Algorithmes et Structures de données | 2021

Journée 2 – Vendredi 12 février

Exercice 1 – Appel de fonction et exécution pas-à-pas

Le but de cet exercice est d'entraîner l'appel de fonction et l'exécution pas-à-pas, avec observation des variables locales, en Java et en Python. Pour ce faire vous allez implémenter dans ces deux langage la fonction `isLeap()` vue dans les slides du Module 2 (Programming Basics), l'appeler depuis le programme principal et exécuter celui-ci pas-à-pas. Attention, vous ne devez faire aucune entrées / sorties dans cet exercice.

Exercice 2 – Utilisation des entrées / sorties en ligne de commande

Le but de cet exercice est d'entraîner les entrées / sorties en ligne de commande, en Java et en Python. Pour ce faire, vous allez modifier le programme de l'exercice précédent de manière à demander interactivement à l'utilisateur d'entrer une année sur la ligne de commande, suivi de return. Après ça, votre programme devra afficher un texte indiquant à l'utilisateur si l'année introduite est bissextile ou non.

Exercice 3 – Traitement des arguments en ligne de commande

Le but de cet exercice est d'entraîner le traitement des arguments passés sur la ligne de commande avant l'exécution du programme, en Python et en Java. Pour ce faire, vous allez étendre le programme de l'exercice précédent de manière à permettre à l'utilisateur de passer une année en ligne de commande avant l'exécution du programme, plutôt que de l'introduire interactivement pendant l'exécution du programme. Dans le cas où aucun argument n'est passé, le programme doit retomber alors sur le comportement de l'exercice précédent, en demandant interactivement à l'utilisateur d'entrer une année.

Exercice 4 – Itération sur les arguments en ligne de commande

Le but de cet exercice est d'entraîner le traitement itératif des arguments passés sur la ligne de commande avant l'exécution du programme, en Python et en Java. Pour ce faire, vous allez étendre le programme de l'exercice précédent de manière à permettre à l'utilisateur de passer plusieurs année sen ligne de commande avant l'exécution du programme. L'utilisation d'une boucle *for-each* est fortement conseillée dans ce cas-là, puisqu'il s'agit d'itérer sur une collection.

Exercice 5 – Itération sur un tableau associatif

Le but de cet exercice est d'entraîner le traitement itératif des éléments d'un tableau associatif, en Java et en Python. Pour ce faire, vous allez simplement reprendre le code vu à la Slide 20 du Module 3 (Iteration & Recursion) et y ajouter le Mont Blanc, qui culmine à 4'808 mètres.

Exercice 6 – Itération sur un tableau associatif avec condition d'arrêt

Le but de cet exercice est d'entraîner le traitement itératif des éléments d'un tableau associatif avec condition d'arrêt, en Java et en Python. Pour ce faire, vous allez remplacer la boucle *for-each* de l'exercice précédent par une boucle *while*, qui devra s'arrêter dès que la dénivelée totale atteint ou dépasse 10'000 mètres.

Indication. Pour cela, vous devez transformer l'ensemble des clés du tableau associatif en une collection sur laquelle il est possible d'itérer avec une boucle while. En Python et en Java, vous pouvez le faire en créant une liste ou un tableau à partir de l'ensemble des clés. En Java, une alternative consiste à utiliser un itérateur. Quel avantage présente selon vous l'utilisation d'un itérateur en Java par rapport à l'utilisation d'un tableau ou d'une liste de clés ?

Exercice 7 – Factorielle en Python et en Java

Le but de cet exercice est d'appréhender la différence entre la version itérative et la version récursive d'une même fonction, en exécutant pas-à-pas la factorielle, en Java et en Python. Pour ce faire, vous pouvez partir de la version en Python donnée dans les slides du Module 3 (Iteration & Recursion), puis écrire une version équivalente en Java.

Observation. Essayez d'appeler les différentes versions, itératives et récursives en Java et en Python, avec n = 100. Qu'observez-vous ? Comment expliquer ces différents comportements ?

Exercice optionnel. Implémentez, en Java ou en Python, une version $factorial\_abs(n)$ de la fonction factorielle qui gère également les nombres négatifs de la manière suivante :

$$factorial\_abs(n) = sign(n) \times factorial(\ |n|\ )$$

Exercice 8 – Dérécursification d'une fonction

Soit la fonction récursive suivante en Python :

```
def fool(n):
    if n <= 0:
        return ''
    else:
        return fool(n//2) + str(n%2)
```

Indication. En Python, l'opérateur `//` désigne la division entière et l'opérateur `%` désigne l'opération *modulo*, autrement dit le reste de cette division. Par exemple, on a que :

- `13//2 = 6` *et* `12//2 = 6`
- `13%2 = 1` *mais* `12%2 = 0`.

a. A votre avis, que fait la fonction `fool(n)` ?

b. Ecrivez une version itérative équivalente cette fonction.

Exercice optionnel. Implémentez les variantes récursives et itératives de ces deux fonctions en Java.