Réflexe :
Importer la librairie nécessaire au calcul scientifique.

Entrée[1]:
```
1
```

TP Python 14 : Calcul matriciel

Partie I - Calcul matriciel : le module `linalg`

Dans le TP précédent, nous avons vu comment définir des matrices de toutes tailles en tant qu'*array*. Nous savons obtenir certaines informations de base sur les matrices, et savons effectuer un produit matriciel.

Il est désormais temps d'**utiliser** ces matrices...

Le module `linalg` (pour *linear algebra*) de la bibliothèque `numpy` contient un grand nombre d'algorithmes classiques d'algèbre linéaires. En ECG, on l'importera sous l'alias `al`, pour "*algèbre linéaire*". Cet acronyme étant français, il ne faut donc pas être perturbée si vous en rencontrez un différent en consultant des sites anglophones!

Réflexe :
Importer le module `linalg` de la bibliothèque `numpy` avec l'alias `al`.

Entrée[2]:
```
1
```

Les algorithmes (ce sont des *fonctions* Python) prêts-à-l'emploi du module `linalg` que nous allons utiliser sont les suivants :

Fonctions	Usage
al.inv(M)	Pour inverser une matrice $\boldsymbol{M}$.
al.matrix_rank(M)	Pour obtenir le rang d'une matrice $\boldsymbol{M}$.
al.det(M)	Pour obtenir le déterminant d'une matrice $\boldsymbol{M}$ (en ECG, seulement pour les matrices 2×2.)
al.matrix_power(M,n)	Pour mettre une matrice **carrée** $\boldsymbol{M}$ à la puissance $n \in \mathbb{N}$.

al.solve(A,B)	Détermine l'exacte solution X d'un système matriciel $AX = B$ où A est une matrice **carrée** et B un **vecteur**.

Partie II - Exercices

Exercice 1 - Déterminant

Soit $M = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \in \mathcal{M}_2(\mathbb{R})$.

1. En vous rappelant de la formule du déterminant de M, implémenter une fonction `determinant` qui prend en argument une matrice carrée M d'ordre 2 et qui renvoie son déterminant.

   On rappelle que le coefficient en i-ème ligne et j-ème colonne de la matrice est accessible avec la commande `M[i-1,j-1]` .
2. *Tester votre fonction pour* $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ *et* $C = \begin{pmatrix} 3 & -1 \\ 0 & 11 \end{pmatrix}$.
3. *Vérifier la robustesse de votre fonction en calculant* $\det(A)$, $\det(B)$ *et* $\det(C)$ *à l'aide de la fonction* `al.det` *. Que remarquez-vous?*

Entrée[ ]:
```
1
```

Exercice 2 - Inverse de matrice $2x2$

1. En vous rappelant de la formule permettant d'obtenir l'inverse d'une matrice $M \in \mathcal{M}_2(\mathbb{R})$, implémenter une fonction `inverse2x2` qui prend en argument une matrice carrée M d'ordre 2 et qui :
   - affiche un message d'erreur si M n'est pas inversible puis **s'arrête**.
   - renvoie son inverse M^{-1} sinon.
2. Tester votre fonction pour $A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}$ et $C = \begin{pmatrix} 3 & -1 \\ 0 & 11 \end{pmatrix}$.
3. Vérifier la robustesse de votre fonction à l'aide de la fonction `al.inv` . Que se passe-t-il pour la matrice B?

Entrée[ ]:
```
1
```

Exercice 3 - Inverse de matrices plus grandes

1. A l'aide de la fonction `al.matrix_rank` , écrire une fonction `test_inverse` qui prend en argument une matrice M quelconque et renvoie `True` si la matrice est inversible et `False` sinon. *On pourra penser à utiliser la fonction `np.shape` .*

2. Tester votre fonction pour $A = \begin{pmatrix} 1 & 2 & 1 \\ 0 & -3 & -2 \\ 0 & -2 & 0 \end{pmatrix}$, $B = \begin{pmatrix} 4 & -3 & 7 \\ -1 & 6 & 3 \\ 2 & 9 & 13 \end{pmatrix}$, $C = \begin{pmatrix} 1 & 2 & 4 \\ -2 & -4 & -8 \\ 3 & 6 & 12 \end{pmatrix}$ et $D = \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 2 & 3 \\ 4 & 0 \end{pmatrix}$
3. Ecrire une fonction `show_inverse` qui prend en argument une matrice quelconque M et qui :
   - affiche un message d'erreur et **s'arrête** si la matrice n'est pas carrée.
   - affiche un message d'erreur et **s'arrête** si la matrice est carrée non-inversible.
   - affiche un message de succès et **renvoie** la matrice inverse M^{-1}.
4. Tester votre fonction avec les matrices A, B, C et D.

Entrée[ ]:
```
1
```

Exercice 4 - Puissances de matrices

On définit la matrice $A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$.

Conjecturer la forme de A^n pour tout entier naturel n, et vérifier votre conjecture à l'aide d'une commande Python bien choisie.

Entrée[ ]:
```
1
```

Exercice 5 - Somme des premières puissances

1. Écrire une fonction prenant en argument une matrice carrée M et un entier p et renvoyant la somme $I_n + M + \ldots + M^p$.
2. Testez votre fonction avec la matrice A de l'exercice 4.

Entrée[ ]:
```
1
```

Exercice 6 - Résolution de systèmes (1)

On considère le système suivant :

$$(S) : \begin{cases} x + 2y + z &= 5 \\ -3y - 2z &= 2 \\ -2y &= 11 \end{cases}.$$

1. Au brouillon, écrire le système (S) sous forme matricielle : $(S) \sim AX = B$ où $A \in \mathcal{M}_3(\mathbb{R})$.

2. Écrire un programme permettant de déterminer la ou les solutions de ce système.

Entrée[ ]:
```
1
```

Exercice 7 - Résolution de systèmes (2)

Soit $(u_n)_n$, $(v_n)_n$ et $(w_n)_n$ les suites réelles définies par $u_0 = 0$, $v_0 = 1$ et $w_0 = 2$ et pour tout entier naturel n :

$$\begin{cases} u_{n+1} = u_n + v_n + w_n \\ v_{n+1} = v_n + w_n \\ w_{n+1} = w_n \end{cases}. \quad \text{On note aussi } C_n = \begin{pmatrix} u_n \\ v_n \\ w_n \end{pmatrix}.$$

1. Définir la matrice M telle que, pour tout entier naturel n, $C_{n+1} = M \times C_n$.
2. Écrire un programme renvoyant le vecteur C_n pour un entier n de votre choix.
3. À l'aide des questions précédentes, écrire une fonction `sol_suite(n)` qui affiche les termes u_n, v_n et w_n au rang n, et qui ne renvoie rien.

Entrée[ ]:
```
1
```