1. Bibliothèque NumPy

NumPy est une bibliothèque pour langage de programmation Python, Elle fournit des structures de données de haut niveau ainsi que des fonctions pour manipuler des matrices ou tableaux multidimensionnels.

Importation de NumPy

Pour importer NumPy dans un programme Python:

```
import numpy as np
```

1.1. Classe ndarray

L'objet principal fourni par NumPy est un tableau multidimensionnel homogène appelé **ndarray**. Il fournit un moyen efficace pour stocker et manipuler des données de manière optimisée pour les opérations numériques rapides.

1.1.1. *Attribut de la classe ndarray*

Les attributs les plus importants d'un objet **ndarray** sont :

Attribut	Description
`ndarray.ndim`	Renvoie le nombre de dimensions du tableau.
`ndarray.shape`	Renvoie un tuple indiquant la taille du tableau dans chaque dimension. Pour une matrice avec n lignes et m colonnes, la shape sera (n,m).
`ndarray.size`	Renvoie le nombre total d'éléments du tableau. Ceci est égal au produit des éléments de shape.
`ndarray.dtype`	Renvoie le type de données des éléments du tableau.
`ndarray.itemsize`	Renvoie la taille en octets de chaque élément du tableau.
`ndarray.data`	Renvoie le tampon de mémoire contenant les éléments du tableau.

Exemple

```
import numpy as np
# Création d'un tableau multidimensionnel
m = np.array([[1, 2, 3], [4, 5, 6]])

# Attributs de la classe ndarray
print("Shape du tableau : ", m.shape) # Renvoie la taille du tableau dans chaque dimension
print("Nombre de dimensions du tableau : ", m.ndim) # Renvoie le nombre de dimensions du tableau
print("Nombre total d'éléments du tableau : ", m.size) # Renvoie le nombre total d'éléments du tableau
print("Type de données des éléments du tableau : ", m.dtype) # Renvoie le type de données des éléments du tableau
print("Taille en octets de chaque élément du tableau : ", m.itemsize) # Renvoie la taille en octets de chaque élément du tableau
print("Tampon de mémoire contenant les éléments du tableau : ", m.data) # Renvoie le tampon de mémoire contenant les éléments du tableau
```

Résultats affichés par le code précèdent :

```
Shape du tableau : (2, 3)
Nombre de dimensions du tableau : 2
Nombre total d'éléments du tableau : 6
Type de données des éléments du tableau : int32
Taille en octets de chaque élément du tableau : 4
Tampon de mémoire contenant les éléments du tableau : <memory at
0x000001986AFE9450>
```

1.1.2. *Création de matrices (ndarray)*

Fonction	Description
np.array()	Convertit une liste ou un tuple en un tableau NumPy. Par exemple, np.array([[1, 2], [3, 4]]) crée une matrice 2x2.
np.zeros()	Crée une matrice de la forme spécifiée remplie de zéros. Par exemple, np.zeros((2, 3)) crée une matrice 2x3 de zéros.
np.ones()	Crée une matrice de la forme spécifiée remplie de uns. Par exemple, np.ones((3, 2)) crée une matrice 3x2 de uns.
np.empty()	Crée une matrice non initialisée de la forme spécifiée. Les valeurs des éléments de la matrice seront initialement indéterminées.
np.arange()	Crée un tableau avec des valeurs espacées uniformément dans un intervalle donné. Par exemple, np.arange(0, 10, 2) crée un tableau de 0 à 10 par incréments de 2.
np.linspace()	Crée un tableau avec des valeurs uniformément espacées dans un intervalle donné. Par exemple, np.linspace(0, 2, 5) crée un tableau de 5 valeurs espacées uniformément entre 0 et 2.
np.eye()	Crée une matrice identité de la taille spécifiée. Par exemple, np.eye(3) crée une matrice identité 3x3.

