### Introduction à NumPy

NumPy (Numerical Python) est une bibliothèque fondamentale pour le calcul scientifique en Python. Elle offre des structures de données puissantes comme les tableaux (arrays) et les matrices, ainsi que des fonctions mathématiques pour manipuler ces tableaux avec une grande efficacité. Pour les sciences de la nature, où l'analyse de grandes quantités de données est courante, NumPy est un outil indispensable.

#### Installation

Pour installer NumPy, utilisez pip:

```bash

pip install numpy

```

#### Création et Manipulation de Tableaux

1. \*\*Création de tableaux\*\*:

```python

import numpy as np

# Création d'un tableau 1D

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

# Création d'un tableau 2D

b = np.array([[1, 2, 3], [4, 5, 6]])

print(a)

print(b)

```

2. \*\*Accès aux éléments\*\*:

```python

# Accès à un élément du tableau 1D

print(a[2]) # Sortie: 3

# Accès à un élément du tableau 2D

print(b[1, 2]) # Sortie: 6

```

3. \*\*Opérations mathématiques\*\*:

```python

# Addition

c = a + 10

print(c) # Sortie: [11 12 13 14 15]

# Multiplication

d = a \* 2

print(d) # Sortie: [ 2 4 6 8 10]

# Racine carrée

e = np.sqrt(a)

print(e) # Sortie: [1. 1.41421356 1.73205081 2. 2.23606798]

```

4. \*\*Statistiques\*\*:

```python

# Moyenne

print(np.mean(a)) # Sortie: 3.0

# Écart-type

print(np.std(a)) # Sortie: 1.4142135623730951

# Somme

print(np.sum(a)) # Sortie: 15

```

5. \*\*Manipulation de la forme des tableaux\*\*:

```python

# Redimensionnement

b\_reshaped = b.reshape(3, 2)

print(b\_reshaped)

# Transpose

b\_transposed = b.T

print(b\_transposed)

```

### Tableau des fonctions et méthodes incontournables

| Fonction/Méthode | Description |

|-----------------------------|-----------------------------------------------------------------------------------|

| `np.array()` | Crée un tableau NumPy à partir d'une liste ou d'une liste de listes. |

| `np.mean()` | Calcule la moyenne des éléments du tableau. |

| `np.std()` | Calcule l'écart-type des éléments du tableau. |

| `np.sum()` | Calcule la somme des éléments du tableau. |

| `np.sqrt()` | Calcule la racine carrée des éléments du tableau. |

| `np.reshape()` | Change la forme du tableau sans modifier ses données. |

| `np.transpose()` | Transpose le tableau (intervertit les lignes et les colonnes). |

| `np.arange()` | Crée un tableau contenant une séquence d'entiers. |

| `np.linspace()` | Crée un tableau contenant une séquence de valeurs régulièrement espacées. |

| `np.random.random()` | Génère un tableau de dimensions spécifiées rempli de nombres aléatoires compris entre 0 et 1. |

### Exercices Pratiques

#### Exercice 1: Analyse des données de précipitations

Objectif: Calculer des statistiques descriptives des précipitations et visualiser les données.

Données: Un fichier CSV nommé `precipitations.csv` contenant les colonnes `Année` et `Précipitation` (en mm).

Instructions:

1. \*\*Charger les données\*\*:

```python

import numpy as np

import pandas as pd

df = pd.read\_csv('precipitations.csv')

precipitations = df['Précipitation'].values

```

2. \*\*Calculer des statistiques descriptives\*\*:

```python

moyenne = np.mean(precipitations)

ecart\_type = np.std(precipitations)

somme = np.sum(precipitations)

print(f'Moyenne: {moyenne}, Écart-type: {ecart\_type}, Somme: {somme}')

```

3. \*\*Visualiser les données avec Matplotlib\*\*:

```python

import matplotlib.pyplot as plt

plt.plot(df['Année'], precipitations)

plt.xlabel('Année')

plt.ylabel('Précipitation (mm)')

plt.title('Précipitations annuelles')

plt.show()

```

#### Exercice 2: Analyse des données de patients

Objectif: Étudier la distribution d'âges des patients et comparer les taux de cholestérol par groupe d'âge.

