TP 07

Algorithmes gloutons

Savoirs et compétences :

- ❐ *AA.C9 : Choisir un type de données en fonction d'un problème à résoudre*
- ❐ *AA.S11 : Manipulation de quelques structures de données.*
- ❐ *AA.S12 : Fichiers*

Exercice 1 – Obserations

Absorption

Question 1 *Tester dans le shell ces 3 propositions et discuter les résultats.*

```
>>> 1.0 + (2**53 - 2**53)
>>> (1.0 + 2**53) - 2**53
>>> (1 + 2**53) - 2**53
```

Des erreurs d'arrondi.

Question 2 *Tester dans le shell cette proposition et discuter le résultat.*

```
>>> (0.1+0.2) - 0.3 == 0
```

Phénomène de cancellation

Question 3 *Tester dans le shell ces 2 propositions et discuter les résultats.*

```
>>> 1/1000-1/1001\\
>>> 1/(1000*1001)
```

Recommandation : ne jamais tester l'égalité entre deux nombres flottants, mais tester si leur distance est inférieure à un nombre très petit.

Exercice 2 – Détermination du nombre de bit de la mantisse d'un flottant

On se propose de vérifier que le stockage de la mantisse d'un flottant `python` s'effectue sur 52 bits.

On note $m_c = m - 1$, `m` étant la mantisse du flottant. `mc` est la valeur stockée en mémoire en binaire.

L'idée est de se servir du nombre $0,5$ dont on connait parfaitement la décomposition binaire :

$$0,5 = \frac{1}{2} = 1 \times \frac{1}{2} + 0 \times \frac{1}{4} + 0 \times \frac{1}{8} \ldots$$

On observe qu'en divisant m_c successivement par 2 on obtient :

$m_c = 0.5$	$m = 1.5$	stockée en mémoire sous la forme	1000...000
$m_c = 0.25$	$m = 1.25$	stockée en mémoire sous la forme	0100...000
$m_c = 0.125$	$m = 1.125$	stockée en mémoire sous la forme	0010...000
...			
$m_c = 0.00...1$	$m = 1.00...1$	stockée en mémoire sous la forme	0000...001

Au bout d'un nombre suffisamment grand de divisions par 2 le chiffre 1 disparait complètement. En comptant le nombre de divisions par 2 nécessaires pour aboutir à 0, on a accès au nombre de bits disponibles pour coder m_c. On propose l'algorithme suivant :

Initialisation (à compléter)
Tant que $1 + m_c \neq 1$ **faire**
  $m_c \leftarrow m_c/2$
  $i \leftarrow i+1$
fin
retourner i

Question 4 *Commenter ou compléter l'algorithme proposé :*
- *compléter la partie initialisation;*
- *justifier le type de boucle choisie;*
- *repérer la condition d'arrêt;*
- *invariant de boucle;*
- *la boucle a-t-elle une fin?*
- *l'algorithme effectue t-il ce que l'on attend?*

Question 5 *Implémenter cet algorithme dans* `python`. *Conclure quant au nombre de bits disponibles pour coder la mantisse d'un flottant.*

Il est possible d'appliquer la méthode `hex()` sur un flottant pour avoir sa représentation en hexadécimal.

```
>>> f=5.25
>>> f.hex()
    '0x1.5000000000000p+2'
```

Cela dit que 5,25 est représentée par le nombre $1 \times (5000000000000)_{16} \times 2^2$ dont la mantisse est 1,5000000000000 et l'exposant 2.

Question 6 *Déterminer la mantisse de* $\sqrt{2}$ *à partir de son expression hexadécimale.*