TP - Récursivité

M. Tellene

Conseil d'organisation : il vous est conseillé pour ce TP de créer dans votre dossier personnel, un dossier « recursivité ». C'est ce dossier qui contiendra le travail demandé lors de ce TP. De la même manière, il vous est conseillé de faire un fichier python par exercice.

Exercice 1

La factorielle d'un nombre n correspond à l'opération $1 \times 2 \times ... \times n$, c'est-à-dire le produit de tous les nombres inférieurs ou égaux à n. Il est à noter que la factorielle s'arrête à 1.

Écrire une fonction récursive `facto` permettant de calculer la factorielle de n.

Exercice 2

La récursivité permet, en plus de résoudre des problèmes mathématiques, de faire des opérations sur des structures de données. Cet exercice va s'intéresser à la manipulation de tableaux grâce à des fonctions récursives.

1. Écrire une fonction récursive `somme_tableau` qui renvoie la somme des éléments du tableau passé en paramètre
2. Écrire une fonction récursive `concaténation` qui renvoie la concaténation de deux tableaux qui seront passés en paramètre.

   Petit rappel : la concaténation de deux tableaux se fait grâce à l'opérateur « + »et consiste à créer un tableau à partir de deux

```
#Exemple de concaténation
l1 = [1,2,3]
l2 = [4,5,6]
l3 = l1 + l2
print(l3)
>>> [1,2,3,4,5,6]
```

3. Écrire une fonction récursive `renverse` qui renvoie le tableau passé en paramètre renversé.
4. Écrire une fonction récursive `nb_occ` qui renvoie le nombre d'occurrences d'un élément (passé en paramètre) dans un tableau, lui aussi passé paramètre.

Exercice 3

La suite de Fibonacci a été vue lors du dernier TD, pour rappel, voici sa définition :

$$F_n = \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F_{n-1} + F_{n-2} & sinon \end{cases}$$

1. Écrire une fonction récursive `fibo` qui renvoie la valeur du n^{ime} élément de la suite

2. Il est donné ci-dessous une version itérative (sans récursivité) de `fibo` :

```
def fibo_ite(n):
    premier = 0
    deuxieme = 1

    if n == 0:
        return premier

    for _ in range(n):
        prochain = premier + deuxieme
        premier = deuxieme
        deuxieme = prochain

    return deuxieme
```

La récursivité est une bonne méthode de programmation, mais peut parfois mener à des programmes moins performants que la version itérative.

Nous allons comparer l'efficacité des deux algorithmes. pour ce faire, nous utiliserons le module `time`.

(a) Mettre la fonction `fibo` et `fibo_ite` dans le même fichier (si ce n'est pas déjà fait)

(b) Importer la fonction `time` du module `time`

(c) Utiliser la fonction `time` pour mesurer le temps d'exécution. Pour se faire voici la démarche à suivre :

```
depart = time()
#### appel de la fonction à tester ####
arrivee = time()

# temps total d'exécution de la fonction s'obtient par arrivee - depart
print(arrivee-depart)
```

(d) Remplir le tableau

`n`	temps de `fibo` récursif	temps de `fibo` itératif
5	1.45e-05	1.45e-05
10		
15		
20		
25		
30		
33		
34		
35		
36		