Représenter les nombres réels

1. En base 2

- En base 10, le nombre $652,375$ se décompose comme suit :
  $6 \times 100 + 5 \times 10 + 2 \times 1 + 3 \times 0,1 + 7 \times 0,01 + 5 \times 0,001$ c'est à dire
  $6 \times 10^2 + 5 \times 10^1 + 2 \times 10^0 + 3 \times 10^{-1} + 7 \times 10^{-2} + 5 \times 10^{-3}$
- En base 2, c'est le même principe : `110,101` est la représentation en base 2 du nombre
  $1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}$, c'est à dire
  $1 \times 4 + 1 \times 2 + 0 \times 1 + 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 1 \times \frac{1}{8}$, ce qui donne $4 + 2 + 0,5 + 0,125 = 6,625$

Méthode : de la base 2 vers la base 10

- On a vu que pour retrouver l'écriture en base 10 d'un nombre entier, on le décompose à l'aide des puissances de 2 (1,2,4,8,...)
- Cette méthode se prolonge à la partie décimale d'un nombre en utilisant les puissances négatives de 2, c'est à dire les inverses des puissances de 2 ($\frac{1}{2}$,$\frac{1}{4}$,$\frac{1}{8}$,...)

Exemple : conversion de `101,011` en base 10

`101,011` s'écrit
$1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3} = 4 + 1 + 0,25 + 0,125 = 5,375$

Exercice 1 :

Convertir les nombres suivants en base 10.

1. `1,001`
2. `0,1001`
3. `101,101`

Réponses :

1.
2.
3.

Méthode : De la base 10 vers la base 2

Le principe est celui déjà vu dans un précédent chapitre pour la partie entière du nombre à coder, et se prolonge pour sa partie décimale.

Pour la partie entière :

- On écrit la partie entière en base 10.
- On le divise par 2 et on note le reste de la division euclidienne (soit `1` soit `0`)
- On refait la même chose avec le quotient précédent, et on met de nouveau le reste de côté.
- On réitère la division, jusqu'à ce que le quotient soit `0` .
- La partie entière du nombre en base 2 se lit à l'aide des restes en commençant par celui de la dernière opération effectuée.

Pour la partie décimale :

- On écrit la partie décimale en base 10.
- On le multiplie par 2 et on note le chiffre à gauche de la virgule (`1` ou `0`)
- On recommence avec la partie décimale obtenue en la multipliant par 2.
- On continue jusqu'a ce que la partie décimale soit égale à zéro.

Exemple : Conversion de 6,375 en base 2

- Partie entière :
  - $6 = 2 \times 3 +$ `0`
  - $3 = 2 \times 1 +$ `1`
  - $1 = 2 \times 0 +$ `1`
  - 6 s'écrit `110` en base 2

- Partie décimale :
  - $0,375 \times 2 =$ `0` $,75$
  - $0,75 \times 2 =$ `1` $,5$
  - $0,5 \times 2 =$ `1` $,0$
  - $0,625$ s'écrit `0,011` en base 2

- Conclusion : Le nombre $6,375$ s'écrit `110,011` en base 2

Exercice 2 :

1. Convertir $0,25$ en base 2
2. Convertir $12,75$ en base 2
3. Convertir $0,1$ en base 2

Réponses :

1.
2.
    -
    -
    -
    -
    -
    -
    -

1.
    -
    -
    -
    -
    -
    -
    -
    -
    -
    -
    -
    -
    -

In [3]:

```
# Exécuter le code ci-dessous
print(0.1+0.2)
print(0.1+0.2==0.3)
```

```
0.30000000000000004
False
```

A retenir :

- Un nombre à développement décimal fini en base 10 ne l'est pas forcément en base 2(exemples : 0.1, 0.3...)
- Cela peut engendrer des comportements étonnants(lors de tests d'égalités) et avoir de lourdes conséquences(voir exercice 4)

Exercice 3 :

Convertir $\frac{1}{3}$ en base 2

Réponse :

-
-
-
-
-
-

Exercice 4 : Missile Patriot

Le 25 février 1991, pendant la Guerre du Golfe, une batterie américaine de missiles Patriot, à Dharan (Arabie Saoudite), a tenté d'intercepter un missile Scud irakien :

- Les nombres dans le système du Patriot étaient arrondis à 23 chiffres binaires après la virgule.
- Le temps était compté par l'horloge interne du système en dixième de seconde.
- Au moment de l'attaque, la batterie de missile Patriot était allumée depuis environ 100 heures.

