

TP5 - Représentation Binaire

Exercice 1 – Puissance de 2

1. Donner les résultats dans python des calculs suivants:
 - `5 & 3`
 - `4 & 3`
 - `7 & 3`
 - `8 & 7`
 - `0 & 5`
2. Inférer des tests ci-dessus le comportement de `&`. Peut être écrire les nombres ci-dessus et le résultat des calculs binaires simplifiera les choses.
3. Implémenter, dans le fichier `test_power_2.py`, deux fonctions suivantes :
 - `is_power_of_two_naive(n: int) → bool`
Cette fonction doit vérifier si l'entier `n` est une puissance de 2 en utilisant une méthode dite naïve, dans le sens qu'elle ne profite pas de sa représentation binaire. Pour ça on pourra par exemple faire des divisions successives sur `n`.
 - `is_power_of_two_bit(n: int) → bool`
Cette fonction doit déterminer si `n` est une puissance de 2 en utilisant l'opérateur binaire `&`. On pourra par exemple comparer `n` avec un autre nombre bien choisi.
4. Expliquer le reste du code et donnez une interprétation du résultat.

Exercice 2 – Comparaison de signe

1. Donner l'expression en nombre flottant de 1.33 et `-12.5`
2. Compléter, dans le fichier `test_sign.py`, les fonctions suivantes :
 - `is_positive_naive(x: float) → bool`
Cette fonction doit déterminer si `x` est positif en utilisant des comparateurs.
 - `is_positive_bit(x: float) → bool`
Cette fonction doit utiliser la représentation binaire du nombre flottant `x` pour déterminer son signe. Pour ce faire, vous pouvez exploiter la fonction `get_nth_bit(x, n)` qui retourne le `n`-ième bit.
3. Expliquer le reste du code et donnez une interprétation du résultat.