

Semaine n°18 du 24 février au 28 février

Informatique(Python) : cf exemples en annexe

- ⇒ boucle `while`, boucle `for`.
- ⇒ listes en Python : création d'une liste, extraction d'un élément, parcours d'une liste, concaténation, `len`, `append`...etc
- ⇒ Chaîne de caractères.
- ⇒ Dictionnaire.
- ⇒ Tri par sélection.

dénombrément

- ⇒ Cardinal d'un ensemble : définition, cardinal d'un sous-ensemble, lien avec les ensembles de départ et d'arrivée des applications injectives, surjectives, bijectives (cas particulier d'une application ayant un ensemble de départ et d'arrivée de même cardinal).
- ⇒ Cardinal d'une union : disjointe de deux ensembles, disjointe de n ensembles, union quelconque de deux ensembles.
- ⇒ Cardinal d'un produit cartésien.
- ⇒ p -liste sans répétition : définition, nombre de p -listes sans répétition d'un ensemble à n éléments.
- ⇒ permutations : définition, nombre de permutations d'un ensemble.
- ⇒ combinaisons : définition, nombre de p -combinaisons d'un ensemble à n éléments.
- ⇒ cardinal de l'ensemble des parties d'un ensemble E fini (démonstration exigible).

Espaces vectoriels

- ⇒ Espace vectoriel \mathbb{K}^n : vecteurs, scalaires, addition de deux vecteurs, multiplication d'un vecteur par un scalaire.
- ⇒ Propriétés : Soient $\lambda \in \mathbb{K}$ et $\vec{u} \in \mathbb{K}^n$. Alors
 - $0 \cdot \vec{u} = \vec{0}_{\mathbb{K}^n}$
 - $\lambda \cdot \vec{0}_{\mathbb{K}^n} = \vec{0}_{\mathbb{K}^n}$
 - $\lambda \cdot \vec{u} = \vec{0}_{\mathbb{K}^n} \iff (\lambda = 0 \text{ ou } \vec{u} = \vec{0}_{\mathbb{K}^n})$
- ⇒ Combinaison linéaire de vecteurs.
- ⇒ Sous espace vectoriel de \mathbb{K}^n : partie de \mathbb{K}^n contenant le vecteur nul et stable par combinaison linéaire.
- ⇒ Intersection de deux sous-espaces vectoriels (démonstration exigible)
- ⇒ Notation $\text{Vect}(\vec{u}_1, \dots, \vec{u}_p)$ où \vec{u}_1, \vec{u}_p sont des vecteurs de \mathbb{K}^n : ensemble des combinaisons linéaires des vecteurs $\vec{u}_1, \dots, \vec{u}_p$. C'est un sous espace vectoriel de \mathbb{K}^n appelé le sous-espace vectoriel de \mathbb{K}^n engendré par les vecteurs $\vec{u}_1, \dots, \vec{u}_p$.
- ⇒ Différentes écritures d'un sous-espace vectoriel de \mathbb{K}^n :
 - Ecriture cartésienne : $A = \{(x, y, z) \in \mathbb{R}^3, x + y + z = 0 \text{ et } x - y = 0\}$.
 - Ecriture paramétrée : $A = \{(x, x, -2x), x \in \mathbb{R}\}$.
 - Ecriture sous forme d'un sous espace vectoriel engendré par une famille de vecteurs : $A = \text{vect}((1, 1, -2))$.
- ⇒ Famille génératrice : définition,
- ⇒ Famille liée, libre : définitions, propriétés, caractérisation des familles libres :

$$(\vec{u}_1, \dots, \vec{u}_p) \text{ libre} \iff (\forall (\alpha_1, \dots, \alpha_p) \in \mathbb{K}^p, \alpha_1 \vec{u}_1 + \dots + \alpha_p \vec{u}_p = \vec{0}_{\mathbb{K}^n} \Rightarrow \alpha_1 = \dots = \alpha_p = 0)$$

- ⇒ Bases : définition, unicité de la décomposition d'un vecteur dans une base (démonstration exigible), dimension.

⇒ Liens entre famille génératrice, libre et base :

Soit E un sous-espace vectoriel de \mathbb{K}^n de dimension p .

- Toute famille libre de E a au plus p éléments.
- Toute famille génératrice de E a au moins p éléments.

Soit E un sous-espace vectoriel de \mathbb{K}^n de dimension p .

- Une famille libre ayant p éléments de E est une base de E .
- Une famille génératrice ayant p éléments de E est une base de E .
- De toute famille génératrice de E , on peut extraire une base de E .

⇒ propriétés des dimensions :

Si E et F sont deux sous espaces vectoriels de \mathbb{K}^n tels que $E \subset F$ et $\dim(E)=\dim(F)$ alors $E = F$.

⇒ Rang d'une famille de vecteur : définition $Rg(\vec{u}_1, \dots, \vec{u}_p) = \dim(\text{vect}(\vec{u}_1, \dots, \vec{u}_p))$, détermination pratique par extraction d'une base utilisation du rang pour prouver qu'une famille est libre, génératrice, ou base.

Remarques aux colleurs

- Merci aussi de poser une petite question d'informatique (cf Annexe).
- Le calcul du rang en utilisant le rang de la matrice de la famille de vecteurs n'a pas encore été vu en TD (un seul exercice d'application en cours).

Exemples de programmes informatiques

Exercice 1

Ecrire en Python une fonction `existence` qui prend en entrée une liste L et un nombre $element$ et renvoie `True` si $element$ se trouve dans la liste L , `False` sinon.

```
def existence(L,element):
    n=len(L) # taille de la liste
    for i in range(n):
        if L[i]==element:
            return True
    return False # si on n' a pas trouvé element après avoir parcouru toute la liste
```

Exercice 2

Ecrire en Python une fonction `MaximumListe` qui prend en entrée une liste L et renvoie la plus grande valeur de cette liste

```
def MaximumListe(L):
    n=len(L) #taille de la liste
    maxi=L[0] #on considère temporairement que le max est le premier élément
    for i in range(n):
        if L[i]>maxi:
            maxi=L[i] #on a trouvé une plus grande valeur
    return maxi
```

Exercice 3

Ecrire en Python une fonction `Somme` qui prend en entrée une liste L et renvoie la somme de ses éléments :

```
def Somme(L):
    n=len(L) #taille de la liste
    S=0 #initialisation de la somme
    for i in range(n):
        S=S+L[i]
    return S
```

Exercice 4

Ecrire une fonction `experience` qui prend en paramètre un entier n et simule n lancers successifs d'une pièce de monnaie équilibrée en renvoyant une liste aléatoire composée de n valeurs égales à 0 ou 1. On considérera que 0 correspond à Face et 1 à Pile.

```
from random import * # bibliothèque nécessaire pour créer des nombres aléatoires
def experience(n):
    L=[] #liste vide initialement
    for i in range(n):
        L.append(randint(0,1)) # 0 ou 1 choisi de manière aléatoire
    return L
```