Questions :

1. Donner la décomposition en base 2 d'un dixième avec 23 chiffres après la virgule.
2. Quels sont les 6 chiffres suivants de cette décomposition?
3. Montrer que l'erreur d'arrondi correspond alors à environ $0,000000095$ secondes.
4. Quelle était l'erreur d'arrondi en secondes au moment de l'attaque ?
5. La vitesse d'un scud était d'environ 1676 m/s. Quelle distance parcourt-il pendant ce temps?
6. A votre avis , que s'est-il passé ?

Réponses :

1.
2.
3.
4.
5.
6.

2. Virgule fixe, virgule flottante

Pour représenter les nombres à virgule, il existe deux principales méthodes: la virgule fixe et la virgule flottante.

La virgule fixe

On fixe le nombre de chiffres avant et après la virgule.

Exercice 5 :

Si l'on utilise 4 chiffres avant la virgule et 4 chiffres après la virgule(en base 10), quels sont les nombres que l'on peut obtenir ?

Réponse:

Exercice 6

1. On suppose que l'on veut coder des nombres à virgule sur 3 bits : le premier à gauche réservé à la partie entière et les 2 autres pour la partie décimale. Lister les nombres que l'on peut alors représenter.
2. Avec 2 bits pour la partie décimale, quelle est la précision que l'on obtient ?
3. Avec 3 bits ? 4 bits ? n bits ?

Réponses:

1.
   - `000` :
   - `001` :
   - `010` :
   - `011` :
   - `100` :
   - `101` :
   - `110` :
   - `111` :

1.
   - Avec 2 bits :
   - Avec 3 bits :
   - Avec 4 bits :
   - Avec n bits :

La virgule flottante (floating point)

On utilise une notation similaire à la notation scientifique avec un exposant qui fait varier l'ordre de grandeur (c'est lui qui fait "flotter" la virgule)

Exercice 7 :

On suppose que l'on code sur 4 bits. On réserve les 2 bits à gauche pour la précision et les deux autres pour l'exposant(en complément à 2, ce qui permet d'obtenir des exposants négatifs). Ainsi `1010` représente $2 \times 10^{-2} = 0,02$.

1. Lister les nombres que l'on peut représenter.
2. Quelle est la différence avec précédemment ?

Réponses :

1.
   - `0000` :
   - `0001` :
   - `0010` :
   - `0011` :
   - `0100` :
   - `0101` :
   - `0110` :
   - `0111` :
   - `1000` :
   - `1001` :
   - `1010` :
   - `1011` :
   - `1100` :
   - `1101` :
   - `1110` :
   - `1111` :

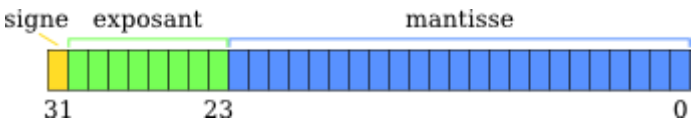
1.

3. La norme IEEE-754

Bien que sa précision soit limitée, la plupart des systèmes utilisent le système de la virgule flottante qui suffit dans la grande majorité des calculs. Depuis 1985, la norme IEEE-754 spécifie deux formats de nombres en virgule flottante en base 2. Ce deux formats sont sur 32 bits (« simple précision ») et 64 bits (« double précision »).

Sur 32 bits :

Le premier bit est le signe (+ ou -), les 8 bits suivants codent la valeur de l'exposant, les 23 bits restants servent à coder la mantisse (la partie après la virgule)



En simple précision, le plus petit nombre strictement positif représenté est $1,4 \times 10^{-45}$,le plus grand nombre(avant l'infini) est $3,40282346 \times 10^{38}$

Sur 64 bits :

Le premier bits code le signe, les 11 suivants l'exposant, les 52 autres la mantisse.

En double précision, le plus petit nombre strictement positif représenté est $2,2250738585072014 \times 10^{-308}$,le plus grand nombre(avant l'infini) est $1,7976931348623157 \times 10^{308}$

En savoir plus :

Aucune connaissance précise de la norme IEEE-754 n'est pas au programme de N.S.I mais pour en savoir plus :

- https://www.youtube.com/watch?v=mtizhxkB-Zw (https://www.youtube.com/watch?v=mtizhxkB-Zw)
- https://fr.wikipedia.org/wiki/IEEE_754 (https://fr.wikipedia.org/wiki/IEEE_754)

FIN