TP(3) Prise en main de Python

Spyder est un environnement de développement pour Python.
L'interpréteur de commandes IPython permet d'exécuter et tester des commandes.
Dans l'écriture d'un script, on utilise les sauts à la ligne et **l'indentation** (4 espaces)
→ pas besoin d'autres symboles délimiteurs de blocs.
Bibliothèques utilisées : **numpy** pour le calcul scientifique, **matplotlib** pour la visualisation.

les types de séquences :
type **liste** (défini avec [ ] , hétérogène, mutable),
type **tuple** (défini avec ( ) immuable),
type **array** du package numpy : une liste dont tous les éléments sont de même type.
Comme pour les listes, les indices des éléments d'un tableau commencent à zéro.

Tester les commandes qui suivent dans l'interpréteur IPython :

- importer le package numpy : import numpy as np

Contient des fonctions de calcul et surtout des fonctions qui permettent de manipuler des tableaux (array) → correspondent à des matrices lorsque leur dimension vaut 2

> A = np.array([[1, 2], [3, 4]]) > type(A) >np.ndim(A) > np.shape(A)

exemple de slicing (récupérer une sous séquence) : > A[0, 0] > A[0, :] > A[:, 1]

redimentionner un tableau :
> B = np.array([1, 2, 1]) > np.ndim(B)
> C = B.reshape(3, 1) > np.ndim(C) > print(C)
concaténation de tableaux :
> D = np.array([[1, 1], [2, 2]])
> E = np.concatenate((A,D), axis=1) # ou axis=0

création d'une copie de A : A2=A.copy() # create a deepcopy of A
> A1 = A > A2 = A.copy() > id(A) > id(A1) > id(A2)

opérations matricielles (avec des array de dim=2) :
np.dot(A, B) # produit matriciel
np.transpose(A) ou A.T # matrice transposée

matrices particulières :
> np.ones((3, 2)) > np.zeros((2, 2)) > np.eye(3) > np.diag([1, 2, 3])
> np.random.rand(2, 3)

générations de listes particulières :
> np.arange(7) > np.linspace(1, 2, 5) # linspace(début,fin,nombre)
> m = np.arange(3, 10) > type(m)

> x = np.linspace(-np.pi/2, np.pi, 4)
> np.sin(x) # action d'une fonction mathématique sur un tableau

- Pour vos visualisations import matplotlib.pyplot as plt et tester :

> x = np.array([1, 3, 4, 6])
> y = np.array([2, 3, 5, 1])
> plt.plot(x, y, 'b :')

Exercice 1

Dans l'interpréteur de commandes IPython, définir les matrices $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 5 \\ 2 & 3 \end{pmatrix}$
Calculer $A \times B$ et $B \times A$ que remarquez vous ?

Exercice 2

Effectuer des produits matriciels entre deux matrices choisies parmi les suivantes :
(Effectuer ces produits sur papier et vérifier avec Python)

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 3 & 1 & 1 \\ 0 & 1 & 2 \end{pmatrix} \quad B = \begin{pmatrix} 2 & 1 & 0 \\ 3 & 1 & 1 \end{pmatrix} \quad C = \begin{pmatrix} 2 & 1 & 0 \end{pmatrix} \quad D = \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \quad E = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$$

Exercice 3

1. Tracer la fonction $\exp(x)$ dans l'intervalle $[-1.5, 3]$
2. Tracer la fonction $\cos(t)$ pour $t \in [0, 4\pi]$

Exercice 4

Recopier dans votre script les lignes de code suivantes :

```
plt.close()
plt.axis('scaled') # même échelle
xmin, xmax = -1, 2
ymin, ymax = -1, 2
plt.axis([xmin,xmax,ymin,ymax])
plt.plot([xmin,xmax],[0,0],'k-', linewidth=1)
plt.plot([0,0],[ymin,ymax],'k-', linewidth=0.7)
plt.grid(True)

A = np.array([[0, 1, 1, 0, 0],
              [0, 0, 1, 1, 0]])
x = A[0, :]
y = A[1, :]
plt.plot(x, y, 'r-')
```

Appliquer aux points dont les coordonnées sont écrites dans les colonnes de la matrice A, la transformation matricielle décrite par la matrice $B = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$
et compléter votre script afin d'afficher les points transformés.

Exercice 5

Soit A une matrice. On note A^t sa **matrice transposée**.
A^t est la matrice dont chaque colonne j contient les éléments de la ligne j de la matrice A.

Calculer si cela est possible les produits matriciels suivants : $A^t\,A$, $A\,A^t$, $V^t\,V$, $V^t\,W$

$$A = \begin{pmatrix} 1 & 1 \\ 2 & 4 \\ 3 & 1 \end{pmatrix} \quad V = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad W = \begin{pmatrix} -3 \\ 0 \\ 1 \end{pmatrix}$$