Lycée Loges

Chapitre 3

Codages des nombres en Python Exercices

Simon Dauguet
simon.dauguet@gmail.com

14 novembre 2024

Exercice 1 :
Calculer (puis vérifier le résultat) la somme et le produit de :

1. $\overline{132}^4$ et $\overline{1002}^4$.
2. $\overline{46}^8$ et $\overline{307}^8$.
3. $\overline{0010\ 0011}^2$ et $\overline{0110\ 1100}^2$.

Exercice 2 :
Convertir en binaire et hexadécimal, les entiers :

1. 37, 487 et 1322.
2. 72, 761 et 3879.

Exercice 3 :
Déterminer la valeur décimale de :

1. $\overline{0001\ 1110}^2$, $\overline{1001\ 1011}^2$ et $\overline{B7}^{16}$.
2. $\overline{1101\ 1000}^2$, $\overline{0110\ 1101\ 0011}^2$ et $\overline{121}^{16}$.

Exercice 4 :
Convertir en complément à deux (binaire signé) sur 1 octet les entiers :

1. 54, -119, -43.
2. 121, -93 et -128.

Exercice 5 :
Déterminer la valeur décimale des binaires signés sur 1 octet en complément à 2 suivants :

1. $\overline{1001\ 0010}^2$ et $\overline{0111\ 1011}^2$.
2. $\overline{0001\ 1011}^2$ et $\overline{1001\ 1010}^2$.

Exercice 6 :
Déterminer la valeur décimale du flottant :

1. $0'1000\ 1111'1011\ 0001\ 0110\ 0100\ 1000\ 000$?
2. $1'0111\ 1100'1110\ 0000\ 0000\ 0000\ 0000\ 000$?

Exercice 7 :
Écrire les nombres suivant comme flottant simple précision, puis double précision :

1. 100
2. $-999,5.$
3. $-7,21875.$
4. $-10,67.$

Exercice 8 :
Écrire le nombre $n_i = 1 + 2^{-i}$ avec $i \in \mathbb{N}$ en tant que flottant 64 bits. À quelle condition sur i le flottant n_i est-il égal à 1 ?

Exercice 9 :
Expliquer pourquoi python renvoie $2^{-51} + 2 \neq 2$ mais $2^{-51} + 4 = 4$.

Exercice 10 :
Trouver la plus grande puissance de 2 de sorte qu'en python `x+1.==1`, soit vraie. On considérera des flottants double précision (64b, ceux qu'utilisent python). Et avec des flottants simple précision ? Et avec `x+1000.=1000`

Exercice 11 :
On souhaite coder l'opérateur ET bit à bit qui, appliquer à deux entiers, consiste à renvoyer l'entier résultant d'un ET entre chacun des bits des deux entiers. Par exemple, `etbits(5, 6)` renverra 4 car pour $5 = \overline{101}^2$ et $6 = \overline{110}^2$ le ET bit à bit donne $\overline{100}^2$ (Remarque : `&` implémente le ET bit à bit ; *i.e.* `5&6` renvoie 4).

1. En faisant les calculs, que renvoie le ET bit à bit entre 14 et 12 ? entre 11 et 19 ?
2. Existe-t-il un élément neutre pour le ET bit à bit lorsque les entiers sont codés sur 8 bits ? Un élément absorbant ?
3. Écrire une fonction `etbits(x: int, y :int) -> int`, qui réponde au problème sans utiliser `&`.

Exercice 12 (Codage de Lebesgue, (*X-ENS 2017, partie III*)) :
Le codage de Lebesgue d'un point de coordonnées entières (x, y) s'obtient en entrelaçant les bits des représentations binaires de x et y en commençant par les bits de x. Ainsi pour encodage sur $n = 3$ bits, si $x = 6 = \overline{110}^2$ et $y = 3 = \overline{011}^2$, le point $(6, 3)$ est codé par $\overline{101101}^2 = 45$.