Données: Un fichier CSV nommé `patients.csv` contenant les colonnes `Age`, `Sexe`, `Cholestérol`.

Instructions:

1. \*\*Charger les données\*\*:

```python

import numpy as np

import pandas as pd

df = pd.read\_csv('patients.csv')

ages = df['Age'].values

cholesterol = df['Cholestérol'].values

```

2. \*\*Calculer des statistiques descriptives pour les âges\*\*:

```python

age\_moyen = np.mean(ages)

age\_ecart\_type = np.std(ages)

print(f'Âge moyen: {age\_moyen}, Écart-type: {age\_ecart\_type}')

```

3. \*\*Créer des groupes d'âge\*\*:

```python

bins = np.arange(0, 101, 20)

groups = np.digitize(ages, bins)

```

4. \*\*Calculer la moyenne de cholestérol par groupe d'âge\*\*:

```python

cholesterol\_moyen\_par\_groupe = [np.mean(cholesterol[groups == i]) for i in range(1, len(bins))]

```

5. \*\*Visualiser les données avec Matplotlib\*\*:

```python

import matplotlib.pyplot as plt

plt.bar(range(len(cholesterol\_moyen\_par\_groupe)), cholesterol\_moyen\_par\_groupe, tick\_label=[f'{bins[i]}-{bins[i+1]-1}' for i in range(len(bins)-1)])

plt.xlabel('Groupe d\'âge')

plt.ylabel('Cholestérol moyen (mg/dL)')

plt.title('Cholestérol moyen par groupe d\'âge')

plt.show()

```

Ces exercices te donneront une bonne base pour utiliser NumPy en combinaison avec Pandas et Matplotlib pour analyser des données scientifiques. Bon courage

! 🚀

### Chapitre sur la Régression Linéaire

La régression linéaire est une technique statistique utilisée pour modéliser la relation entre une variable dépendante (souvent notée \(y\)) et une ou plusieurs variables indépendantes (notées \(x\)). En termes simples, c'est une méthode qui permet de tracer une ligne droite (appelée droite de régression) qui s'ajuste le mieux aux données.

#### Régression Linéaire sans Bibliothèques

1. \*\*Concept de Base\*\*:

La formule de la régression linéaire simple est :

\[

y = mx + b

\]

où :

- \(y\) est la variable dépendante.

- \(x\) est la variable indépendante.

- \(m\) est la pente de la ligne.

- \(b\) est l'ordonnée à l'origine (le point où la ligne coupe l'axe des y).

2. \*\*Calcul de \(m\) et \(b\)\*\*:

Pour trouver les valeurs de \(m\) et \(b\), on utilise les formules suivantes :

\[

m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}

\]

\[

b = \frac{(\sum y)(\sum x^2) - (\sum x)(\sum xy)}{n(\sum x^2) - (\sum x)^2}

\]

où \(n\) est le nombre de points de données.

3. \*\*Exemple en Python\*\*:

```python

# Données

x = [1, 2, 3, 4, 5]

y = [2, 3, 5, 7, 11]

# Calcul des paramètres

n = len(x)

sum\_x = sum(x)

sum\_y = sum(y)

sum\_x\_sq = sum([i\*\*2 for i in x])

sum\_xy = sum([x[i]\*y[i] for i in range(n)])

m = (n \* sum\_xy - sum\_x \* sum\_y) / (n \* sum\_x\_sq - sum\_x\*\*2)

b = (sum\_y \* sum\_x\_sq - sum\_x \* sum\_xy) / (n \* sum\_x\_sq - sum\_x\*\*2)

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

#### Régression Linéaire avec NumPy

NumPy simplifie grandement le processus de calcul des paramètres de la régression linéaire. Voici comment :

1. \*\*Utilisation de NumPy\*\*:

```python

import numpy as np

# Données

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

y = np.array([2, 3, 5, 7, 11])

# Calcul des paramètres

A = np.vstack([x, np.ones(len(x))]).T

m, b = np.linalg.lstsq(A, y, rcond=None)[0]

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

### Exercices Pratiques

#### Exercice 1: Régression linéaire en biologie

\*\*Objectif\*\*: Trouver la relation entre la concentration d'un nutriment dans le sol et la croissance des plantes.

