Programme de colle 27 - Semaine du 19 mai 2025

Informatique (voir exemples en annexe)

- **Tableaux 1D et 2D** : commande `import numpy as np` et utilisation de `array`, `linspace`, `zeros`, `shape` etc et manipulations de base pour parcourir les éléments d'un tableau 2D, commande `M[i,j]`, etc..

Chapitre 20 : Statistiques univariées

- Définitions : Série statistiques, médiane, quartiles, écart interquartiles, moyenne (linéarité de la moyenne Démonstration exigible).
- Variance : définition, théorème de Koenig-huygens (Démonstration exigible), formule $V(ax_i + b) = a^2V(x_i)$ (Démonstration exigible)
- Ecart-type.

Chapitre 21 : Applications linéaires

- Applications linéaires
  - Définition d'une application linéaire, image du vecteur nul, vocabulaire (endomorphisme, isomorphisme, automorphisme)
  - Opérations : somme, multiplication par un scalaire, composition (démonstration exigible), réciproque d'applications linéaires.
- Noyau, Image
  - Noyau : définition, sous-espace vectoriel de l'ensemble de départ (démonstration exigible), caractérisation de l'injectivité par le noyau (démonstration exigible).
  - Image : définition, sous-espace vectoriel de l'ensemble d'arrivée, $Im(f) = vect(f(e_1), ...f(e_p))$ où $(e_1, ..., e_p)$ est une base de l'ensemble de départ, caractérisation de la surjectivité.
  - Détermination d'une application linéaire à partir de l'image d'une base, liens entre l'image d'une base et injection, surjection, bijection.
- Représentation matricielle
  - Matrice d'une application linéaire : définition, expression matricielle de l'image d'un vecteur.
  - Matrice d'une combinaison linéaire, d'une composée ou d'une réciproque d'applications linéaires.
  - Rang d'une application linéaire : définition, intérêt pratique pour déterminer si une application linéaire est injective, surjective, bijective. Théorème du rang.

Chapitre 22 : Variables aléatoires réelles finies

- Variable aléatoire
  - Vocabulaire : variable aléatoire réelle (VAR), univers image, évènements $(X = x)$, $(X \leq x)$ $(X \geq x)$, système complet associé à une VAR, exemple de la fonction indicatrice.
  - La somme des probabilités des évènements $(X = x_k)$ où les x_k sont les valeurs prises par X vaut 1.
- Outils pour étudier les variables aléatoires
  - Loi d'une variable aléatoire définition, représentation avec un tableau, représentation avec un diagramme bâton.

- Fonction de répartition : définition, représentation graphique. La fonction de répartition est croissante (pas d'autre propriété).
- Méthode pour obtenir la loi à partir de la fonction de répartition et vice-versa.

→ Loi usuelles : loi certaine, loi uniforme, loi de Bernouilli, loi Binomiale (expérience type, univers image, loi de probabilité).

→ Indépendance :

- Définition de l'indépendance, de la mutuelle indépendance. Méthode pour déterminer si deux VAR sont indépendantes ou non.
- Une somme de loi de Bernouilli mutuellement indépendantes de même paramètre suit une loi Binomiale

→ Espérance

- Définition de l'espérance et d'une variable centrée. Propriété de linéarité, positivité, croissance.
- Théorème de transfert.

Remarques aux colleurs

1. Merci de commencer par poser une question de cours, puis une question d'informatique.
2. Les formules d'espérance des lois usuelles n'ont pas encore été vue.
3. Pas de décalage entre les programmes de colles des deux classes cette semaine (Semaine 27).

Annexe 1 : Exemples

Exemples de question d'informatique

1. Ecrire en Python une fonction `existence` qui prend en entrée un tableau T et un nombre *element* et renvoie `True` si *element* se trouve dans le tableau T, `False` sinon.

```
def existence(T,element):
    a=len(T)                    # nombre de lignes
    b=len(T[0])                 # nombre de colonnes
    for i in range(a):          # parcours des lignes
        for j in range(b):      # parcours des colonnes
            if T[i,j]==element:  # on teste si T[i,j] est égal à élément
                return True
    return False  # si on n' a pas trouvé element après avoir parcouru tout le tableau
```

2. Ecrire en Python une fonction `Moyenne` qui prend en entrée un tableau T et renvoie la moyenne de ses éléments :

```
def Moyenne(T):
    n,p = T.shape           # n nombre de lignes et p nombre de colonnes
    S=0                     # initialisation de la somme
    for i in range(n):      # parcours des lignes
        for j in range(p):  # parcours des colonnes
            S=S+T[i,j]      # on rajoute l'élément T[i,j]
    return S/(n*p)          # formule pour la moyenne
```

3. Créer une fonction `SommeLignes` prenant en entrée un tableau T et qui renvoie une liste contenant la somme de chaque lignes.

```
def SommeLignes(T):
    n=len(T)                  # nombre de lignes
    p=len(T[0])               # nombre de colonnes
    L=[]                      # la liste est vide au debut
    for i in range(n):        # parcours des lignes
        S=0                   # initialisation pour la somme de la ligne i
        for j in range(p):    # parcours des colonnes
            S = S + T[i,j]    # on rajoute l'élément T[i,j]
        L.append(S)           # on met la somme de la ligne i dans la liste L
    return L
```