4TPM205U: Algorithmique des tableaux: feuille 1

Itérer dans un intervalle avec la boucle `for`

Travaux dirigés

Dans toutes les fonctions de la feuille on réalisera les itérations en utilisant la boucle `for`.

Exercice 1 – On considère les fonctions :

```
def mystere1(a,b):
    for i in range(a,b):
        print(i)
```

```
def mystere2(a,b):
    for i in range(a,b,-1):
        print(i)
```

1. Qu'obtient-on si on appelle :
   i) `mystere1(10,20)` et `mystere1(20,10)` ?
   ii) `mystere2(10,20)` et `mystere2(20,10)` ?
2. Soit n un nombre naturel, écrire une fonction `affichePairsPlusPetitsQue(n)` qui affiche tous les nombres naturels pairs strictement inférieurs à n.
3. Écrire une fonction `existeDiviseur(k,a,b)` qui renvoie `True` s'il existe un diviseur de `k` dans l'intervalle fermé $[\![a, b]\!]$, et renvoie `False` sinon.

Exercice 2 – On considère la suite définie par :

$$\begin{cases} u_0 = 2 \\ u_n = 3 \times u_{n-1} - 1 \end{cases}$$

Écrire une fonction `suite(n)` qui prend comme paramètre un entier naturel `n` et calcule (et renvoie) le n-ième terme ($n \geq 0$) de la suite.

Exemple : l'appel `suite(4)` doit renvoyer 122.

Exercice 3 – Suite de Fibonacci.
Écrire une fonction `fibonacci(n)` qui calcule (et renvoie) le n-ième terme ($n \geq 0$) de la suite de Fibonacci définie par :

$$\begin{cases} u_0 = u_1 = 1 \\ u_n = u_{n-1} + u_{n-2} \end{cases}$$

Exercice 4 – Factorielle itérative.
On définit la factorielle d'un entier positif ou nul par :

$$\begin{cases} 0! = 1! = 1 \\ \forall n > 1,\ n! = n \times (n-1) \times (n-2) \times \cdots \times 2 \times 1 \end{cases}$$

En utilisant cette définition, écrire une fonction `factorielle(n)` qui calcule (et renvoie) $n!$.

Exercice 5 – On souhaite écrire une fonction `sommeFactorielles(n)` qui calcule (et renvoie) la somme $0! + 1! + 2! + \ldots + n!$.

1. Écrire une première version de `sommeFactorielles(n)` en utilisant la fonction `factorielle` déjà écrite précédemment.
   Combien de multiplications sont nécessaires pour calculer `sommeFactorielle(5)` ?
2. Modifier la version précédente pour calculer le résultat sans utiliser d'appel à la fonction `factorielle`.
   Combien de multiplications sont nécessaires maintenant pour calculer `sommeFactorielle(5)` ?

Moralité : C'est en général très bien d'utiliser des fonctions déjà écrites, mais . . . l'expression "*en général*" ne signifie pas l'expression "*toujours*", pourvu qu'on obtienne un code lisible qui effectue un nombre significativement inférieur d'opérations.