Semaine n°27 du 27 au 31 mai

Informatique(Python) : cf exemples en annexe

- Tableau 2D, bibliothèque `Numpy`
- Simulation expérience aléatoire. Estimation des probabilités et de l'espérance.

Limites de fonctions

- Définitions avec les quantificateurs des limites suivantes :
  $\lim\limits_{x\to x_0} f(x)=\ell$, $\lim\limits_{x\to x_0} f(x)=+\infty$, $\lim\limits_{x\to x_0} f(x)=-\infty$, $\lim\limits_{x\to+\infty} f(x)=\ell$, $\lim\limits_{x\to+\infty} f(x)=+\infty$, ...
- Limite à gauche et limite à droite en un point, définition avec les quantificateurs.
- Asymptote verticale et horizontale.
- Opérations sur les limites : addition, multiplication, quotient, composition, formes indéterminées.
- Limites et inégalités : Si $\lim\limits_{x\to x_0} f(x)=l>0$ alors il existe un voisinage de x_0 sur lequel f est strictement positive, théorème des gendarmes, théorèmes de comparaison.
- Une fonction monotone sur un intervalle ouvert admet une limite en chacune des bornes de l'intervalle.
- Croissances comparées (logarithme, puissance,exponentielle)
- fonctions équivalentes : définition, équivalents usuels :
  - polynômes en $+\infty$, $-\infty$ et 0.
  - $\sin(x) \underset{x\to0}{\sim} x$
  - $\tan(x) \underset{x\to0}{\sim} x$
  - $\ln(1+x) \underset{x\to0}{\sim} x$
  - $e^x-1 \underset{x\to0}{\sim} x$
  - pour tout $\alpha\in\mathbb{R}^*$, $(1+x)^\alpha-1 \underset{x\to0}{\sim} \alpha x$ et en particulier (pour $\alpha=\frac{1}{2}$), $\sqrt{1+x}-1 \underset{x\to0}{\sim} \frac{x}{2}$
  - $1-\cos(x) \underset{x\to0}{\sim} \frac{x^2}{2}$
- Propriétés : transitivité, produit, quotient, puissance (pas de somme ni de composition !), théorème de substitution.

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 : définition, sous-espace vectoriel de l'ensemble de départ (démonstration exigible), caractérisation de l'injectivité,
- Image : définition, sous-espace vectoriel de l'ensemble d'arrivée, $Im(f) = vect(f(\vec{e_1}), ...f(\vec{e_p}))$ où $(\vec{e}_1), ...\vec{e}_p))$ 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.
- 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.

Continuité

- Définition de la continuité en un point, continuité à gauche et à droite.
- Continuité sur un intervalle, continuité des fonctions usuelles, opérations algébriques sur les fonctions continues, composition de fonctions continues.
- Prolongement par continuité.

Remarques aux colleurs

— Merci aussi de poser une petite question d'informatique (cf Annexe).
— Pouvez vous vérifier la rédaction de l'étude des intervalles de continuité ?

Exemples de programmes informatiques

Exercice 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
```

Exercice 2

Ecrire en Python une fonction `MaximumTableau` qui prend en entrée un tableau T et renvoie la plus grande valeur de ce tableau

```
def MaximumTableau(T):
    a=len(T)       # nombre de lignes
    b=len(T[0])    # nombre de colonnes
    maxi = T[0,0]  # initialisation avec la première valeur du tableau
    for i in range(a):       # parcours des lignes
        for j in range(b):   # parcours des colonnes
            if T[i,j]>maxi:  # on teste si T[i,j] est plus grand
                maxi=T[i,j]    # on a trouvé une plus grande valeur
    return maxi
```

Exercice 3

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

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

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):         #on répète n fois
        L.append(randint(0,1)) # on rajoute 0 ou 1, choisi de manière aléatoire
    return L
```

Exercice 5

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]
        L.append(S)         # on met la somme de la ligne i dans le liste L
    return L
```