# Semaine n°24 du 06 au 10 avril

## 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 caractère.
- Tri par selection, tri par comptage

## Matrices

- Définitions : matrice nulle, carrée, identité, ligne, colonne, diagonale, triangulaire supérieure ou inférieure.
- Opérations : additions de deux matrices, multilpication par un scalaire, produit de matrices.
- Propriétés du produit : produit avec la matrice identité ou la matrice nulle, associativité, distributivité, non commutativité, non intégrité, $AB = AC \not\Rightarrow B = C$.
- Ecriture matricielle d'un système linéaire.
- Rang d'une matrice (= rang de son système associé), méthode de calcul du rang en échelonnant la matrice.
- Puissances de matrices carrées, cas particulier des matrices diagonales, polynômes de matrices.
- Binôme de Newton quand les matrices sont commutatives.
- Transposée d'une matrice : définition, propriétés, matrices symétriques et antisymétriques.
- Matrice carrée inversible : définition, propriétés, puissances avec exposant négatif.
- Recherche pratique de l'inverse : $A$ est inversible si et seulement si son système associé $AX = B$ est un système de Cramer, détermination de l'inverse en résolvant le système en question.
- Critère d'inversibilité avec le rang.
- Recherche de l'inverse à l'aide d'un polynôme annulateur.
- critère d'inversabilité pour les matrices de taille 2 : déterminant. ( Pas de formule donnant directement l'inverse.)

## Statistiques univariée

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

## Probabilités : révision

- système complet d'évènements
- Formules des probabilités composées (simple et généralisée).
- Formule des probabilités totales

## Variables aléatoires réelles

- Définitions :variable aléatoire réelle, univers images, système complet associé à une VAR.

- Loi de probabilité et représentation graphique, fonction de répartition et représentation graphique, croissance de la fonction de répartition, méthode de construction de la fonction de répartition à partir de la loi de probabilité et inversement.
- lois usuelles : certaine, uniforme, Bernoulli, binomiale expérience type, univers image, loi de probabilité).
- Variables aléatoires indépendantes
- Espérance : définition, linéarité, variable centrée, théorème de transfert, positivité et croissance de l'espérance.
- Espérance des lois usuelles (démonstration exigible pour la loi uniforme)

Remarques aux colleurs

— Merci aussi de poser une petite question d'informatique

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
```