Exercices improvisés 2025

BCPST 2
2022/2023

Sujet 1
Type Agro

Corrigé de l'exercice improvisé !

1. (a) Écrire une fonction qui prenant une liste de réels en argument permet de renvoyer la somme de ses éléments.

On propose le programme suivant :

```
def sommeliste(L):
    s=0
    for k in L:
        s=s+k
    return s
```

(b) Écrire une fonction qui prenant une liste de réels en argument permet de renvoyer son minimum.

On propose le programme suivant :

```
def minliste(L):
    m=L[0]
    for k in L:
        if k<m:
            m=k
    return m
```

(c) Écrire une fonction qui prenant une liste de réels en argument permet de renvoyer son minimum et le plus petit indice de la cellule où ce minimum est atteint.

On propose le programme suivant :

```
def minliste(L):
    m,i=L[0],0
    for k in range(len(L)):
        if L[k]<m:
            m,i=L[k], k
    return m, i
```

(d) Que faut-il changer pour obtenir le plus grand indice de la cellule où ce minimum est atteint ?

Voici notre petit changement, $<$ devient $\leqslant$:

```
def minliste(L):
    m,i=L[0],0
    for k in range(len(L)):
        if L[k]<=m:
            m,i=L[k], k
    return m, i
```

2. On dira qu'une cellule d'une liste contient un minimum local si ses deux voisines contiennent des valeurs supérieures ou égales (pas de minimum local aux extrémités). Déterminer une fonction qui renvoie les positions des minima locaux d'une liste.

On propose le programme suivant :

```
def posminlocaux(L):
    l=[]
    for k in range(1,len(L)-1):
        if L[k-1]>=L[k] and L[k+1]>=L[k]:
            l.append(k)
    return l
```

BCPST 2
2022/2023

Sujet 2
Type Agro

Corrigé de l'exercice improvisé !

Soit une matrice A carrée de taille n avec n un entier supérieur à 3, dont les coefficients sont des réels $a_{i,j}$. On la représente sous forme de tableau `array` dans la bibliothèque `numpy` (Dans ce sujet, on n'utilisera que la bibliothèque numpy).

1. On appelle Tr(A) l'application qui calcule la somme des termes diagonaux de A. Écrire une fonction qui prend en argument A et renvoie Tr(A).
   On propose le programme suivant :

```
def Trace(A):
    s=0
    n=size(A, 1)
    for j in range (n):
        s=s+A[j, j]
    return s
```

2. On définit l'anti-diagonale de A comme étant celle qui part du coin en bas à gauche pour remonter vers le coin en haut à droite. Écrire une fonction qui calcule la somme des coefficients anti-diagonaux.
   On propose le programme suivant :

```
def Antitrace(A):
    s=0
    n=size(A, 1)
    for j in range (n):
        s=s+A[n-1-j, j]
    return s
```

3. Écrire une fonction qui renvoie le booléen T*rue* si la somme des éléments diagonaux de A est supérieure à la somme des éléments anti-diagonaux, F*alse* sinon.
   On propose le programme suivant :

```
def compare(A):
    return Trace(A)>=Antitrace(A)
```

4. Écrire une fonction qui renvoie la somme des éléments de A situés strictement au-dessus de la diagonale.
   On propose le programme suivant :

```
def audessus(A):
    n=size(A, 1)
    s=0
    for i in range(n):
        for j in range(i+1,n):
            s+=A[i][j]
    return s
```

BCPST 2
2023/2024

Sujet 3
Type Agro

Corrigé de l'exercice improvisé !

Soit $(u_n)_{n\in\mathbb{N}}$, la suite définie par : $u_0 = 10$ et $\forall n \in \mathbb{N}, u_{n+1} = f(u_n)$ avec : $f : x \mapsto x - \ln(x)$.

1. Écrire une fonction `Lsuite` prenant en entrée un entier naturel n et renvoyant la liste $[u_0, u_1, \cdots, u_n]$.
   Réponse : On propose le programme suivant :

```
import numpy as np
def f(x):
    return x - np.log(x)

def graph2(n):
    L=[]
    u=10
    for k in range(n+1):
        L.append(u)
        u=f(u)
    return L

