Programme de khôlle

Semaine 8 (10 novembre 2025)

Chapitre 3 : Raisonnements par récurrence - Sommes et produits

Exercices réalisés : TD3, toute la feuille

- ($\star$) Connaître et savoir utiliser le raisonnement par récurrence **simple**. **Exercices n°1, 2, 3 et 4**
- ($\star$) Soient $n, p \in \mathbb{N}$, $p \leq n$, $q, \lambda \in \mathbb{R}$. Connaître parfaitement la valeur de chacune des sommes usuelles suivantes :

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

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

- ($\star$) Savoir reconnaître et calculer des sommes arithmétiques ou géométriques.
- ($\star$) 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**
- ($\star$) 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**

Chapitre 4 : Systèmes linéaires

Exercices réalisés : TD4, toute la feuille excepté l'exercice n°7

- ($\star$) Connaître la définition de système linéaire échelonné. Connaître les définitions de : pivots, inconnues principales, inconnues secondaires. Connaître les définitions de second membre d'un système linéaire et de système linéaire homogène. **Exercices n°3**
- Savoir utiliser la méthode de combinaison des lignes (pivot de Gauss) pour échelonner un système linéaire donné. **Exercices n°1, 2 et 5**
- Savoir repérer, une fois l'échelonnement effectué, si le système possède : une unique solution, une infinité de solutions ou aucune solution. Dans le cas où le système est compatible, savoir résoudre le système linéaire échelonné par « remontée »des lignes. **Exercices n°4**
- ($\star$) Connaître la définition de système de Cramer ainsi que sa caractérisation grâce au nombre de pivots non nuls (nombre d'inconnues principales).
- Savoir résoudre un système linéaire à second membre variable. **Exercices n°6**

Les points ($\star$) peuvent être l'objet d'une question de cours

Python : Conditionnelle `if...else` et boucles `for`, `while`

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 $f : \mathbb{R}_+^* \to \mathbb{R}, x \mapsto \ln(x)$ 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

- Soit (w_n) une suite réelle. Savoir produire le script d'une fonction Python prenant en argument un réel M et renvoyant le premier rang n (ou le premier terme de la suite) pour lequel $w_n > M$. On admet que cette condition est vérifiée pour un certain rang.

  Par exemple, pour $\begin{cases} w_0 = 0, \\ \forall n \in \mathbb{N},\, w_{n+1} = 2w_n + 1 \end{cases}$ pourra être codée par le script :

```
def seuil_w(M):
    w=0
    n=0
    while w <= M:
        w = 2*w + 1
        n = n + 1
    return n # ou w en fonction de ce qui est demandé
```