

## Programme de khôlle

### Semaine 7 (3 novembre 2025)

---

## Chapitre 3 : Raisonnements par récurrence - Sommes et produits

*Exercices réalisés : TD3, Exercices n°1 à 8*

- (★) Connaître et savoir utiliser le raisonnement par récurrence **simple**.

**Exercices n°1, 2, 3 et 4**

- (★) Soient  $n, p \in \mathbb{N}$ ,  $p \leq n$ ,  $q, \lambda \in \mathbb{R}$ . Connaître parfaitement la valeur de chacune des sommes usuelles suivantes :

$$\sum_{k=p}^n k = \frac{(n-p+1)(p+n)}{2} \quad ; \quad \sum_{k=p}^n \lambda = (n-p+1)\lambda \quad ; \quad \sum_{k=1}^n k^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{k=1}^n k^3 = \frac{n^2(n+1)^2}{4} \quad ; \quad \sum_{k=p}^n q^k = \begin{cases} q^p \frac{1-q^{n-p+1}}{1-q} & \text{si } q \neq 1 \\ n-p+1 & \text{si } q = 1 \end{cases}$$

- (★) Savoir reconnaître et calculer des sommes arithmétiques ou géométriques.
- (★) Connaître et savoir utiliser la linéarité, la relation de Chasles. **Exercices n°8**
- Lorsque la valeur d'une somme est donnée, savoir la justifier à l'aide d'un raisonnement par récurrence. **Exercices n°7**
- (★) Savoir reconnaître une somme télescopique et savoir déterminer sa valeur. **Exercices n°9**

- Connaître et savoir utiliser la notation produit. **Exercices n°11 et 12**
- Connaître et savoir utiliser la notation factorielle. **Exercices n°10**

## Python : Conditionnelle `if...else` et boucle `for`

*TP n°2 et 3*

- Soit  $f$  une fonction définie sur un ensemble  $D$  à valeurs dans  $\mathbb{R}$ . Savoir produire le script d'une fonction Python prenant en argument un nombre  $x$ , qui affiche un message d'erreur lorsque  $x \notin D$  et qui renvoie la valeur de  $f(x)$  sinon.

**On prendra soin de correctement placer les indentations.**

Par exemple, la fonction  $f : \mathbb{R}_+^* \rightarrow \mathbb{R}$ ,  $x \mapsto \ln(x)$  pourra être codée par le script :

```
import math as m

def f(x):

    if x<=0:

        print('La valeur', x, "n'appartient pas à D.f")

    else:

        return m.log(x)
```

- Soit  $(u_n)$  une suite réelle. Savoir produire le script d'une fonction Python prenant en argument un entier  $n$  et renvoyant la valeur du terme de rang  $n$  de la suite  $(u_n)$  dans le cas où la suite est définie par récurrence.

Par exemple, pour  $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N}, u_{n+1} = u_n^2 + 1 \end{cases}$  pourra être codée par le script :

```
def suite_u(n):

    u=1

    for k in range(n):

        u = u**2 + 1

    return u
```

- Soit  $(u_n)$  dont une expression explicite est connue. Savoir produire une fonction Python prenant en argument un entier  $n$  et qui renvoie la valeur de la somme

$$\sum_{k=p}^n u_k.$$

Par exemple, pour  $\sum_{k=2}^n k2^k$  pourra être codée par le script :

```
def somme_u(n):
    S=0
    for k in range(2,n+1):
        S += k*2**k
    return S
```

- Soit  $(v_n)$  une suite réelle. Savoir produire le script d'une fonction Python prenant en argument un entier  $n$  et renvoyant la valeur du terme de rang  $n$  de la suite  $(v_n)$  dans le cas où la suite est définie par récurrence d'ordre 2.

Par exemple, pour  $\begin{cases} v_0 = 0, v_1 = 1 \\ \forall n \in \mathbb{N}, v_{n+2} = v_{n+1} + v_n \end{cases}$  pourra être codée par le script :

```
def suite_v(n):
    (v0, v1) = (0, 1)
    for k in range(n):
        (v0, v1) = (v1, v0+v1)
    return v0
```

**Avec affectation simultanée**

ou également par le script

```
def suite_v(n):
    v0=0
    v1=1
    for k in range(n):
        z=v1
        v1=v0+v1
        v0=z
    return v0
```

**Avec variable auxiliaire**