De plus, on utilisera la notation décimale 0, 1, 2 et 3 pour représenter la base $\overline{00}^2$, $\overline{01}^2$, $\overline{10}^2$ et $\overline{11}^2$ respectivement. On notera $\overline{c_{n-1}\ldots c_1 c_0}^l$ la représentation en base 4 du codage de Lebesgue d'un point à coordonnées entières. Par exemple, le point (6,3) s'écrit $\overline{10}^2\overline{11}^2\overline{01}^2 = \overline{231}^l$ et s'écrira en python comme un tableau `[2,3,1]`.

1. Soit $n = 3$, quel tableau python représente le codage de Lebesgue du point $(1, 6)$?
2. Écrire une fonction `bits(x: int, k: int) -> int`, qui renvoie la valeur du bit de coefficient 2^k dans la représentation binaire de x.

3. Écrire une fonction `code(n: int, p: tuple) -> list`, qui prend en arguments un entier strictement positif n et un point p représenté par un tuple de longueur 2 représentant les coordonnées entières d'un point. Cette fonction renvoie le codage de Lebesgue de p représenté sous la forme d'un tableau de longueur n.

Exercice 13 (Système de numération avec chiffres négatifs (*ENS 2012, partie IV*)) :
On se place en base $b = 3$ avec comme symboles $\mathcal{S} = \{-1, 0, 1\}$.

Un entier $n = \sum_{i=0}^{k} a_i b^i$ sera représenté par le mot $(a_k \dots a_1 a_0)_3$, où les $a_i \in \mathcal{S}$. Par convention, le mot vide représente l'entier zéro. Pour simplifier les notations, on écrira $\overline{1}$ au lieu de -1. Ainsi le mot $(1\overline{1}0\overline{1}\overline{1}11)_3$ représente l'entier $3^6 - 3^5 - 3^3 - 3^2 + 3 + 1 = 454$.

La représentation machine d'un nombre entier sera stocké dans un tableau T de sorte que $\forall i,\ T[i] = a_i$.

1. Donner une représentation pour chaque entier de $[\![-5, 5]\!]$

   On appelle plus petite représentation de n, un mot sur $\mathcal{S}$, ne commençant pas par zéro, et représentant n. Le code suivant permet de l'obtenir et vous permettra de tester vos fonctions.

   Pour comprendre comment elle fonctionne, vous pouvez la dérouler pour $n = 454$.

```python
def representant3(n: int) -> list:
    if n==0: return []
    sgn = 1 if n >= 0 else -1    # vaut 1 si n>=0, -1 sinon
    p = ceil( log3(sgn*n) )
    if sgn*n < (3**p+1)/2: p-= 1
    tab = []
    while p >= 0:
        N = 3**p
        if abs(sgn*N-n) <= (N-1)/2:
            tab += [sgn]
            n -= sgn*N
        else:
            tab += [0]
        p -= 1
        sgn = 1 if n >= 0 else -1
    return tab
```

2. Proposer une fonction **`nombre(rep: list) -> int`**, qui renvoie le nombre représenté par le tableau **`rep`**.

   Pour les quatre dernières questions, on demande des algorithmes qui calculent directement à partir des chiffres de représentations, sur le modèle de l'algorithme d'addition des représentations en base 10 enseigné à l'école primaire. En particulier, il est interdit d'utiliser les fonctions multiplications et exponentiation du langage, et *a fortiori* toute fonction plus évoluée. Seules l'addition et les structures de contrôle élémentaires (tests, boucles, ...) sont autorisées. L'efficacité des algorithmes sera un élément à considérer.

3. Écrire une fonction qui prend en entrée un tableau représentant un nombre entier et renvoie un tableau représentant son opposé.
4. Écrire une fonction qui prend en entrée un tableau représentant un nombre entier et renvoie un tableau représentant son successeur.
5. Écrire une fonction qui prend en entrée deux tableaux représentant deux nombres entiers et renvoie la représentation de leur somme.
6. Écrire une fonction qui prend en entrée deux tableaux représentant deux nombres entiers et renvoie la représentation de leur différence.