

Semaine n°14 du 13 au 17 janvier

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
- Graphique : module `pyplot` de `matplotlib`, fonction `plot` et `show`. (Attention la fonction `linespace` (de `numpy`) n'est pas encore connue). On rappellera l'importation `import matplotlib.pyplot as plt`

Équations différentielles linéaires

- Définitions : équations différentielles linéaires du premier ordre, équations différentielles linéaires du second ordre à coefficients constants et second membre non constant.
- Théorème fondamental : Soit E une équation différentielle linéaire d'ensemble de solutions \mathcal{S} et E_h son équation homogène associée d'ensemble de solutions \mathcal{S}_h et f_0 une solution particulière de E , alors :

$$\mathcal{S} = \{g + f_0, g \in \mathcal{S}_h\}$$

- Résolution des équations différentielles homogène linéaires du premier ordre.
- Recherche d'une solution particulière des équations différentielles linéaires du premier ordre : solution évidente, méthode de la variation de la constante,
- Exemples et résolution d'un problème de Cauchy du premier ordre.
- Résolution d'équations différentielles du 2nd ordre : solutions de l'équation homogène, recherche d'une solution particulière (la forme de la solution particulière sera donnée par l'examinateur) , exemples et résolution d'un problème de Cauchy du second ordre.

Suites usuelles

- Variation d'une suite, suite majorée, minorée, bornée.
- suite arithmétique : définition, terme général en fonction de n , somme des termes d'une suite arithmétique.
- suite géométrique : définition, terme général en fonction de n , somme des termes d'une suite géométrique.
- suite arithmético-géométrique : définition, méthode pour déterminer le terme général en fonction de n .
- suite récurrente linéaire d'ordre 2 : définition, équation caractéristique, détermination du terme général en fonction de n .
- Théorème sur les limites d'une suite : théorème de la limite monotone, théorème des gendarmes, théorème de comparaison.

Applications

- Définitions : application, image, antécédent.
- Application identité, application nulle, fonction indicatrice.
- Image d'une partie de l'ensemble de départ pour une application $f : E \rightarrow F$:
 $f(A) = \{f(x) | x \in A\}$
- Surjection : définition, méthode pour montrer qu'une application est surjective (en résolvant l'équation $f(x) = y$) ou qu'elle n'est pas surjective (en exhibant un élément qui n'a pas d'antécédent).
- Injection : définition, méthode pour montrer qu'une application est injective ($(\forall(a, b) \in E^2, f(a) = f(b) \Rightarrow a = b)$) ou qu'elle n'est pas injective (en exhibant deux éléments ayant la même image).
- bijection : définition,

Remarques aux colleurs

— Merci aussi de poser une petite question d'informatique (cf Annexe).

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
```