Programmation récursive :

Exercice 1 :

On considère la fonction`mystere(n)` définit par le code :

```
def mystere(n):
    if n == 0:
        return 1
    else:
        return n * mystere(n-1)
```

1. Quel est le résultat renvoyé par `mystere(4)` ? 24

   > `mystere(4)` = 4 × `mystere(3)` = 4 × 3 × `mystere(2)` = 4 × 3 × 2 ×`mystere(1)` = 4 × 3 × 2 × 1 × `mystere(0)` = 4 × 3 × 2 × 1 × 1 = 24

2. Décrire, en français, ce que fait cette fonction :

   > Cette fonction calcule le produit de tous les entiers de 1 à n, c'est à dire la factorielle de n.

3. Proposer une version itérative documentée pour cette fonction :

```
def mystere(n : int) ->  int :
    '''
    Calcule le produit de tous les entiers de 1 à n,
    c'est à dire la factorielle de n.
    Précondition :
        n (int) : entier naturel
    Postcondition :
        n! (int)
    Exemple :
    >>> mystere(4)
    24
    '''
    factorielle = 1
    for i in range(n) :
        factorielle = factorielle * (i + 1)
        print(factorielle)
    return factorielle
```

Exercice 2 :

On considère la fonction`sigma(n)` définit par le code :

```
def sigma(n):
    resultat = 0
    for i in range(n+1) :
        resultat = resultat + i
    return resultat
```

1. Quel est le résultat renvoyé par `sigma(0)` ?

> 0

1. Quel est le résultat renvoyé par `sigma(4)` ?

> 10

1. Décrire, en français, ce que fait cette fonction :

   > Cette fonction calcule la somme de tous les entiers de 1 à n

2. Proposer une version récursive documentée pour cette fonction :

```
def sigma(n : int) -> int:
    '''
    Calcule la somme de tous les entiers de 1 à n.
    Précondition :
        n (int) : entier naturel
    Postcondition :
        somme de 1 à n (int)
    Exemple :
    >>> sigma(4)
    10
    '''
    if n==0 :
        return 0
    else :
        return n + sigma(n-1)
```