRES ENTIERS - CORRECTION

Exercice 1

1. $5279 = 5 \times 10^3 + 2 \times 10^2 + 7 \times 10^1 + 9 \times 10^0$

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 possibilités 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 unités 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 $4 \times 3 \times 2 \times 1 = 24$ nombres différents possibles qui s'écrivent avec les chiffres 5,2,7 et 9.

Exemple : $2957 = 2 \times 10^3 + 9 \times 10^2 + 5 \times 10^1 + 7 \times 10^0$

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) $(11101001)_2 = (233)_{10}$

3. a) $(14C)_{16} = (332)_{10}$

b) $(7FA)_{16} = (2042)_{10}$

c) $(BEEF)_{16} = (48879)_{10}$

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.")
```

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)

```
1 def conversion(n,b):
2     result=''
3     q=n
4     while q!=0:
5         result=str(q%b)+result
6         q=q//b
7     return result
8
9 ##### CORPS DU PROGRAMME #####
10 nb=int(input("Choisir un nombre exprimé en base 10 : "))
11 base=int(input("Choisir une base : "))
12 nbconvert=conversion(nb,base)
13 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.

```
1 def conversion(nb):
2     nb10=0
3     p=len(nb)-1
4     for i in nb:
5         if i in '0123456789':
6             nb10=nb10+int(i)*16**p
7         elif i=='A':
8             nb10=nb10+10*16**p
9         elif i=='B':
10            nb10=nb10+11*16**p
11        elif i=='C':
12            nb10=nb10+12*16**p
13        elif i=='D':
14            nb10=nb10+13*16**p
15        elif i=='E':
16            nb10=nb10+14*16**p
17        else :
18            nb10=nb10+15*16**p
19        p=p-1
20    return nb10
21
22 ##### CORPS DU MESSAGE #####
23 nb16=input("Donner un nombre hexadécimal : ")
24 nb10=conversion(nb16)
25 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 suite à la division euclidienne par 16 est 10 ou 11 ou 12 ou 13 ou 14 ou 15 ...