# Chapitre 3

# Représentation approximative des nombres réels

1ère NSI

Thème 1 - Représentation des données : types et valeurs de base

Germain Becker, Sébastien Point, Lycée E. Mounier, ANGERS

# Introduction

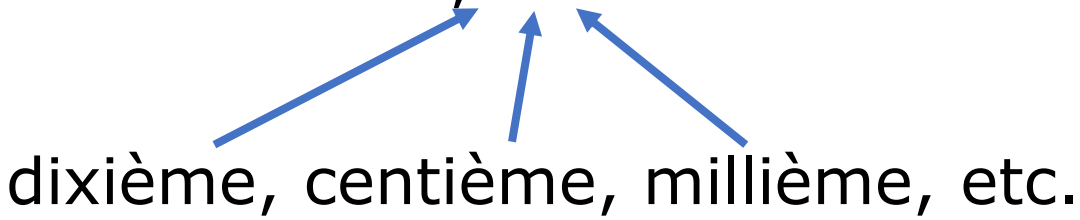
- Les nombres réels : $\mathbb{R}$ = ensemble de tous les nombres que vous connaissez
- On s'intéresse au codage des nombres décimaux = nombres à virgule (car entiers déjà traités)
- En machine : ce sont les **flottants** qui représentent les nombres réels (ex: 1.4 ; 2.5 ; -127.956 ; 5.0 ; etc.) → type `float` en Python
- Ces flottants sont représentés en binaire en machine
- Combien y a-t-il de nombres réels ? → une infinité

# Introduction

- Les nombres réels : $\mathbb{R}$ = ensemble de tous les nombres que vous connaissez
- On s'intéresse au codage des nombres décimaux = nombres à virgule (car entiers déjà traités)
- En machine : ce sont les **flottants** qui représentent les nombres réels (ex: 1.4 ; 2.5 ; -127.956 ; 5.0 ; etc.) → type `float` en Python
- Ces flottants sont représentés en binaire en machine
- Combien y a-t-il de nombres réels ? → une infinité
- Problème car dans un ordinateur on ne peut pas représenter un nombre infini de valeurs : on est toujours limité en nombres de bits, même plusieurs Téraoctets (To) = nombre fini d'octets/bits

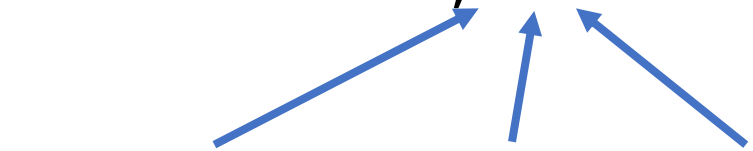
# Ecriture binaire d’un nombre réel

- En notation décimale : 124,3575



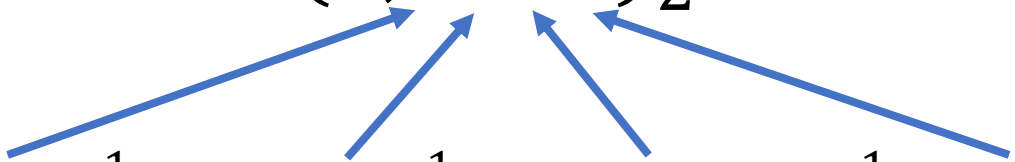
# Ecriture binaire d'un nombre réel

- En notation décimale : 124,3575



dixième, centième, millième, etc.

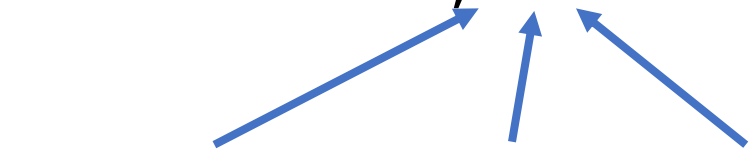
- En binaire : $(1{,}1011)_2$



demi ($\frac{1}{2}$), quart ($\frac{1}{4}$), huitième ($\frac{1}{8}$), seizième ($\frac{1}{16}$), etc.

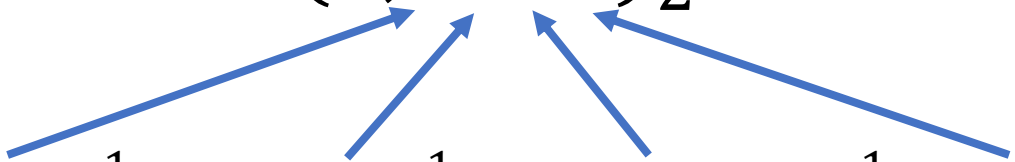
# Ecriture binaire d'un nombre réel

- En notation décimale : 124,3575



dixième, centième, millième, etc.

