Nombres flottants

1. Le problème

Exercice 1.

Supposons que nous disposions de 8 cases pour écrire n'importe quel nombre. Toutes les cases doivent être remplies.
Comment coderiez-vous, avec les dix chiffres usuels :

- le nombre "trois" →

- le nombre "treize" →

- le nombre "douze virgule cinq" →

- le nombre "un tiers" →

- le nombre "mille tiers" →

Exercice 2.

Supposons que nous disposions de 8 cases pour écrire n'importe quel nombre. Toutes les cases doivent être remplies.
Comment coderiez-vous, avec les dix chiffres usuels :

- le nombre "sept" $\rightarrow$
- le nombre "soixante-dix" $\rightarrow$
- le nombre "sept mille" $\rightarrow$
- le nombre "zéro virgule sept" $\rightarrow$
- le nombre "sept milliards" $\rightarrow$

Quelle règle avez-vous utilisée ?

Nous voudrions coder des nombres ”à virgule”.
Le problème, comme pour les nombres entiers, est que :

- il existe une infinité de nombres différents ;
- dans un ordinateur, la ”place” pour coder un nombre est limitée.

Il va donc falloir faire des choix et trouver une règle.

2. Écriture scientifique

L'écriture scientifique permet d'écrire des nombres de façon normalisée et compacte. Chaque nombre est écrit avec :

- une ***mantisse***, qui est est un nombre avec un seul chiffre avant la virgule ;
- un ***exposant*** qui désigne la puissance de 10 par laquelle il faut multiplier la mantisse pour obtenir le nombre voulu.

$$\text{nombre} = \text{mantisse} \times 10^{\text{exposant}}$$

Exemple 1.

$$\begin{aligned} 15 &= 1,5 \times 10^1 \\ 150 &= 1,5 \times 10^2 \\ 1,5 &= 1,5 \times 10^0 \\ 0,015 &= 1,5 \times 10^{-2} \end{aligned}$$

Exercice 3.

Écrivez en écriture scientifique :

- le nombre "trois"
- le nombre "treize"
- le nombre "douze virgule cinq"
- le nombre "un tiers"
- le nombre "mille tiers"
- le nombre "sept"
- le nombre "soixante-dix"
- le nombre "sept mille"
- le nombre "zéro virgule sept"
- le nombre "sept milliards"

Revenons au cas où nous ne disposons que de huit cases, mais utilisons l'écriture scientifique.
Choisissons par exemple d'utiliser :

1. six cases pour la mantisse ;
2. deux cases pour l'exposant :
   2.1 une case pour le signe de l'exposant
   2.2 une case pour sa valeur absolue

Comme on sait que la virgule est toujours au même endroit (après le premier chiffre de la mantisse), on n'a pas besoin de la coder.

$150 \to 1,50000 \times 10^2 \to$

1	5	0	0	0	0	+	2

Exercice 4.

Codez avec la méthode ci-dessus :

1. le nombre ”trois” →
2. le nombre ”treize” →
3. le nombre ”douze virgule cinq” →
4. le nombre ”un tiers” →
5. le nombre ”mille tiers” →
6. le nombre ”sept” →
7. le nombre ”soixante-dix” →
8. le nombre ”sept mille” →
9. le nombre ”zéro virgule sept” →
10. le nombre ”sept milliards” →

Exercice 5.

1. Codez avec la méthode ci-dessus , les nombres :
   1.1 "un tiers" →

   1.2 "deux tiers" →

2. Si on calcule à partir des codages précédents, qu'obtient-on pour l'addition de "un tiers" et "deux tiers" ?
   "un tiers plus deux tiers" →

3. Que se passerait-il si on avait non pas huit cases, mais dix ?
4. Conclusion ?

3. Du décimal au binaire

Soit x un nombre compris strictement entre 0 et 1 et noté en écriture décimale. Pour l'écrire en binaire dans un texte T, on utilise l'algorithme :

T est le texte "0,"
tant que $x \neq 0$ **faire**
    x prend la valeur de $2 \times x$
    si $x \geq 1$ **alors**
        écrire 1 à droite de T
        x prend la valeur $x - 1$
    sinon
        écrire 0 à droite de T
    fin
fin

Exercice 6.

Écrire $0,25$; $0,125$; $0,5$; $0,1$; $0,2$ en binaire.

Pour écrire un nombre positif en binaire, on le décompose entre sa partie entière (celle qui est à gauche de la virgule) et sa partie fractionnaire (celle qui est à droite de la virgule).
Par exemple, pour écrire $6,5$:

- la partie entière 6 est écrite comme d'habitude : 110 ;
- la partie fractionnaire est $0,5$ qui s'écrit en binaire $0,1$;
- on additionne les deux parties : $6,5 = (110,1)_2$

On peut remarquer dans l'autre sens que

$$\begin{array}{rcccccccc} (110,1)_2 & = & 1\times 2^2 & + & 1\times 2^1 & + & 0\times 2^0 & + & 1\times 2^{-1} \\ & = & 4 & + & 2 & + & 0 & + & 0,5 \\ & = & 6,5 & & & & & & \end{array}$$

4. Écriture scientifique en binaire

On peut utiliser l'écriture scientifique en binaire en décomposant un nombre sous la forme :

$$\text{mantisse} \times 2^{\text{exposant}}$$

Par exemple :

$$6,5 = (110,1)_2 = (1,101)_2 \times 2^{(10)_2}$$

Comme le premier chiffre est toujours un 1, on ne l'écrit pas. On convient que le signe de l'exposant est désigné par 1 s'il est négatif et 0 s'il est positif. Si on a huit cases avec 5 pour la mantisse, 1 pour le signe de l'exposant et 2 pour la valeur absolue de l'exposant, cela donne :

$6,5 \to 1,101 + 10 \to$

1	0	1	0	0	0	1	0
$\cancel{1},10100$					+	2	

Exercice 7.

Codez avec la méthode précédente :

1. le nombre ”trois” →
2. le nombre ”sept” →
3. le nombre ”treize” →
4. le nombre ”six virgule cinq” →
5. le nombre ”un tiers” →
6. le nombre ”un quart” →
7. le nombre ”douze virgule cinq” →
8. le nombre ”zéro virgule un” →

Exercice 8.

Décodez, avec la méthode précédente :

1.

1	1	0	0	0	0	1	0

2.

1	1	0	0	0	0	1	1

3.

1	0	0	0	1	1	1	0

4.

0	1	0	0	0	1	0	1

Exercice 9.

Exécutez, dans une console Python, le code suivant :

```
0.1 + 0.2 == 0.3
```

Expliquez ce résultat. Quelle conclusion en tirer ?

Exercice 10.

Combien de nombres différents peut-on coder sur 64 bits ?
Est-ce suffisant pour coder tous les nombres ?

5. Compléments

Les ordinateurs modernes codent en général les nombres avec la norme IEEE-754. Cela consiste à écrire les nombres avec une méthode semblable à celle que nous avons vue dans la section précédente, mais sur 64 bits. La norme précise aussi des algorithmes pour calculer de façon ”pas trop fausse”.
Ces méthodes où la virgule est indiquée par un exposant sont appelées ”à ***virgule flottante***” et les nombres sont des ***nombres à virgule flottante*** ou simplement ***flottants***.
En Python, nous utiliserons deux types de nombres :

- les entiers, de type **`int`** ;
- les flottants, de type **`float`**.

De même que la fonction **`int`**, il existe une fonction **`float`** qui tente de renvoyer, à partir d'une chaîne de caractères, le flottant qui lui correspond.