I - Calcul matriciel avec Python

I.1 - Création de matrices

Python

On utilise la bibliothèque "numpy".

```
import numpy as np
```

- Création à la main : "np.array".

```
L = np.array([1, 2, 3])                 # vecteur ligne
C = np.array([[1], [2], [3]])           # vecteur colonne
M = np.array([[1, 2, 3], [4, 5, 6]])    # matrice 2 ligne 3 colonnes
```

- Matrices préremplies avec de zéros ("np.zeros") ou avec des uns ("np.ones") ou matrice identité ("np.eye").

```
np.zeros(3)        # [0,0,0]
np.zeros((2,3))    # [[0,0,0],[0,0,0]]
np.ones(3)         # [1,1,1]
np.ones((2,2))     # [[1,1],[1,1]]
np.eye(3)          # matrice identité d'ordre 3
```

- Tableau de nombres entre deux bornes : "np.arrange" et "np.linspace".

```
np.arrange(2,6)         # entiers entre 2 inclus et 6 exclu [2,3,4,5]
np.linspace(0, 2, 4)    # 4 valeurs équiréparties entre 0 et 2 inclus
```

- Manipulations.
  - ⋆ Taille d'une matrice.

```
A = np.array([[1,-2,3],[0,4,5]])
np.shape(A)  #  renvoie (2, 3), la taille de la matrice
```

  - ⋆ Transposée d'une matrice : "np.transpose(A)".

```
A = np.array([[1,2],[0,-1]])
np.transpose(A)  #  renvoie [[1,0],[2,-1]] : la transposée de A
```

  - ⋆ Produit matriciel : "np.dot(A,B)".

```
A, B = np.array([[1,2],[0,-1]]),  np.array([[0,2],[1,1]])
np.dot(A,B)           # renvoie  le produit matriciel A.B
```

  - ⋆ Extraction d'un élément, d'une ligne ou d'une colonne.

```
A = np.array([[1,-2,3],[0,4,5],[-1,2,0]])
A[1,2]  # renvoie 5 : l'élément de la ligne 2 et colonne 3
A[0,:]  # renvoie [1,-2,3] : toute la ligne 1
A[:,1]  # renvoie [-2,4,2] : toute la colonne 2
```

Exercice 1

1. Construire la matrice $A = \begin{pmatrix} -2 & 1 & 1 \\ 1 & -2 & 1 \\ 1 & 1 & -2 \end{pmatrix}$ à l'aide des commandes "np.ones" et "np.eye".
2. Quelle matrice obtient-on après l'exécution des commandes suivantes, A étant la matrice de la question 1 ?

```
A[:,1] = np.zeros(3)
A[0,0] = 3
```


I.2 - Calculs sur les matrices

Python

On utilise la bibliothèque "numpy.linalg".

```
import numpy.linalg as al
```

- Inverse d'une matrice : "al.inv".

```
A = np.array([[1,0,1],[1,-1,1],[1,1,0]])
al.inv(A)       # renvoie l'inverse de A
```

- Rang d'une matrice : "al.matrix_rank".

```
al.matrix_rank(A)       # renvoie le rang de A
```

- Puissance d'une matrice : "al.matrix_power".

```
al.matrix_power(A,3)       # renvoie la matrice A^3
```

- Valeurs propres et vecteurs propres d'une matrice : "al.eig".

```
D, V = al.eig(A) # renvoi deux éléments
# D est un vecteur ligne contenant les valeurs propres de A
# V est une matrice dont les colonnes sont les vecteurs propres de A
```

Exercice 2

Soit B la matrice définie par : $B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$ et on considère les instructions Python suivantes et le résultat associé.

```
>>> B = np.array([[0,1,0],[1,0,0],[0,0,1]])
>>> P = np.array([[1,1,0],[1,-1,0],[0,0,1]])
>>> np.dot( np.dot(al.inv(P),B) , P )
array([[ 1.,  0.,  0.],
       [ 0., -1.,  0.],
       [ 0.,  0.,  1.]])
```

1. Déduire de la séquence Python précédente que la matrice B est diagonalisable puis donner les valeurs propres de B ainsi que les vecteurs propres associés à ces valeurs propres.

On considère à présent les instructions Python suivantes :

```
>>> D, V = al.eig(B)
>>> D
array([ 1., -1.,  1.])
>>> V
array([[ 0.70710678, -0.70710678,  0.        ],
       [ 0.70710678,  0.70710678,  0.        ],
       [ 0.        ,  0.        ,  1.        ]])
```

2. Cette sortie Python est-elle cohérente avec vos résultats de la question 1 ?
3. Au vu des résultats précédentes, que renvoient les instructions suivantes ?

```
>>> al.matrix_rank(B-np.eye(3,3))
>>> al.matrix_rank(B+np.eye(3,3))
>>> al.matrix_rank(B+4*np.eye(3,3))
```

II - Exercices

Exercice 3

1. Ecrire une fonction Python `SommeCarrés(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie le réel $\sum_{1\leqslant i,j\leqslant n} a_{i,j}^2$.
2. Ecrire une fonction Python `SommeLignes(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie une liste contenant la somme des termes de chaque ligne de A.
3. On définit la trace d'une matrice A de $\mathcal{M}_n(\mathbb{R})$ comme la somme des ses éléments diagonaux. On note alors $tr(A)$ la trace de A : $tr(A) = \sum_{i=1}^{n} a_{i,i}$.

   Ecrire une fonction Python `Trace(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie sa trace.
4. On suppose que A est diagonalisable et on note $\lambda_1, \ldots, \lambda_n$ les valeurs propres de A. On définit alors l'énergie de A, notée $\mathcal{E}(A)$, comme la somme des valeurs abolues des valeurs propres de A : $\mathcal{E}(A) = \sum_{k=1}^{n} |\lambda_k|$.

   La fonction `al.eigvals(A)`, appliquée à un tableau `numpy` représentant une matrice diagonalisable, renvoie un tableau `numpy` contenant les valeurs propres A.
   Écrire une fonction PYTHON `Energie(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie l'énergie de A.

Exercice 4

1. Ecrire une fonction Python `minimax(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie le minimum de la liste contenant les maximums de chaque ligne de A.
2. Ecrire une fonction Python `maximin(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie le maximum de la liste contenant les minimums de chaque colone de A.
3. Ecrire une fonction Python `Test(A)` qui prend en paramètre une matrice A représentée par un tableau numpy `A` et qui renvoie 1 si son minimax est égal à son maximin et 0 sinon.