- En binaire : $(1{,}1011)_2$



demi ($\frac{1}{2}$), quart ($\frac{1}{4}$), huitième ($\frac{1}{8}$), seizième ($\frac{1}{16}$), etc.

- $(1{,}1011)_2 = 1 + \frac{1}{2} + \frac{1}{8} + \frac{1}{16} = 1{,}6875$

# Ecriture binaire d'un nombre réel

- **Conversion de la base 2 vers la base 10**
  - Quel nombre décimal correspond à l'écriture binaire 101,1101 ?

| Puissances de 2 | $2^2$ | $2^1$ | $2^0$ | | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
|---|---|---|---|---|---|---|---|---|
| Ecriture binaire | 1 | 0 | 1 | , | 1 | 1 | 0 | 1 |

$$(101{,}1101)_2 = 2^2 + 2^0 + 2^{-1} + 2^{-2} + 2^{-4} = 4 + 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{16} = \boxed{5{,}8125}$$

- **Exercice :** Quel est le nombre décimal correspondant à l'écriture binaire 111101,001101 ?

# Ecriture binaire d’un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l’écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu’à obtenir 1 :

# Ecriture binaire d’un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l’écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu’à obtenir 1 :

    $0{,}8125 \times 2 = \mathbf{1}{,}625$

# Ecriture binaire d’un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l’écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu’à obtenir 1 :

    $0{,}8125 \times 2 = \mathbf{1}{,}625$
    $0{,}625 \times 2 = \mathbf{1}{,}25$

# Ecriture binaire d'un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l'écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu'à obtenir 1 :

    $0{,}8125 \times 2 = \mathbf{1}{,}625$
    $0{,}625 \times 2 = \mathbf{1}{,}25$
    $0{,}25 \times 2 = \mathbf{0}{,}5$

# Ecriture binaire d'un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l'écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu'à obtenir 1 :

    $$0{,}8125 \times 2 = \mathbf{1}{,}625$$
    $$0{,}625 \times 2 = \mathbf{1}{,}25$$
    $$0{,}25 \times 2 = \mathbf{0}{,}5$$
    $$0{,}5 \times 2 = \mathbf{1}$$

# Ecriture binaire d'un nombre réel

- **Conversion de la base 10 vers la base 2**
  - Quelle est l'écriture binaire du nombre décimal 5,8125 ?

    On convertit la partie entière (à gauche de la virgule) : $5 = (\mathbf{101})_2$

    On fait des multiplications successives par 2 sans reporter la partie entière, jusqu'à obtenir 1 :

    $0{,}8125 \times 2 = \mathbf{1}{,}625$
    $0{,}625 \times 2 = \mathbf{1}{,}25$
    $0{,}25 \times 2 = \mathbf{0}{,}5$
    $0{,}5 \times 2 = \mathbf{1}$

    Les parties entières (de haut en bas) donnent les chiffres après la virgule

    Réponse : $5{,}8125 = (\mathbf{101}{,}\mathbf{1101})_2$

# Ecriture binaire d’un nombre réel

- **Exercice :**
  1) Quelle est l’écriture binaire du nombre décimal 7,78125 ?
  2) Quelle est l’écriture binaire du nombre décimal 1,2 ?

# Ecriture binaire d’un nombre réel

- **Ecritures infinies**
  - $1{,}2 = (1{,}001100110011\ldots)_2$, le cycle « 0011 » se répétant à l’infini.
  - Impossible de représenter 1,2 en machine de manière exacte !

# Représentation des nombres réels

# Représentation des nombres réels

- Codés en virgule flottante = nombre flottants
- Analogie avec l'écriture scientifique :
  - Ecriture scientifique (rappels)

$$\begin{aligned} 941{,}58 &= +9{,}4158 \times 10^{2} \\ -0{,}00145 &= -1{,}45 \times 10^{-3} \end{aligned}$$

  - Dans l'écriture :
    - Un *signe* (+ ou −)
    - Un nombre décimal, appelé *mantisse*, compris dans $[1;10[$
    - Un entier relatif $n$, appelé *exposant*

  - Forme générale de l'écriture scientifique :

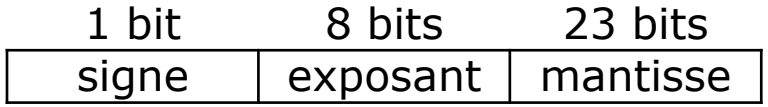
$$\pm m \times 10^{n}$$

# Représentation des nombres réels

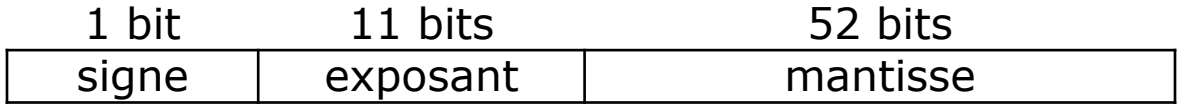
- **La norme IEEE 754**
  - Pour représenter les nombres flottants sur 32 bits (format *simple précision*) ou sur 64 bits (format *double précision*)
  - Les nombres flottants sont représentés sous la forme :

$$s.m \times 2^n$$

    - $s$ : **signe** codé sur 1 bit (0 pour + et 1 pour −)
    - $n$ : **exposant** en puissance de 2, codé sur 8 bits (simple précision) ou 11 bits (double précision)
    - $m$ : **mantisse** codée sur 23 bits (simple p.) ou 52 bits (double p.)
  - Simple précision sur 32 bits :




