4TPM205U: Array Algorithms : Sheet 1

Iterate a given interval with the `for` loop

Travaux dirigés

In all the functions of the sheet, we implement the iterations utilizing the loop `for`.

Exercice 1 – Consider the following functions :

```
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. What do the following function calls print :
   i) `mystere1(10,20)` and `mystere1(20,10)` ?
   ii) `mystere2(10,20)` and `mystere2(20,10)` ?
2. Let n be a natural number, write a function `affichePairsPlusPetitsQue(n)` which shows all the even natural numbers strictly less than n.
3. Write a function `existeDiviseur(k,a,b)` which returns `True` if there exists a divisor of `k` in the closed interval $[\![a, b]\!]$, and returns `False` otherwise.

Exercice 2 – Consider a sequence defined by :

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

Write a function `suite(n)` which takes as a parameter, an natural integer `n` and calculates (and returns) the n^{th} term ($n \geq 0$) of the sequence.

Example : the function call `suite(4)` must return 122.

Exercice 3 – Fibonacci sequence.
Write a function `fibonacci(n)` which calculates (and returns) the n^{th} term ($n \geq 0$) of the Fibonacci sequence defined by :

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

Exercice 4 – Factorial (iterative).
We define the factorial of a non-negative integer :

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

Using this definition, write a function `factorielle(n)` which calculates (and returns) $n!$.

Exercice 5 – We would like to write a function `sommeFactorielles(n)` which calculates (and returns) the sum $0! + 1! + 2! + \ldots + n!$.

1. Write a first version of `sommeFactorielles(n)` using the function `factorielle` written already.
   How many multiplications are necessary to calculate `sommeFactorielle(5)` ?
2. Modify the previous version to calculate the result without using the call to the function `factorielle`.
   How many multiplications are necessary now, to calculate `sommeFactorielle(5)` ?

Moral : It is, generally, good to use the functions that are already written, but not necessarily always. The goal is to write a readable code which performs significantly less number of opérations.