Python scientifique débutant
TP `numpy`

Dominique Benielli, Eulalio Torres Garcia, Valentin Emiya - Aix-Marseille Université

Novembre 2023

Les exercices de ce TP ont pour objectif de s'exercer à manipuler les tableaux numpy en améliorant progressivement la technique de programmation. Dans la plupart des exercices, il y a plusieurs solutions possibles, notamment en utilisant des boucles ou pas.

Exercice 1. En utilisant des tableaux numpy à une dimension,

1. créez un vecteur u de $N = 20$ entiers tirés aléatoirement entre 0 et 100 (`np.random.randint`) ;
2. créez un vecteur b de N booléens tel que $b[i] =$ `True` si et seulement si $u_i > 60$;
3. modifiez u en remplaçant tous les coefficients supérieurs à 60 par -1 ;
4. créez une matrice a de taille 5×10 composée d'entiers tirés aléatoirement entre -100 et 100 puis modifiez-la en multipliant par deux tous les coefficients dont la valeur absolue est inférieure à 50.

Exercice 2 (Matrice damier)**.** Créez une matrice 5×9 contenant des 0 et des 1 en alternance sur les lignes et les colonnes.

$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$$

Recommencez avec une matrice 6×10.

Exercice 3 (Exploration de données de contrôle non-destructif.)**.**
Le fichier `5MHz_S45S_PCS250mm_SONA1.txt` contient un tableau à 2 dimensions contenant des données issues de signaux ultrasonores (de type A-scan pour les intimes). Il y 1505 enregistrements, chaque enregistrement étant un vecteur de 2226 coefficients (un signal enregistré avec une fréquence d'échantillonnage de 25Mhz, toujours pour les intimes). Ce vecteur correspond au signal transmis à travers un matériau, en avançant par pas de 1mm entre deux enregistrement successif. On peut ainsi détecter des défauts lorsque le signal enregistré a peu d'énergie.

1. Chargez le contenu du fichier dans un tableau numpy (`np.loadtxt`)
2. Affichez le nombre de dimensions (`.ndim`).
3. Affichez le nombre de lignes et de colonnes (`.shape`).
4. Affichez le nombre total d'éléments (`.size`).
5. Vérifiez que tous les coefficients sont positifs (`np.all`)
6. Calculez la somme des coefficients de chaque ligne (`np.sum(..., axis=...)`).
7. Déterminez dans quel canal il y a le plus d'énergie en extrayant le maximum dans le vecteur précédent. Idem pour le minimum d'énergie (`np.argmax`, `np.argmin`). Affichez le numéro de la ligne et la valeur du minimum et du maximum.