TP: représentation des flottants
Informatique pour tous

Dans ce TP, tous les flottants seront compris entre 0 et 1 (donc sont de la forme 0, ...).
On rappelle que, par définition, $0, x_1x_2...{}_b = x_1 \times b^{-1} + x_2 \times b^{-2} + ...$

1. Écrire une fonction `to_base10f` ayant comme arguments une base b et une liste de la forme $[x_1, x_2, ...]$, et renvoyant le flottant $x_1 \times b^{-1} + x_2 \times b^{-2} + ...$
2. Convertissez $< 0,011 >_2$ à la main puis vérifier avec `to_base10f`.
3. Modifier `to_base10f` pour éviter le calcul de puissance. Quelle est sa complexité?

Pour convertir un flottant x de la base 10 vers une base b, c'est à dire l'écrire sous la forme $x = 0, x_1x_2...{}_b$, on utilise le fait que $x \times b = x_1, x_2...{}_b$ donc $x_1 = \lfloor x \times b \rfloor$ et $x \times b - \lfloor x \times b \rfloor = 0, x_2x_3...{}_b$.

4. Écrire une fonction `from_base10f` ayant un flottant x et une base b en arguments et renvoyant la liste des chiffres après la virgule de x en base b.
5. Convertir 0.59375 en base 2 à la main puis vérifier avec `from_base10f`.
6. Convertir 0.1 en base 2 à la main. Que donne/devrait donner `from_base10f(0.1, 2)`?
7. Rajouter un argument à `from_base10f` de façon à ce que `from_base10f(x, b, p)` renvoie les p premiers chiffres après la virgule de x en base b. Tester avec $x = 0.1$
8. Calculer `0.1 + 0.1 + 0.1` avec Python. Expliquer pourquoi le résultat est faux.
9. Calculer `2.0**53 + 1.0 - 2.0**53` avec Python. Comparer avec `2.0**53 - 2.0**53 + 1.0` et `2**53 + 1 - 2**53`. Expliquer les différentes valeurs obtenues.
10. Comparer le calcul de `2.0**1024` avec `2**1024`