TP: matrice 2
Informatique pour tous

Comme toujours, toutes les fonctions doivent être testées.

Rappels sur les tableaux:

- `array` (tableau) est un type similaire à `list`, défini dans le module `numpy`. Il faut donc écrire `import numpy as np` pour l'utiliser.
- `np.array(L)` créé un tableau à partir d'une liste `L`. Par exemple `T = np.array([1, 7, 4])` créé un tableau `T` contenant les éléments 1, 7 et 4.
- Un tableau doit contenir des éléments tous de même type.
- Il est impossible de modifier la taille d'un tableau (sa taille est fixée à sa création).
- **On peut effectuer n'importe qu'elle opération arithmétique sur deux tableaux (de même taille), ce qui donne le tableau obtenu en appliquant l'opération case par case.**
  Exemple: `np.array([7, 1]) + np.array([2, 5])` donne le tableau `np.array([9, 6])`.
- On peut accéder/modifier le $i$ème élément d'un tableau `T` en écrivant `T[i]` (comme pour les listes).

Une ligne d'une matrice est représentée par un tableau. On rappelle qu'une matrice est représentée par un tableau de ses lignes (donc un tableau de tableaux). Par exemple, la matrice $\begin{pmatrix} 1 & 2 \\ 4 & 3 \\ 2 & 0 \end{pmatrix}$ est définie par `np.array([[1, 2], [4, 3], [2, 0]])`.

1. Écrire une fonction `transpose` renvoyant la transposée d'une matrice. On fera en sorte que cette fonction marche même pour des matrices qui ne sont pas carrés. On pourra utiliser `np.zeros((n, p))` pour créer une matrice de taille $n \times p$ avec que des zéros.
2. Écrire une fonction déterminant si une matrice est symétrique, c'est à dire égale à sa transposée.
3. Écrire une fonction `inverse` renvoyant l'inverse d'une matrice 2×2, en supposant que cet inverse existe. On rappelle que l'inverse de $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ est $\dfrac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.
   Vérifier que votre fonction marche en calculant le produit d'une matrice inversible `M` par `inverse(M)`. Pour cela vous pouvez utiliser `np.dot(A, B)` qui renvoie le produit matriciel de `A` par `B`.
4. Écrire une fonction `dilatation` telle que `dilatation(M, i, lambda)` multiplie la ligne `i` de la matrice `M` par `lambda`, c'est à dire qui effectue l'opération $L_i \leftarrow \lambda L_i$.
5. Écrire une fonction `transvection` telle que `transvection(M, i, j, lambda)` effectue l'opération $L_i \leftarrow L_i + \lambda L_j$.
6. Écrire une fonction `echange` telle que `echange(M, i, j)` échange les lignes i et j de `M`.