Notions sur la précision des calculs flottants

Complément - niveau avancé

Le problème

Comme pour les entiers, les calculs sur les flottants sont, naturellement, réalisés par le processeur. Cependant contrairement au cas des entiers où les calculs sont toujours exacts, les flottants posent un problème de précision. Cela n'est pas propre au langage python, mais est dû à la technique de codage des nombres flottants sous forme binaire.

Voyons tout d'abord comment se matérialise le problème:

```
>_ 0.2 + 0.4
```

Il faut retenir que lorsqu'on écrit un nombre flottant sous forme décimale, la valeur utilisée en mémoire pour représenter ce nombre, parce que cette valeur est codée en binaire, ne représente **pas toujours exactement** le nombre entré.

```
>_ 0.3 - 0.1 == 0.2
```

Aussi comme on le voit les différentes erreurs d'arrondi qui se produisent à chaque étape du calcul s'accumulent et produisent un résultat parfois surprenant.

Dans une grande majorité des cas, ces erreurs d'arrondi ne sont pas pénalisantes. Il faut toutefois en être conscient car cela peut expliquer des comportements curieux.

Une solution : penser en termes de nombres rationnels

Tout d'abord si votre problème se pose bien en termes de nombres rationnels, il est alors tout à fait possible de le résoudre avec exactitude.

Alors qu'il n'est pas possible d'écrire exactement $3/10$ en base 2, ni d'ailleurs $1/3$ en base 10, on peut représenter **exactement** ces nombres dès lors qu'on les considère comme des fractions et qu'on les encode avec deux nombres entiers.

Python fournit en standard le module `fractions` qui permet de résoudre le problème. Voici comment on pourrait l'utiliser pour vérifier, cette fois avec succès, que $0.3 - 0.1$ vaut bien 0.2

En anticipant un tout petit peu sur l'utilisation des modules et des classes en Python, voici comment on pourrait créer des objets de type `Fraction` pour faire de manière ***exacte*** le calcul ci-dessus:

```
from fractions import Fraction
```

```
Fraction(3,10) - Fraction(1,10) == Fraction(2,10)
```

Ou encore aussi

```
Fraction('0.3') - Fraction('0.1') == Fraction('2/10')
```

Une autre solution : le module decimal

Si par contre vous ne manipulez pas des nombres rationnels et que la représentation sous forme de fractions ne peut pas convenir dans votre cas, signalons l'existence du module standard `decimal` qui offre des fonctionnalités très voisines du type `float`, tout en éliminant la plupart des inconvénients, au prix naturellement d'une consommation mémoire supérieure.

Pour reprendre l'exemple de départ, mais en utilisant le module decimal, on écrirait alors

```
from decimal import Decimal
```

```
Decimal('0.3') - Decimal('0.1') == Decimal('0.2')
```

Pour aller plus loin

Tous ces documents sont en anglais

- Un tutoriel sur les nombres flottants (https://docs.python.org/2/tutorial/floatingpoint.html)
- La documentation sur la classe Fraction (https://docs.python.org/2/library/fractions.html)
- La documentation sur la classe Decimal (https://docs.python.org/2/library/decimal.html)