

# Algorithmes et Pensée Computationnelle

## Structure de données, Itération et récursivité - Exercices Avancés

Le but de cette séance est de mettre en pratique les connaissances apprises en cours. Elle portera essentiellement sur les structures de données qui seront utilisées lors des prochaines séances de cours/Travaux Pratiques. Les structures de données abordées lors de cette séance sont les tuples, les listes et les maps. Lors de cette séance, nous aborderons aussi les notions d'itération et de récursivité.

Au terme de cette séance, l'étudiant sera capable de distinguer une structure de données immuable et non immuable, écrire un programme de façon simplifiée en utilisant les notions d'itération et de récursivité.

Cette feuille d'exercices avancés vous permettra d'approfondir vos connaissances des notions vues en cours. Le code présenté dans les énoncés se trouve sur Moodle, dans le dossier "Ressources".

## 1 L'itération

Quelques rappels de concepts théoriques :

- L'itération désigne l'action de répéter un processus (généralement à l'aide d'une boucle) jusqu'à ce qu'une condition particulière soit remplie.
- Il y a deux types de boucles qui sont majoritairement utilisées :

**Boucle for** : Une boucle **for** permet d'itérer sur un ensemble. Cet ensemble peut être une liste, un dictionnaire, une collection, ... La syntaxe d'une boucle en Python est la suivante **for nom de variable in** suivi du nom de l'élément sur lequel vous voulez itérer. La variable prendra la valeur de chaque élément dans la liste, un par un.

La syntaxe en java est la suivante : **for (type nom.de.variable; condition de fin de boucle; incrémentation à chaque itération)** suivi d'une accolade.

**Boucle while** : Les boucles **while** sont des boucles qui s'exécutent de façon continue jusqu'à ce qu'une condition soit remplie. La syntaxe est la suivante : En Python, on écrit d'abord **while**, suivi de la condition à atteindre.

En Java, il n'y a pas de différence majeure à part le fait qu'il faille mettre entre parenthèses la condition et les deux points sont remplacés par des accolades.

- La fonction **range(n)** permet de créer une liste de nombres de 0 à la valeur passée en argument moins 1 (n-1). Lorsqu'elle est combinée à une boucle **for**, on peut itérer sur une liste de nombres de 0 à n-1.
- Si vous avez fait une erreur dans votre code, il se peut que la boucle **while** ne remplisse jamais la condition lui permettant de sortir. On parle dans ce cas d'une **boucle infinie**. Ceci peut entraîner un crash de votre programme, voire même de votre ordinateur. Il faut toujours s'assurer qu'il y ait une condition valide dans une boucle **while**.

### Question 1: (🕒 10 minutes) Boucle while et des étudiants

Considérons une liste d'étudiant-e-s contenant le nom de l'étudiant-e suivi d'une valeur booléenne indiquant si l'étudiant-e est présent-e en classe ou non.

```
l = ["Schmitt", True, "Irma", False, "Khalif", True, "Yasser", False, "Wang", True]
```

À partir de cette liste et du type de boucle de votre choix, créez un dictionnaire ayant pour clés les noms et pour valeur un booléen indiquant la présence au cours.

### Conseil

- `i % 2` : renvoie le reste de `i` par la division euclidienne de 2. (`i % 2 == 0`) Indique si `i` est pair.
- Si vous utilisez la boucle `for` : Utiliser les indices pour accéder aux éléments de la liste et éviter d'itérer manuellement sur les éléments de la liste.
- Si vous utilisez la boucle `while` : La condition de sortie devrait être en rapport avec la longueur de la liste.

### Question 2: (🕒 5 minutes) Différence entre Itération et Récursivité

Quel est la différence dans le traitement de problèmes entre l'itération et la récursivité ?

### Conseil

- L'itération traite les problèmes de manière "contrôlée". En effet, lors d'un appel itératif, la boucle est répétée tant que la condition est vérifiée.
- La récursivité résout les problèmes en résolvant des sous-problèmes de ceux-ci.

## 2 La récursivité

La récursivité est le fait d'appeler une fonction de façon récursive c'est-à-dire une fonction qui s'appelle elle-même de façon directe ou indirecte.

### Question 3: (🕒 5 minutes) Lecture de code - 3 Optionnel

Qu'affiche le programme suivant ?

```
1 def fct_a(mot) :
2
3 if len(mot) == 1 :
4     if mot[0] == "a" :
5         return 1
6     else :
7         return 0
8 else :
9     if mot[0] == "a" :
10        return 1 + fct_a(mot[1:])
11    else :
12        return 0 + fct_a(mot[1:])
13
14 print(fct_a("blablaba"))
```

Choisissez parmi les possibilités suivantes :

- 1
- 2
- 3
- 4

A quoi sert la fonction `fct_a(mot)` ?

### Conseil

Aidez vous d'un papier et effectuez les étapes une à une afin de bien comprendre comment le code est articulé.

**Question 4:** (🕒 10 minutes) **Fonction factoriel()**

Ecrivez la fonction **factoriel()** suivant une approche récursive.

Pour rappel, cette fonction prend un entier et retourne le factoriel de ce dernier tel que :

$\text{factoriel}(1) = 1$ ,  $\text{factoriel}(2) = 1 * 2 = 2$ ,  $\text{factoriel}(3) = 1 * 2 * 3 = 6$ ,  $\text{factoriel}(4) = 1 * 2 * 3 * 4 = 24, \dots$

💡 **Conseil**

On peut écrire cette fonction comme suit :

$f(n) = f(n) * f(n-1)$  si  $n > 0$

$f(n) = 1$  si  $n = 0$

**Question 5:** (🕒 15 minutes) **Fibonacci - Java**

Ecrivez deux fonctions permettant de calculer un nombre donné de la suite de Fibonacci. L'une doit utiliser la récursivité, et l'autre l'itération. Pour rappel, chaque élément de la suite de Fibonacci est la somme des deux derniers éléments. L'élément 0 vaut 0, l'élément 1 vaut 1 et l'élément 2 vaut 1.

Début de la suite : [0,1,1,2,3,5,8,13,...]

💡 **Conseil**

L'algorithme permettant de faire une suite de Fibonacci a été présenté dans les diapositives du cours. En cas de difficulté, n'hésitez pas à vous y référer.