EXERCICE 1

# Créer le vecteur x

x = np.array([1, 2, 3, 4, 5])

# Afficher le type de x puis sa longueur

print("Type de x:", type(x))

print("Longueur de x:", len(x))

# Extraire le premier élément, puis en faire de même avec le dernier

print("Premier élément de x:", x[0])

print("Dernier élément de x:", x[-1])

# Extraire les trois premiers éléments et les stocker dans un vecteur nommé a

a = x[:3]

print("Vecteur a:", a)

# Extraire les 1er, 2e et 5e éléments du vecteur et les stocker dans un vecteur nommé b

b = x[[0, 1, 4]]

print("Vecteur b:", b)

# Additionner le nombre 10 au vecteur x, puis multiplier le résultat par 2

x\_transformed = (x + 10) \* 2

print("Vecteur x transformé:", x\_transformed)

# Effectuer l'addition de a et b, commenter le résultat

addition\_a\_b = a + b

print("Addition de a et b:", addition\_a\_b)

# Commentaire : L'addition de a et b additionne les éléments correspondants des deux vecteurs.

# Effectuer l'addition suivante: x + a; commenter le résultat

# Pour que cette opération soit possible, nous devons ajuster la taille de a

a\_padded = np.pad(a, (0, len(x) - len(a)), 'constant')

addition\_x\_a = x + a\_padded

print("Addition de x et a:", addition\_x\_a)

# Commentaire : a a été étendu avec des zéros pour correspondre à la taille de x avant l'addition.

# Multiplier le vecteur x par le scalaire c que l'on fixe à 2

c = 2

x\_multiplied = x \* c

print("Vecteur x multiplié par 2:", x\_multiplied)

# Effectuer la multiplication de a et b; commenter le résultat

multiplication\_a\_b = a \* b

print("Multiplication de a et b:", multiplication\_a\_b)

# Commentaire : La multiplication de a et b multiplie les éléments correspondants des deux vecteurs.

# Effectuer la multiplication suivante: x \* a; commenter le résultat

multiplication\_x\_a = x \* a\_padded

print("Multiplication de x et a:", multiplication\_x\_a)

# Commentaire : a a été étendu avec des zéros pour correspondre à la taille de x avant la multiplication.

# Récupérer les positions des multiples de 2 et les stocker dans un vecteur nommé ind

ind = np.where(x % 2 == 0)[0]

print("Positions des multiples de 2 dans x:", ind)

# Conserver uniquement les multiples de 2 de x dans un vecteur nommé mult\_2

mult\_2 = x[x % 2 == 0]

print("Multiples de 2 dans x:", mult\_2)

# Afficher les éléments de x qui sont multiples de 3 et multiples de 2

multiples\_3\_2 = x[(x % 3 == 0) & (x % 2 == 0)]

print("Éléments de x multiples de 3 et de 2:", multiples\_3\_2)

# Afficher les éléments de x qui sont multiples de 3 ou multiples de 2

multiples\_3\_or\_2 = x[(x % 3 == 0) | (x % 2 == 0)]

print("Éléments de x multiples de 3 ou de 2:", multiples\_3\_or\_2)

# Calculer la somme des éléments de x

somme\_x = np.sum(x)

print("Somme des éléments de x:", somme\_x)

# Remplacer le premier élément de x par un 4

x[0] = 4

print("Vecteur x après remplacement du premier élément par 4:", x)

# Remplacer le premier élément de x par la valeur NaN, puis calculer la somme des éléments de x

x[0] = np.nan

somme\_x\_nan = np.nansum(x)

print("Somme des éléments de x avec NaN:", somme\_x\_nan)

# Supprimer le vecteur x

del x

print("Vecteur x supprimé")

EXERCICE 2

1. Créer la matrice `A`

```python

import numpy as np

A = np.array([[-1, 2], [-1, -1], [2, 1]])

print("Matrice A créée:\n", A)

```

2. Afficher la dimension de `A`, son nombre de colonnes, son nombre de lignes et sa longueur

```python

print("Dimension de A:", A.shape)

print(" Nombre de colonnes de A:", A.shape[1])

print(" Nombre de lignes de A:", A.shape[0])

print(" Longueur de A:", A.size)

```

3. Extraire la seconde colonne de `A`, puis la première ligne

```python

seconde\_colonne = A[:, 1]

premiere\_ligne = A[0, :]

print("Seconde colonne de A:", seconde\_colonne)

print(" Première ligne de A:", premiere\_ligne)

```

4. Extraire l'élément en troisième position à la première ligne

```python

element\_troisieme\_position = A[0, 1]

print("Élément en troisième position à la première ligne:", element\_troisieme\_position)

```

5. Extraire la sous-matrice de dimension 2x2 du coin inférieur de `A`

```python

sous\_matrice = A[1:, :]

print("Sous-matrice 2x2 du coin inférieur de A:\n", sous\_matrice)

```

6. Calculer la somme des colonnes puis des lignes de `A`

```python

somme\_colonnes = np.sum(A, axis=0)

somme\_lignes = np.sum(A, axis=1)

print("Somme des colonnes de A:", somme\_colonnes)

print(" Somme des lignes de A:", somme\_lignes)

```

7. Afficher la diagonale de `A`

```python

diagonale = np.diag(A)

print("Diagonale de A:", diagonale)

```

8. Ajouter le vecteur `[1, 2, 3]` à droite de la matrice `A` et stocker le résultat dans un objet appelé `B`

```python

vecteur = np.array([[1], [2], [3]])

B = np.hstack((A, vecteur))

print("Matrice B après ajout du vecteur à droite de A:\n", B)

```

9. Retirer le quatrième vecteur de `B`

```python

B\_sans\_quatrieme = B[:, :-1]

print("Matrice B après retrait du quatrième vecteur:\n", B\_sans\_quatrieme)

```

10. Retirer la première et la troisième ligne de `B`

```python

B\_sans\_1\_3 = np.delete(B, [0, 2], axis=0)

print("10. Matrice B après retrait des première et troisième lignes:\n", B\_sans\_1\_3)

```

11. Ajouter le scalaire 10 à `A`

```python

A\_plus\_10 = A + 10

print("Matrice A après ajout du scalaire 10:\n", A\_plus\_10)

```

Ajouter le vecteur `[1, 2, 3]'` à `A`

```python

vecteur\_colonne = np.array([[1], [2], [3]])

A\_plus\_vecteur = A + vecteur\_colonne

print("Matrice A après ajout du vecteur [1, 2, 3]':\n", A\_plus\_vecteur)

```