  - Double précision sur 64 bits :



# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - Objectif :

| signe | exposant | mantisse |
|---|---|---|
| **?** | **?** | **?** |

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - Objectif :

| signe | exposant | mantisse |
|---|---|---|
| **?** | **?** | **?** |

  - **Codage du signe** : 5,8125 est positif donc le bit de signe vaut **0** :

| signe | exposant | mantisse |
|---|---|---|
| **0** | **?** | **?** |

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - Objectif :

| signe | exposant | mantisse |
|---|---|---|
| **?** | **?** | **?** |

  - **Codage du signe** : 5,8125 est positif donc le bit de signe vaut **0** :

| signe | exposant | mantisse |
|---|---|---|
| **0** | **?** | **?** |

  - Ecriture sous la forme $s.m \times 2^n$ :

$$5{,}8125 = (+101{,}1101)_2 = (+1{,}011101 \times 2^2)_2$$

donc : *mantisse* = 1,011101 et *exposant* = 2

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - Objectif :

| signe | exposant | mantisse |
|---|---|---|
| **?** | **?** | **?** |

  - **Codage du signe** : 5,8125 est positif donc le bit de signe vaut **0** :

| signe | exposant | mantisse |
|---|---|---|
| **0** | **?** | **?** |

  - Ecriture sous la forme $s.m \times 2^n$ :

$$5{,}8125 = (+101{,}1101)_2 = (+1{,}011101 \times 2^2)_2$$

donc : *mantisse* = 1,011101 et *exposant* = 2

  - Codages de la mantisse et de l'exposant ?

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - **Codage de la mantisse** $(s.\boldsymbol{m} \times 2^n)$
    - La mantisse est toujours de la forme : $1,xxxx$ …
    - Le « 1 » à gauche de la virgule n'est pas codé
    - Seuls les chiffres à droite de la virgule sont codés sur 23 bits ou 52 bits selon le format
    - $5{,}8125 = (+101{,}1101)_2 = (+\mathbf{1,011101} \times 2^2)_2$

      donc mantisse = $\mathbf{1,011101}$
    - On code juste la partie décimale (à droite de la virgule) : **011101**
    - On complète avec des zéros pour atteindre 23 ou 52 bits :

| signe | exposant | mantisse |
|---|---|---|
| 0 | ? | **01110100…0** |
| | | (23 ou 52 bits) |

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - **Codage de l'exposant** $(s.m \times 2^{n})$
    - L'exposant $n$ est un entier relatif codé sur 8 bits ou 11 bits (selon format)
    - **Attention** : la norme IEEE 754 n'utilise pas l'encodage en complément à 2 des entiers relatifs
    - Elle prévoit un **décalage** qui dépend de l'encodage utilisé : 127 pour le format simple précision et 1023 pour le format double précision
    - Pourquoi 127 dans le format simple précision ? *(idem pour 1023 en double précision)*
      → obtenir un entier positif pour coder l'exposant

      Codage sur 8 bits : $2^8 = 256$ entiers possibles
      Les exposants signés pouvant être codés : $-127$ à $128$ ($\neq$ **complément à 2**)
      En ajoutant 127 à un exposant, on obtient un entier positif

| Exposants signés | −127 | −126 | … | 0 | … | 127 | 128 |
|---|---|---|---|---|---|---|---|
| Entier $n$ à coder | 0 | 1 | | 127 | | 254 | 255 |

+ 127

Exemple : si l'exposant est $4$, son codage est l'écriture binaire de 131 (4 + 127)

# Représentation des nombres réels

- **Exemple : représentation en machine du réel 5,8125**
  - **Codage de l'exposant** $(s.m \times 2^{n})$
    - $5{,}8125 = (+101{,}1101)_2 = \left(+1{,}011101 \times 2^{\mathbf{2}}\right)_2$
    - Exposant = $2$
    - On ajoute 127 : $2 + 127 = 129$
    - On code l'entier (positif) 129 en binaire : $(\mathbf{10000001})_2$.
    - On ajoute des zéros (à gauche) pour arriver à 8 bits si nécessaire : ici pas besoin !