```

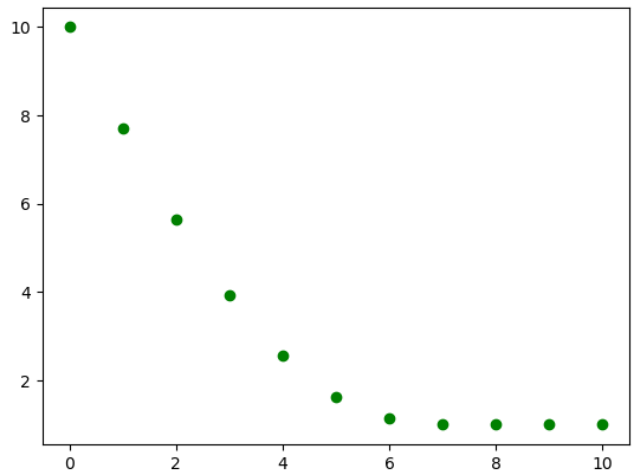
2. En déduire une fonction renvoyant un graphique de $(u_n)_{n\in\mathbb{N}}$.
   Réponse : On propose le programme suivant :

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return x - np.log(x)

def graph(n):
    Abs=[k for k in range(n+1)]
    Ord=[]
    u=10
    for k in range(n+1):
        Ord.append(u)
        u=f(u)
    plt.plot(Abs,Ord, 'o')
    plt.show()

```

On a obtenu :

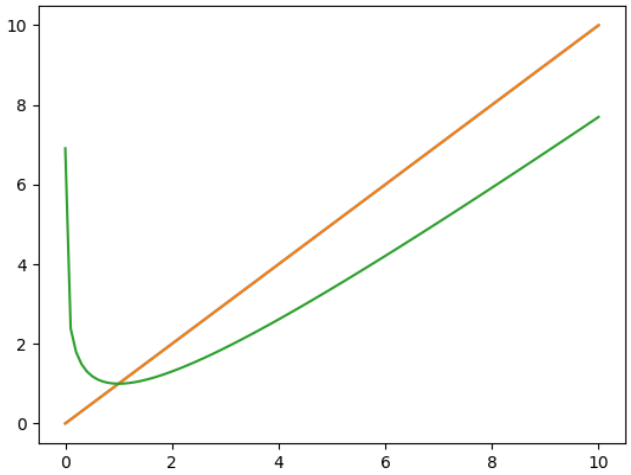


3. Proposer une fonction renvoyant un graphique de f sur $[e^{-3},10]$ ainsi que de la fonction $x \mapsto x$.
Réponse : On propose le programme suivant :

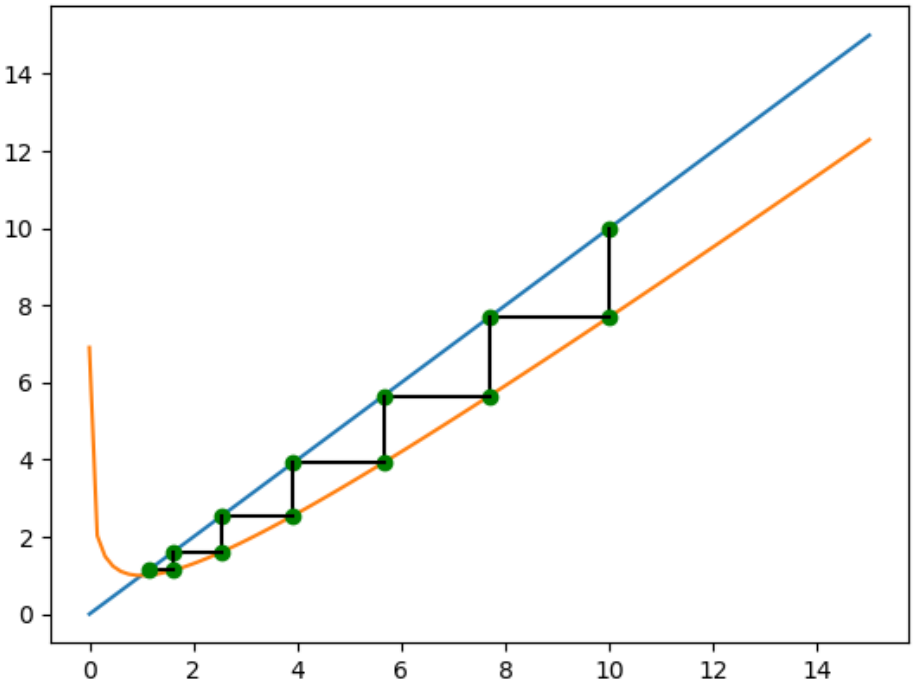
```
def tracef():
    xb = 1e-3
    xe = 10
    x = np.linspace(xb, xe, 100)
    plt.plot(x, x)
    y=[f(a) for a in x]
    plt.plot(x, y)
    plt.show()

```

On a obtenu :



4. On veut obtenir ce graphique :



Expliquer ce graphique, conjecturer la limite de $(u_n)_{n\in\mathbb{N}}$ et donner une fonction permettant d'obtenir ce graphique.

Réponse :

- On représente les courbes de f et de Id $: x \mapsto x$ ainsi que le point (u_0, u_0).
- On obtient ensuite le point $(u_0, f(u_0))$, i.e. (u_0, u_1) grâce à la courbe de f (il suffit de tracer la droite verticale d'équation $x = u_0$ puis de prendre l'intersection avec la courbe de f).
- On obtient ensuite le point (u_1, u_1) grâce à la courbe de Id (il suffit de prendre le point précédent et de tracer la droite horizontale d'équation $y = u_1$ puis de prendre l'intersection avec la courbe de Id).
- On obtient ensuite le point $(u_1, f(u_1))$, i.e. (u_1, u_2) grâce à la courbe de f (il suffit de prendre le point précédent et de tracer la droite verticale d'équation $x = u_1$ puis de prendre l'intersection avec la courbe de f).
- et on poursuit ce processus !

Sur ce graphique, la limite de $(u_n)_{n\in\mathbb{N}}$ semble être 1 qui est l'abscisse du point d'intersection des courbes de f et de Id . Voici une fonction permettant d'obtenir ce graphique.