\*\*Données\*\*: Un fichier CSV nommé `plantes.csv` contenant les colonnes `Concentration` (en ppm) et `Croissance` (en cm).

\*\*Instructions\*\*:

1. \*\*Charger les données\*\*:

```python

import pandas as pd

df = pd.read\_csv('plantes.csv')

x = df['Concentration'].values

y = df['Croissance'].values

```

2. \*\*Effectuer la régression linéaire sans bibliothèques\*\*:

```python

n = len(x)

sum\_x = sum(x)

sum\_y = sum(y)

sum\_x\_sq = sum(x\*\*2)

sum\_xy = sum(x \* y)

m = (n \* sum\_xy - sum\_x \* sum\_y) / (n \* sum\_x\_sq - sum\_x\*\*2)

b = (sum\_y \* sum\_x\_sq - sum\_x \* sum\_xy) / (n \* sum\_x\_sq - sum\_x\*\*2)

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

3. \*\*Effectuer la régression linéaire avec NumPy\*\*:

```python

import numpy as np

A = np.vstack([x, np.ones(len(x))]).T

m, b = np.linalg.lstsq(A, y, rcond=None)[0]

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

4. \*\*Visualiser les résultats\*\*:

```python

import matplotlib.pyplot as plt

plt.scatter(x, y, label='Données')

plt.plot(x, m\*x + b, color='red', label='Régression linéaire')

plt.xlabel('Concentration (ppm)')

plt.ylabel('Croissance (cm)')

plt.title('Croissance des plantes en fonction de la concentration de nutriments')

plt.legend()

plt.show()

```

#### Exercice 2: Régression linéaire en santé

\*\*Objectif\*\*: Trouver la relation entre l'âge et le taux de cholestérol.

\*\*Données\*\*: Un fichier CSV nommé `sante.csv` contenant les colonnes `Age` et `Cholestérol`.

\*\*Instructions\*\*:

1. \*\*Charger les données\*\*:

```python

import pandas as pd

df = pd.read\_csv('sante.csv')

x = df['Age'].values

y = df['Cholestérol'].values

```

2. \*\*Effectuer la régression linéaire sans bibliothèques\*\*:

```python

n = len(x)

sum\_x = sum(x)

sum\_y = sum(y)

sum\_x\_sq = sum(x\*\*2)

sum\_xy = sum(x \* y)

m = (n \* sum\_xy - sum\_x \* sum\_y) / (n \* sum\_x\_sq - sum\_x\*\*2)

b = (sum\_y \* sum\_x\_sq - sum\_x \* sum\_xy) / (n \* sum\_x\_sq - sum\_x\*\*2)

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

3. \*\*Effectuer la régression linéaire avec NumPy\*\*:

```python

import numpy as np

A = np.vstack([x, np.ones(len(x))]).T

m, b = np.linalg.lstsq(A, y, rcond=None)[0]

print(f'Pente (m): {m}')

print(f'Ordonnée à l\'origine (b): {b}')

```

4. \*\*Visualiser les résultats\*\*:

```python

import matplotlib.pyplot as plt

plt.scatter(x, y, label='Données')

plt.plot(x, m\*x + b, color='red', label='Régression linéaire')

plt.xlabel('Age')

plt.ylabel('Cholestérol (mg/dL)')

plt.title('Cholestérol en fonction de l\'âge')

plt.legend()

plt.show()

```

Ces exercices te donneront une compréhension pratique de la régression linéaire en utilisant des données scientifiques. Bon apprentissage! 🚀