Université de Tours
Licence 1 Informatique

TP 1 : Informatique

M1-3 Informatique 1 Semestre 1

Exercice 1 (`Int` vs `Float`)**.** Dans ce qui suit on utilise le symbole ; pour séparer des instructions distinctes.

1. Effectuer les opérations suivantes.

   ```
   10 * 10; 10 + 10; 10 - 1; 10 ** 3; 100 // 3; 100 / 3; 100 % 3
   ```

2. Effectuer les opérations suivantes.

   ```
   10. * 10; 10 + 10.; 10. - 1; 10. ** 3; 100. // 3; 100 / 3.; 100. % 3
   ```

3. Quelles sont **les** différences entre les deux suites de commande précédentes ?

Exercice 2 (Autres notations numériques)**.** On va voir ici d'autres façons, dans certains cas plus pratiques, de rentrer des nombres.

1. Évaluer les expressions suivantes.

   ```
   0.000001 == 1e-6; 1000 == 1e3; 0.0134 == 1.34e-2
   ```

2. Qu'en déduisez-vous quant à la notation `_e_` ?

Exercice 3 (Erreurs d'arrondis et `fractions`)**.**

1. Est-il vrai mathématiquement que

$$\frac{(10^{10} + 10^{-10})}{10^{10}} = 1$$

2. Et en python ?
3. Évaluer l'expression `1eX + 1e-X` en remplaçant `X` par 5, 7, 9 et 10.
4. Entrer le code suivant

   ```
   from fractions import Fraction
   a = Fraction(1, 2)
   print(a)
   b = Fraction(1, 3)
   print(b)
   print(a + b)
   ```

5. Comment interpréter `Fraction(a, b)` ?
6. Dans le code suivant que vaut `Y`

   ```
   X = 10 ** 10
   Y = Fraction(X + Fraction(1, X), X)
   ```

Exercice 4 (Précédence des opérateurs)**.** On va explorer ici l'importance des parenthèses dans les expressions numériques.

1. Évaluer

   ```
   1 + 2 - 3 * 5 == (1 + (2 - (3 * 5)))
   ```

2. Déterminer un parenthésage des expressions suivants de façon à ce qu'un seul opérateur apparaisse dans un bloc de parenthèse. (Comme pour l'expression ci-dessus)

```
3 * 5 / 2; 1 / 2 + 3 / 4; 1 + 2 * 3 ** 4; 1 + 2 / 3 + 4
```

3. Déterminer les niveaux de précédence

Exercice 5 (Chaines de caractères)**.**

1. Quelle est l'action de l'opérateur `+` entre deux chaines de caractères ?
2. Quelle est l'action de l'opérateur `*` entre une chaine de caractère et un nombre ?
3. l'expression suivante est-elle valide ?

```
"a" * 2 + "b" * 3 + "c" * 4
```

4. Les opérateurs `-` et `/` peuvent ils être utilisés avec les chaines de caractères ?
5. Exécuter l'instruction suivante

```
'ab' * 5 == 'a' + 'ba' * 4 + 'b'
```

6. On peut tester l'égalité entre deux chaines de caractères comme le montre la question précédente. Qu'en est-il des opérateurs de comparaisons `<`, `>`, etc...

Exercice 6 (Identités arithmétiques)**.** On va explorer de manière plus systématique différentes propriétés arithmétiques en python.

1. Pour déterminer si l'identité mathématique suivante est vrai

$$\forall x, y, z \in \mathbb{Z}, \qquad x(yz) = (xy)z.$$

Exécuter le code suivant

```
from random import randint

minimum, maximum = -10, 10
nb_iterations = 1000
compteur = 0

while compteur < nb_iterations:
    compteur = compteur + 1
    a, b, c = (
        randint(minimum, maximum),
        randint(minimum, maximum),
        randint(minimum, maximum),
    )
    if a * (b * c) != (a * b) * c:
        print("La multiplication n'est pas associative!")
```

2. Que fait le code précédent ? L'identité est-elle vraie ?
3. De la même façon tester $a + b = b + a$, $a + (b + c) = (a + b) + c$, et $a - (b - c) = (a - b) - c$

Exercice 7 (Ordre opérateurs)**.** On cherche à déterminer les priorités entre opérateurs. En d'autres termes, on cherche à déterminer où `python` décide de mettre les parenthèses lorsque celles-ci ne sont pas indiquées dans l'expression. Par exemple si on tape `2-2-10`, le résultat est -10 ce qui indique que `python` a calculé $((2 - 2) - 10)$ et non pas $(2 - (2 - 10))$. On s'inspirera de l'exercice précédent.

1. Vérifier que les additions et les soustractions sont juste parenthésée de gauche à droite par exemple via les identités $a + b - c + d = (((a + b) - c) + d)$, $a - b - c + d = (((a - b) - c) - d)$ et une autre identité de votre choix.
2. De la même façon vérifier que l'on a la même façon de parenthéser entre les multiplications et les divisions via $a * b/c * d = (((a * b)/c) * d)$ et deux autres identités de votre choix.

3. Montrer que les opérateurs multiplications/divisions sont parenthésés avant les opérateurs additions/soustractions via $a+b*c-d/e = ((a+(b*c))-(d/e))$ et deux autres identités de votre choix.
4. Vérifier enfin que l'exponentiation est prioritaire sur toutes les précédentes.

Exercice 8 (Pile ou Face)**.** On va s'intéresser ici au jeu de pile ou face.

1. Exécuter le script suivant

```
from random import random

nb_lancers = 20
compteur = 0
while compteur < nb_lancers:
    compteur = compteur + 1
    alea = random()
    if alea < 0.5:
        print("Pile")
    else:
        print("Face")
```

2. Ajuster le script pour stocker le nombre de Pile dans une variable `nb_piles`.
3. Semble-t-il expérimentalement vrai que le nombre de pile est égal au nombre de faces lorsque le nombre de lancers devient très grand comme le suggère le fait que la pièce a une chance sur deux de tomber sur chaque face ?
4. Calculer le rapport `nb_piles` / `nb_lancers` pour `nb_lancers` valant 10, 100, 1000, 10000, 100000, 1000000.
5. Que constatez-vous ?
6. S'il vous reste du temps utiliser des vrais lancers de pièces pour étudier la question.