


CHAPITRE 1 – LE SYSTÈME INFORMATIQUE

PARTIE 2 – SYSTÈMES D'EXPLOITATION

1	Manipulation d'entiers relatifs	1
2	Manipulation de nombres à virgule flottante, flottants	3


1 Manipulation d'entiers relatifs

Nous allons utiliser la console interactive de Python où chaque ligne tapée est immédiatement exécutée. Exécuter les commandes suivantes :



```
3 + 5
6-8
2 * 3
2 ** 3
```

Dans ce mode, on peut d'ores et déjà utiliser des variables pour stocker des valeurs. L'affectation s'écrit avec le symbole = et n'affiche aucune valeur ; mais la variable mémorise la valeur qu'on lui a donnée et peut être utilisée dans la suite de la session. Exécuter les commandes suivantes :



```
a = 2
a + 2
b = 6
b = b + 2
b = b + 2
b
```

La mémoire et l'unité de calcul d'un ordinateur sont constitués de composants qui ne peuvent prendre que deux états, désignés par convention 0 et 1. Une variable prenant valeur dans un ensemble où il n'y a que deux valeurs s'appelle un booléen, un chiffre binaire ou encore un bit (binary digit). Pour représenter un nombre binaire on utilise une suite finie de bits, que l'on appelle un mot ou encore un mot binaire. Les unités de calcul de nos ordinateurs utilisent des mots de 32 bits ou de 64 bits pour les plus récents. Quel est le nombre entier relatif le plus grand représentable sur un mot de 32 bits ? De même pour un mot de 64 bits. Exécuter les commandes suivantes :



```
a = 2147483647
a
```

La valeur du résultat d'un calcul pourrait dépasser les capacités d'un mot de 32 ou 64 bits. On appelle ce phénomène dépassement arithmétique (overflow en anglais). Exécuter les commandes suivantes :



```
a = a + 1
a
```

Il faudrait changer de représentation pour éviter de perdre la valeur du résultat. On constate qu'en Python, ce changement de représentation est fait automatiquement, ce n'est pas le cas dans tous les langages de programmation. En Python, lorsque la capacité des entiers machine (32 ou 64 bits) a été dépassée, les nombres sont suivis du marqueur L, qui explicite que l'entier est devenu Long. Ceci pour les versions Python 2.x. Pour les versions Python 3.x, le marqueur L n'est plus utilisé. Exécuter les commandes suivantes :



```
2 ** 32 - 1
2 ** 31 - 1
2**30-1
```

Comparer la valeur de a, proposée précédemment, aux trois résultats de $2^{32} - 1$, $2^{31} - 1$ et $2^{30} - 1$. On savait qu'un bit était réservé au signe. Ainsi, l'entier relatif le plus grand représentable sur 32 bits devrait être $2^{31} - 1$. Pour changer la représentation des entiers, en cas d'overflow, Python utilise plusieurs mots binaires associés et réserve un bit supplémentaire. Il n'est pas utile d'en savoir plus, les curieux se référeront à (livre informatique pour tous page 51) Les valeurs en Python sont typées, autrement dit, elles sont classées selon l'objet qu'elles représentent. Une valeur peut ainsi être de type entier, flottant, booléen, chaîne de caractères, Exécuter les commandes suivantes :



```
a = 2
type(a)
```

1

Où l'on constate que le type de la valeur de a est le type entier (integer, en anglais).

2 Manipulation de nombres à virgule flottante, flottants

Exécuter les commandes suivantes :



```
a = 2.0  
type(a)
```

Où l'on constate que le type de la valeur de a est le type flottant (float, en anglais).

Exécuter les commandes suivantes :



```
a = 1 + 1  
type(a)  
a = 1 + 1.0  
type(a)
```

Une expression est une suite de caractères définissant une valeur. Par exemple ici l'expression $x + y$ avec $x = 1$ et $y = 1$ ou 1.0 définit la valeur 2. On constate qu'une expression en Python n'a pas de type car le type de sa valeur dépend des types de ses sous-expressions. Exécuter les commandes suivantes :



```
5 / 3  
5.0 / 3
```

Selon la version de Python l'opérateur / calcule le quotient de la division euclidienne de deux entiers ou le quotient décimal. Dans les versions 3.x, l'opérateur / n'existe que pour les flottants, et donc en l'appliquant à des entiers, ils sont convertis en flottants, et le résultat est un flottant égal à leur quotient décimal.

Exécuter les commandes suivantes :



```
5.0 / 3  
b = 5 // 3  
b  
c = 5.0 % 3  
c  
b * 3 + c
```

On vient de voir le calcul du quotient et du reste de la division euclidienne. Dans le contexte de la programmation on parle de division entière et de modulo plutôt que de quotient et de reste. Exécuter les commandes suivantes :



```
2 ** 100
2.0 ** 100
```

Ici on remarque que les flottants offrent une précision moins bonne que les entiers pour les calculs sur des valeurs entières. Exécuter les commandes suivantes :



```
a = 5.0 / 3
b = 100 * a
c = 500.0 / 3
b - c
```

La représentation en virgule flottante d'un nombre est forcément une valeur approchée de ce nombre.

On se propose de calculer : $1 - 2 + 2^x$ pour deux valeurs de $x = -52$ et $x = -53$ d'une part et d'autre part de deux manières différentes. Exécuter les commandes suivantes :



```
1 - 1 + 2** -52
1 + 2** -52 - 1
```

Normalement c'est cohérent. Exécuter les commandes suivantes :



```
1 - 1 + 2** -53
1 + 2** -53 - 1
```

À nouveau, on remarque que la représentation en virgule flottante donne une valeur approchée d'un nombre. On rappelle que la représentation d'un nombre en flottant est de la forme : . Avec le signe, la mantisse et l'exposant. La mantisse est un nombre binaire à virgule compris entre 1 inclus et 2 exclu. L'exposant est un entier relatif.

En Python les nombres sont représentés sur 64 bits de la manière suivante :

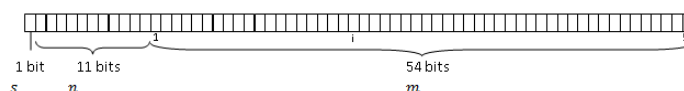


FIGURE 1

Comme la mantisse a toujours le chiffre 1 avant la virgule, il est inutile de le représenter et on utilise les 52 bits pour la représenter. Ainsi,

$$s \cdot m \cdot 2^n = (-1)^s \cdot \left(1 + \sum_{i=1}^{52} m_i \cdot \frac{1}{2^i} \right) \cdot 2^n$$

FIGURE 2

Il a quelques cas particuliers qu'on n'évoquera pas ici. Exprimer la mantisse et l'exposant de 2^{-52} , de même pour $1 + 2^{-52}$. Expliquer alors l'erreur de calcul pour $x = -53$. Utiliser Python comme une calculatrice a finalement peu d'intérêt.

Références

[1] Christophe François, Représentation de l'information, représentation des nombres.