TD 20 : Représentation d'un nombre réel

Exercice 1 (Représentation d'un nombre réel en informatique)

Un nombre réel

$$\pm \texttt{1.xxxxx} \times 2^n \tag{1}$$

est représenté en mémoire (selon la norme IEEE754) sur 64 bits avec dans l'ordre 1 bit de signe (`0` $\rightarrow$ positif), 11 bits pour l'exposant ($n + 1023$) et 52 bits pour la partie décimale `xxxxx` de la mantisse (ne pas oublier le 1 devant !).

$$\texttt{0 / 10000000011 / 0011001100110011001100110011001100110011001100110011} \tag{2}$$

Vous avez appris à convertir un nombre entier en base 2, on continue de faire de même : on développe la représentation binaire sur les puissances positives de 2, puis sur les puissances négatives de 2 pour la partie décimale à représenter en base 2. Une manière de faire est la suivante :

Convertir un réel $x < 1$ en base b
- On multiplie x par b
- On lit le chiffre des unités de x (c'est sa partie entière)
- On retire à x sa partie entière et on recommence

1. Montrer que 19.2 se convertit en 10011.00110011... Pour trouver sa représentation en mémoire, il faudrait encore l'écrire sous la forme (1). Que remarquez-vous sur la partie venant après la virgule ?
2. Appliquer la méthode à 0.1, puis 0.3. Ces valeurs sont-elles représentées de manière exacte en base 2 ?
3. Dans l'interprêteur python, tester `1+1+1 == 3`, puis `0.1+0.1+0.1 == 0.3`. Expliquer.
4. Est-il judicieux de faire des tests sur des flottants ? Par quoi est-il préférable de remplacer ces tests ?
5. L'associativité des réels est-elle préservée en informatique ?
6. Montrer que la représentation en mémoire de 19.2 est donnée par (2).