Ce laboratoire a pour objectif de vous familiariser aux concepts fondamentaux de l'algèbre linéaire et leur manipulation pratique à l'aide de **NumPy**, en ciblant les applications en Intelligence Artificielle.

**Section 1 : Introduction aux Vecteurs (Rappel)**

**Exercice 1.1 : Création et Dimensions**

1. Créez un **vecteur ligne** $\mathbf{v_1}$ contenant les entiers 1, 3, 5, 7.
2. Créez un **vecteur colonne** $\mathbf{v_2}$ contenant les flottants 0.5, -2.1, 4.0.
3. Affichez la **forme** (shape) et le **nombre de dimensions** (ndim) de $\mathbf{v_1}$ et $\mathbf{v_2}$.

**Exercice 1.2 : Produit Scalaire (Dot Product)**

Soient $\mathbf{x} = [2, -1, 0]$ et $\mathbf{y} = [4, 5, 6]$.

1. Calculez le **produit scalaire** $\mathbf{x} \cdot \mathbf{y}$ en utilisant `np.dot()` ou l'opérateur `@`.

**Section 2 : Introduction aux Matrices et Opérations Fondamentales (Rappel)**

**Exercice 2.1 : Création et Attributs**

1. Créez la **matrice** $\mathbf{M}$ suivante :

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

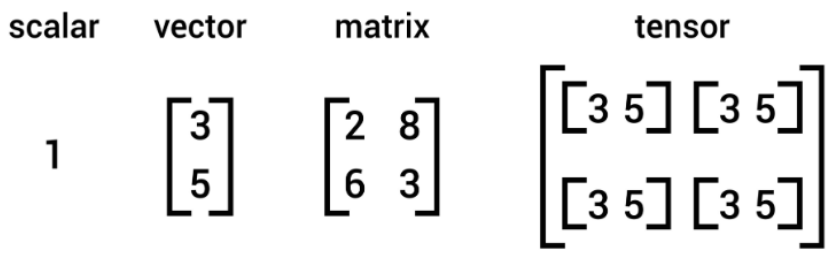
2. Affichez la **forme** ( `shape` ) et calculez la **transposée** $\mathbf{M}^{\mathrm{T}}$.

**Exercice 2.2 : Multiplication Matrice-Matrice**

Soient les matrices $\mathbf{A}$ et $\mathbf{B}$ :

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 0 & 3 \\ 4 & 5 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 6 & 7 & 8 \\ 9 & 10 & 11 \end{pmatrix}$$

1. Calculez le produit matriciel $\mathbf{C} = \mathbf{AB}$ en utilisant l'opérateur `@`.



### Section 3 : Résolution de Systèmes Linéaires 💡

La résolution de systèmes d'équations est essentielle en optimisation et en régression linéaire. Un système peut s'écrire sous la forme $\mathbf{Ax} = \mathbf{b}$.

#### Exercice 3.1 : Utilisation de `np.linalg.solve`

Considérez le système d'équations suivant :

$$\begin{cases} 2x + 1y &= 7 \\ 1x - 3y &= -7 \end{cases}$$

1. Définissez la **matrice des coefficients** $\mathbf{A}$ (les coefficients de $x$ et $y$).
2. Définissez le **vecteur résultat** $\mathbf{b}$.
3. Utilisez la fonction `np.linalg.solve(A, b)` pour trouver le vecteur solution $\mathbf{x} = [x, y]$.
4. Vérifiez votre solution en calculant $\mathbf{Ax}$ et en confirmant qu'il est égal à $\mathbf{b}$.

#### Exercice 3.2 : Inverse de Matrice

La solution d'un système peut aussi être trouvée par $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$.

1. Calculez l'**inverse** de la matrice $\mathbf{A}$ (définie en 3.1) en utilisant `np.linalg.inv()`.
2. Calculez $\mathbf{x}$ en utilisant l'inverse et la multiplication matricielle.
3. *Observation :* Quelle est la différence entre utiliser `solve` et calculer l'inverse pour résoudre un système ? (Réponse : `solve` est généralement **plus rapide** et **plus stable numériquement**).

**Section 4 : Valeurs et Vecteurs Propres (Eigendecomposition)** 📈

Les valeurs et vecteurs propres (Eigenvalues et Eigenvectors) sont à la base de techniques comme la **PCA** (Analyse en Composantes Principales), utilisée pour la réduction de dimensionnalité des données.

**Exercice 4.1 : Calcul de l'Eigendecomposition**

Soit la matrice de covariance $\mathbf{C}$ (symétrique) :

$$\mathbf{C} = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}$$

1. Utilisez la fonction `np.linalg.eig(C)` pour calculer :
   - Le vecteur des **valeurs propres** ($\lambda$, `eigenvalues`).
   - La matrice des **vecteurs propres** ($\mathbf{V}$, `eigenvectors`).

**Exercice 4.2 : Vérification**

La relation fondamentale des vecteurs et valeurs propres est $\mathbf{C}\mathbf{v} = \lambda\mathbf{v}$, où $\mathbf{v}$ est un vecteur propre et $\lambda$ sa valeur propre associée.

1. Extrayez le **premier vecteur propre** $\mathbf{v}_1$ (la première colonne de la matrice de vecteurs propres).
2. Extrayez la **première valeur propre** $\lambda_1$.
3. Calculez le produit matriciel $\mathbf{C}\mathbf{v}_1$ (le côté gauche de l'équation).
4. Calculez le produit scalaire $\lambda_1\mathbf{v}_1$ (le côté droit de l'équation).
5. Vérifiez que les résultats des étapes 3 et 4 sont égaux (ou très proches, compte tenu de la précision des flottants).