TD : Représentation des données — types et valeurs de base

Exercices 1 à 5 : à traiter sur le cahier, sans utiliser l'ordinateur.

Exercice 1 : Entiers en binaire.

1. Combien de bits sont nécessaires, au minimum, pour coder le nombre 267_{10} ?
2. Donner l'écriture binaire des nombres 245_{10} et 1097_{10}.
3. On utilise 5 bits pour coder des entiers. Comment sont codés les nombres 9 et 28 ?

Exercice 2 : Calculs en binaire.

1. Donner l'écriture décimale des nombres binaires 100101_2 et 11101001_2.
2. Poser et calculer en binaire : a. $11001101_2 + 1111001_2$ b. $11101_2 \times 101_2$ c. $101111_2 \times 1101_2$
3. On travaille avec des entiers codés sur 8 bits. Poser l'addition $10010101_2 + 10001110_2$. Que constate-t-on ? Vérifier en écrivant la somme des deux nombres en décimal.

Exercice 3 : Hexadécimal.

1. Convertir en hexadécimal : 11101010_2 et 1100101010_2.
2. Convertir $1E3A_{16}$ en base 2.
3. Convertir 1285_{10} en hexadécimal (divisions successives par 16).
4. En hexadécimal avec 4 symboles, combien d'entiers en base 10 peut-on représenter ?

Exercice 4 : Complément à 2.

1. Sur 8 bits, donner la représentation en complément à 2 de -42_{10}.
2. Sur 10 bits, donner la représentation en complément à 2 de -235_{10} et de -989_{10}.
3. Sur 8 bits, quelle valeur décimale est représentée, en complément à 2, par 10110101_2 ?
4. Quelle plage d'entiers relatifs peut-on représenter en complément à 2 sur n bits ? Donner cette plage pour $n = 8$.

Exercice 5 : Nombres flottants.

1. Écrire $0{,}375_{10}$ en binaire, par multiplications successives par 2 de la partie décimale.
2. Faire de même avec $0{,}7_{10}$. Que remarque-t-on ? Que peut-on en conclure sur son codage en machine ?

Exercices 6 et 7 : à coder en Python (sur ordinateur).

Exercice 6 : Conversions en Python (sans utiliser `int`, `bin` ou `hex`).

1. Écrire une fonction `decimal_en_binaire(n)` qui prend en paramètre un entier naturel et **renvoie** son écriture binaire sous la forme d'un `str`.
   On utilisera la méthode des divisions successives par 2.
2. Écrire une fonction `decimal_en_binaire_n(n, nb_bits)` qui prend en paramètres un entier naturel et un nombre de bits, et **renvoie** le codage de cet entier en binaire sur `nb_bits` bits sous forme de `str`.
   On commencera par vérifier que le nombre peut bien être codé sur `nb_bits` bits. Si ce n'est pas le cas, la fonction **affichera** un message d'erreur du type `"encodage impossible sur les n bits demandés"` et ne renverra rien (`None`).
3. Écrire une fonction `binaire_en_decimal(ch)` qui prend en paramètre une chaîne de caractères représentant un entier en binaire et **renvoie** l'entier correspondant en base 10.
4. Écrire une fonction `bin_en_hexa(ch)` qui prend en paramètre une chaîne de caractères représentant un entier en binaire et **renvoie** une chaîne correspondant à sa représentation hexadécimale.
   On pourra commencer par compléter la chaîne avec des zéros à gauche pour obtenir une longueur multiple de 4.

Exercice 7 : Manipulation de caractères avec `ord` et `chr`.

1. Écrire une fonction `initial_est_majuscule(ch)` qui prend une chaîne de caractères en paramètre et **renvoie** `True` si le premier caractère est une majuscule, `False` sinon.
2. Écrire une fonction `mettre_majuscule_initiale(ch)` qui prend en paramètre une chaîne de caractères et **renvoie** la même chaîne avec le premier caractère en majuscule. Si la chaîne commence déjà par une majuscule ou par un chiffre, elle est renvoyée non modifiée.
3. Écrire une fonction `tout_en_majuscules(ch)` qui prend en paramètre une chaîne de caractères et **renvoie** la même chaîne avec chaque lettre minuscule remplacée par la majuscule correspondante.

Remarque : il existe des méthodes Python qui font cela directement (`'toto'.capitalize()`, `'toto'.upper()` et `'TOTO'.lower()`). Elles sont hors programme et ne sont pas à utiliser dans un devoir surveillé, mais vous pouvez les tester pour vérifier vos résultats.