Fiche d'exercices 2 : Numpy

Ali ZAINOUL

Exercice 1 : Création de tableaux NumPy

- Créez une fonction `create_array_from_list()` qui prend une liste Python en entrée et retourne un tableau NumPy.
- Testez cette fonction avec la liste suivante : `[1, 2, 3, 4, 5]`.
- Ajoutez un commentaire expliquant chaque étape.

Exercice 2 : Utilisation de la fonction `arange()`

- Créez une fonction `create_array_arange()` qui retourne un tableau contenant les nombres de 0 à 20 avec un pas de 2.
- Utilisez ensuite une fonction nommée `reshape_array()` pour transformer ce tableau en un tableau 2x5.
- Documentez chaque fonction avec une docstring.

Exercice 3 : La fonction `linspace()`

- Créez une fonction `generate_linear_space()` qui prend trois arguments : `start`, `end`, et `num`. La fonction doit retourner un tableau NumPy contenant `num` valeurs également espacées entre `start` et `end`.
- Ajoutez des assertions pour valider les entrées.

Exercice 4 : Création de tableaux avec des constantes

- Créez une fonction `create_ones_array()` qui prend en entrée les dimensions souhaitées et retourne un tableau rempli de 1s.
- Créez une fonction similaire `create_constant_array()` qui prend en entrée les dimensions et une constante, et retourne un tableau rempli de cette constante.

Exercice 5 : La fonction `eye()`

- Créez une fonction `create_identity_matrix()` qui prend un entier `n` et retourne une matrice identité de taille `n`.

— Créez une fonction `create_identity_and_flatten()` qui utilise `create_identity_matrix()` et la fonction `ravel()` pour retourner une version aplatie de la matrice identité.

Exercice 6 : Nombres aléatoires

— Créez une fonction `generate_random_array()` qui retourne un tableau 2x3 contenant des nombres aléatoires entre 0 et 1.
— Créez une fonction `analyze_array()` qui prend ce tableau en entrée, affiche sa moyenne et son écart-type, et retourne ces valeurs sous forme de tuple.

Exercice 7 : Manipulation des dimensions

— Créez une fonction `create_3d_array()` qui retourne un tableau 3x4x5.
— Créez une fonction `flatten_and_reshape()` qui utilise `ravel()` et `reshape()` pour aplatir puis remodeler ce tableau en un tableau 6x10.
— Documentez l'importance de la gestion des dimensions dans le code.

Exercice 8 : Fusion de tableaux

— Créez deux fonctions `create_ones_array()` et `create_twos_array()` qui retournent respectivement des tableaux 2x3 remplis de 1s et de 2s.
— Créez une fonction `merge_arrays()` qui utilise `vstack()` et `hstack()` pour fusionner ces tableaux verticalement et horizontalement, puis retourne les deux résultats.

Exercice 9 : Attributs des tableaux NumPy

— Créez une fonction `create_random_int_array()` qui retourne un tableau 3x4x2 rempli de nombres aléatoires entiers entre 0 et 100.
— Créez une fonction `display_attributes()` qui prend un tableau en entrée et affiche ses attributs : `ndim`, `shape`, `size`, `dtype`.

Exercice 10 : La fonction `squeeze()`

— Créez une fonction `create_complex_array()` qui retourne un tableau de forme 1x4x1x5.
— Créez une fonction `simplify_dimensions()` qui utilise `squeeze()` et `reshape()` pour simplifier les dimensions et transformer ce tableau en un tableau 4x5.
— Expliquez pourquoi la simplification des dimensions peut être utile.

Projet final : Analyse de données avec NumPy

- Créez une fonction `generate_temperatures()` qui retourne un tableau NumPy modélisant les températures moyennes journalières (en °C) d'une ville sur un mois (30 jours).
- Créez une fonction `analyze_temperatures()` qui prend le tableau des températures, calcule la température moyenne, les jours au-dessus de la moyenne, la variance et l'écart-type, et retourne ces valeurs.
- Utilisez la bibliothèque Matplotlib pour créer un graphique représentant la température sur chaque jour du mois.
- Créez une fonction `normalize_temperatures()` pour normaliser les températures entre 0 et 1.
- Documentez chaque étape de ce projet avec des commentaires et des docstrings.