    Format simple précision (sur 32 bits) :

| signe | exposant | mantisse |
|---|---|---|
| 0 | **10000001** | 01110100…0 |
| (1 bit) | (8 bits) | (23 bits) |

    En format double précision, on ajouterait 1023 à 2 pour obtenir 1025 dont l'écriture binaire est $(\mathbf{10000000001})_2$ et on obtiendrait le mot binaire suivant :

| signe | exposant | mantisse |
|---|---|---|
| 0 | **10000000001** | 01110100…0 |
| (1 bit) | (11 bits) | (52 bits) |

# Représentation des nombres réels

- **Exercice :**
  1) Quel mot binaire, au format simple précision (32 bits), représente en machine le nombre 7,78125 ?
  (rappel : $7{,}78125 = (111{,}11001)_2$)

  2) Quel est le nombre décimal représenté par le mot de 32 bits suivant ?

$$\overbrace{1}^{\text{signe}}\ \overbrace{10000110}^{\text{exposant}}\ \overbrace{10101101100000000000000}^{\text{mantisse}}$$

# La représentation est **approximative**

- **Codage approchée de certains réels !**
  - On a vu que $1{,}2 = (1{,}001100110011\ldots)_2 = +1{,}001100110011\ldots \times 2^0$
  - Sa mantisse $m = 1{,}001100110011\ldots$ est infinie
  - Il n'y a que 23 ou 52 bits réservés pour la mantisse
  - Elle est tronquée : on ne garde que les 23 (ou 52) premiers bits après la virgule :

$$\underbrace{00110011001100110011001}_{23\text{ bits}}\ \cancel{100110011\ldots}$$

  - Bilan : **le codage de 1,2 n'est qu'une valeur approchée de 1,2 !**
    Autrement dit : **le nombre flottant 1.2 n'est pas égal au réel 1,2 !**

# La représentation est **approximative**

- **Codage approchée de certains réels !**
  - On a vu que $1{,}2 = (1{,}001100110011\ldots)_2 = +1{,}001100110011\ldots \times 2^0$
  - Sa mantisse $m = 1{,}001100110011\ldots$ est infinie
  - Il n'y a que 23 ou 52 bits réservés pour la mantisse
  - Elle est tronquée : on ne garde que les 23 (ou 52) premiers bits après la virgule :

$$\underbrace{00110011001100110011001}_{23\text{ bits}}\ \cancel{100110011\ldots}$$

  - Bilan : **le codage de 1,2 n'est qu'une valeur approchée de 1,2 !**
    Autrement dit : **le nombre flottant 1.2 n'est pas égal au réel 1,2 !**
  - Conséquences : choses « étranges » dans un langage de programmation

```
Entrée [1]:  1  1.2 - 1.0
Out[1]:  0.19999999999999996
```

```
Entrée [5]:  1  0.7 - 0.3
Out[5]:  0.39999999999999997
```

# La représentation est **approximative**

- **Impossible de coder tous les nombres réels !**

| Nombre flottant | Représentation format double précision (mantisse sur 52 bits) |
|---|---|
| 1.5 | $+1.1000000000000000000000000000000000000000000000000000 \times 2^0$ |
| 1.5000000000000002 | $+1.1000000000000000000000000000000000000000000000000001 \times 2^0$ |

  - Il n'y pas de place pour coder un flottant entre 1.5 et 1.5000000000000002
  - Donc 1.5000000000000001 est codé comme 1.5
    → ces deux flottants sont égaux en machine !

# La représentation est **approximative**

- **Impossible de coder tous les nombres réels !**

| Nombre flottant | Représentation format double précision (mantisse sur 52 bits) |
|---|---|
| 1.5 | $+1.1000000000000000000000000000000000000000000000000000 \times 2^0$ |
| 1.5000000000000002 | $+1.1000000000000000000000000000000000000000000000000001 \times 2^0$ |

  - Il n'y pas de place pour coder un flottant entre 1.5 et 1.5000000000000002
  - Donc 1.5000000000000001 est codé comme 1.5
    → ces deux flottants sont égaux en machine !
  - Conséquence : ne **jamais comparer deux flottants** en programmation !

```
Entrée [6]: 1 1.5 == 1.5000000000000001
Out[6]: True
```

```
Entrée [5]: 1 0.1 + 0.2 == 0.3
Out[5]: False
```

# Conclusion

- Chaque nombre réel possède une écriture binaire
- En machine, les nombres réels sont représentés par les nombres flottants
- La norme IEEE 754 précise la façon dont sont codés les nombres flottants : sur 32 ou 64 bits.

$$s.m \times 2^n$$

- Cette représentation des nombres réels est **approximative**
- On ne peut pas représenter de manière exacte en machine tous les nombres réels
- Il faut prendre des précautions lorsque l'on manipule des flottants dans un langage informatique, en particulier ne jamais tester l'égalité entre deux flottants