```
def trace(N):
    u0=10
    xb = 1e-3
    xe = 15
    x = np.linspace(xb, xe, 100)
    plt.plot(x, x)
    y=[f(a) for a in x]
    plt.plot(x, y)
    plt.plot([u0], [u0], 'og')
    for k in range(N):
        u1 = f(u0)
        plt.plot([u0, u0], [u0, u1], 'k')
        plt.plot([u0], [u1], 'og')
        plt.plot([u0, u1], [u1, u1], 'k')
        plt.plot([u1], [u1], 'og')
        u0 = u1
    plt.show()

```

BCPST 2
2023/2024

Sujet 4
Type Agro

Corrigé de l'exercice improvisé !

1. Écrire une fonction `somme_cumul` qui prend en entrée une liste L d'entiers et qui renvoie une liste M des sommes cumulées, c'est à dire une liste M, de même longueur que L, telle que $\mathrm{M}[i] = \sum_{k=0}^{i} \mathrm{L}[k]$ pour tout indice i de L.

   Réponse : On propose le programme suivant :

```
def somme_cumul(L):
    M=len(L)*[0]
    M[0]=L[0]
    for k in range (1,len(L)):
        M[k]=L[k]+M[k-1]
    return M
```

2. On propose à deux joueurs A et F la résolution d'une suite infinie de problèmes numérotés $0, 1, 2, \cdots$. Ils débutent la résolution à l'instant 0 du problème numéro 0. Dès que l'un des joueurs termine la résolution du problème k, il passe au problème $k+1$. Le numéro d'un problème est attribué au joueur qui le résout en premier. En cas de simultanéité de la résolution d'un problème par les deux joueurs, celui-ci n'est pas attribué. On souhaite écrire une fonction Python `repartition(dureesA,dureesF)` qui, étant donnés les deux listes des durées de résolution des n premiers problèmes par les deux joueurs, renvoie les listes des numéros des problèmes attribués à A et F pour les n premiers problèmes. Ainsi, `repartition([7,1,2,1,2],[4,2,4,3,1])` renvoie `([3,4],[0,1])`.

   (a) Quel couple renvoie l'exécution de `repartition([3,2,5,1,1,1],[4,2,2,4,4,2])` ?

   Réponse : Les temps cumulés des deux joueurs sont respectivement $[3, 5, 10, 11, 12, 13]$ et $[4, 6, 8, 12, 16, 18]$. On en déduit que le problème 0 est attribué à A, le problème 1 à A, le problème 2 à B... `repartition([3,2,5,1,1,1],[4,2,2,4,4,2])` va donc renvoyer $([0, 1, 3, 4, 5], [2])$.

   (b) Écrire la fonction recherchée.

   Réponse : On propose le programme suivant :

```
def repartition(L1,L2):
    M1=somme_cumul(L1)
    M2=somme_cumul(L2)
    R1=[]
    R2=[]
    for k in range (len(L1)):
        if M1[k]<M2[k]:
            R1.append(k)
        if M2[k]<M1[k]:
            R2.append(k)
    return (R1,R2)
```