CPES Scientifique

Informatique

TP 1

Expressions Python :
Nombres entiers, nombres flottants et Booléens

Résumé

Dans ce premier TP, on va voir la console Python comme une calculatrice. Le but est de se familiariser avec quelques concepts de base utlisés en Python (ainsi que dans de nombreux langages de programmation). Il concerne ce que l'on appelle des **types de données** et se concentre sur quelques types de base : les nombres entiers, les nombres à virgule flottante, les Booléens et leurs opérations principales.

1 Les nombres entiers

Les nombres entiers s'écrivent normalement, comme des suites de chiffres, éventuellement précédées par un signe « - » pour les nombres négatifs.

Question 1.1. *Écrire dans la console les trois nombres suivants :* 0, 42 *et* −112.

Les entiers sont munis des opérations arithmétiques usuelles, répertoriées dans la Table 1 page 1, auxquelles s'ajouttent la division (« `/` » ou « `//` ») et l'opérateur « `%` » qui renvoie le **reste** de la division entière.

`+`	:	addition
`*`	:	multiplication
`**`	:	exponentiation (« puissance »)
`-`	:	soustraction

Table 1 – Principales opérations arithmétiques

Question 1.2. *Utiliser la console Python pour calculer :*

(a) 2^{30}, 2^{31}, 2^{32}, 2^{62} *et* 2^{63},

(b) les longueurs des représentations binaires des nombres ci-dessus (p. ex. avec `(2**32).bit_length()`*),*

(c) le quotient et le reste de la division euclidienne de 41 *par* 3,

(d) le quotient de la division euclidienne de 41 *par* 0. *Que lit-on dans la console ?*

Attention : Les opérations ont des règles de priorité.

Question 1.3 (Précédence et parenthésage)**.**

(a) Utiliser la console Python pour calculer $(3 + 4) \times 5$ *et* $3 + (4 \times 5)$.

(b) Que vaut $3 + 4 \times 5$ *?*

2 Les nombres à virgule flottante

Les « nombres à virgule flottante » sont des nombres décimaux. Ils s'écrivent comme les nombres décimaux usuels, sauf qu'on utilise le point « . » au lieu de la virgule.

Les nombres à virgule flottante ont les mêmes opérations de base que les nombres entiers (mis à part la division entière), répertoriées Table 1 page 1, auxquelles il faut ajoutter la **division** « _/_ ».

Question 2.1. *Utiliser la console Python pour calculer :*

(a) $41/3$

(b) 2^{-1000}, 2^{-1100}

Attention : Les nombres à virgule flottante ont une précision et une taille fixée.

Question 2.2. *Comparer les résultats Python des calculs de* 2^{53}, $2^{53}+1$ *et* $2^{53}+1.0$.

Question* 2.3 (Plus petit nombre strictement positif)**.** *En utilisant l'exponentiation, essayer de déterminer quel est le plus petit nombre à virgule flottante strictement positif.*

3 Les Booléens

Les « Booléens » servent à représenter ce que l'on appelle des « valeurs de vérité ». Le plus simple est de voir leur utilisation par les **opérateurs de comparaison**.

3.1 Les opérateurs de comparaison

Les **opérateurs de comparaison** servent à comparer des nombres. Les principaux sont répertoriés dans la Table 2 page 2.

`==`	:	égalité
`!=`	:	inégalité
`<=`	:	inférieur ou égal
`<`	:	inférieur strict
`>=`	:	supérieur ou égal
`>`	:	supérieur strict

TABLE 2 – Principaux opérateurs de comparaison

Question 3.1. *Tester dans la console Python*

(a) Si 3 *est inférieur ou égal à* 3.

(b) Si 3 *est strictement supérieur à* 5.

(c) Si 2^{53} *est strictement inferieur à* $2^{53}+1$ *et à* $2^{53}+1,0$.

(d) L'égalité de 2^{53} *et* $2^{53}+1$ *ainsi que l'égalité de* 2^{53} *et* $2^{53}+1,0$,

Dans chacun des cas ci-dessus, que renvoie la console Python ?

3.2 Les Booléens

Les valeurs de vérité Booléennes sont les constantes « `True` » (pour « **Vrai** ») et « `False` » (pour « **Faux** »). Ces valeurs de vérités sont munies des opérations « `and` » (pour le « **Et** » logique) « `or` » (pour le « **Ou** » logique) et « `not` » (pour la négation).

Question 3.2 (Tables de vérité des opérateurs)**.** *Utiliser la console Python pour déterminer les tables de vérité (c'est-à-dire l'ensemble des valeurs possibles) des opérateurs* `and`*,* `or` *et* `not`*.*

Question 3.3 (Opérateurs non-stricts)**.** *Comparer l'exécution des instructions suivantes :*

- $1/0$ `and False`
- `False and` $1/0$
- `True and` $1/0$

De même avec

- $1/0$ `or True`
- `True or` $1/0$
- `False or` $1/0$

Question 3.4 (Opérations bit à bit)**.** *Les opérateurs « `&` » et « | » sont des variantes de* `and` *et* `or`*.*

Ils ont entre autres l'intérêt d'implémenter la comparaison bit-à-bit des entiers. Par exemple, $2|1$ *renvoie l'entier dont la représentation binaire s'écrit comme le* `or` *des représentations binaires de* 2 *et* 1*, c'est-à-dire l'entier* 3*.*

- *Calculer sur papier le résultat de* $2\&1$*, de* $4|1$ *et de* $4\&4$*, et comparer avec le résultat de la console Python.*