Exemple

```
import numpy as np

# Création d'une matrice à partir d'une liste
m = np.array([[1, 2, 3], [4, 5, 6]])
print("Matrice à partir d'une liste : \n", m)

#Accés aux elements d'une matrice
m = np.array([[1, 2, 3], [4, 5, 6]])
print("L'element m[1,2] = ", m[1,2])

# Création d'une matrice de zéros
matrice_zeros = np.zeros((2, 3))
print("Matrice de zéros avec np.zeros : \n", matrice_zeros)

# Création d'une matrice de uns
matrice_ones = np.ones((3, 2))
print("Matrice de uns avec np.ones: \n", matrice_ones)

# Création d'une matrice non initialisée
matrice_non_init = np.empty((2, 2))
print("Matrice non initialisée avec np.empty: \n", matrice_non_init)
```

```
# Utilisation de np.arange pour créer un tableau
tableau_arange = np.arange(0, 10, 2)
print("Tableau avec np.arange : ", tableau_arange)

# Utilisation de np.linspace pour créer un tableau
tableau_linspace = np.linspace(0, 2, 5)
print("Tableau avec np.linspace : ", tableau_linspace)

# Création d'une matrice identité
matrice_identite = np.eye(3)
print("Matrice identité avec np.eye: \n", matrice_identite)
```

Résultats affichés par le code précèdent :

```
Matrice à partir d'une liste :
 [[1 2 3]
 [4 5 6]]
L'element m[1,2] =  6
Matrice de zéros avec np.zeros :
 [[0. 0. 0.]
 [0. 0. 0.]]
Matrice de uns avec np.ones:
 [[1. 1.]
 [1. 1.]
 [1. 1.]]
Matrice non initialisée avec np.empty:
 [[2.  1. ]
 [1.5 0.5]]
Tableau avec np.arange :  [0 2 4 6 8]
Tableau avec np.linspace :  [0.  0.5 1.  1.5 2. ]
Matrice identité avec np.eye:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

1.1.3. *Opérations arithmétiques sur les matrices (ndarray)*

Opération	**Syntaxe NumPy**
Addition de matrices	`np.add( m1, m2) ou m1 + m2`
Soustraction de matrices	`np.subtract( m1, m2) ou m1 - m2`
Multiplication de matrices	`np.matmul( m1, m2) ou m1 @ m2`
Produit élément par élément	`np.multiply( m1, m2) ou m1 * m2`
Division élément par élément	`np.divide( m1, m2) ou m1 / m2`
Puissance d'une matrice	`np.linalg.matrix_power(m1, 10)`
Inverse d'une matrice	`np.linalg.inv(matrix)`
Déterminant d'une matrice	`np.linalg.det(matrix)`
Transposée d'une matrice	`np.transpose(matrix) ou matrix.T`

Exemple

```
import numpy as np

# Création de matrices pour les exemples
m1 = np.array([[1, 2], [3, 4]])
m2 = np.array([[5, 6], [7, 8]])

# Addition de matrices
addition = np.add(m1, m2)
print("Résultat de l'addition de matrices :")
print(addition)

# Soustraction de matrices
subtraction = np.subtract(m1, m2)
print("\nRésultat de la soustraction de matrices :")
print(subtraction)

# Multiplication de matrices
multiplication = np.matmul(m1, m2)
print("\nRésultat de la multiplication de matrices :")
print(multiplication)



# Inverse d'une matrice
inverse_m1 = np.linalg.inv(m1)
print("\nInverse de la matrice :")
print(iinverse_m1)

# Déterminant d'une matrice
determinant_m1 = np.linalg.det(m1)
print("\nDéterminant de la matrice :")
print(determinant_m1)

# Transposée d'une matrice
transpose_m1 = np.transpose(m1)
print("\nTransposée de la matrice :")
print(transpose_m1)

# Résolution de systèmes linéaires
coeff_matrix = np.array([[2, 3], [1, -2]])
const_matrix = np.array([8, -3])
solution = np.linalg.solve(coeff_matrix, const_matrix)
print("\nSolution du système linéaire :")
print(solution)

# Puissance matricielle
m = np.array([[0.9,0.075,0.025],[0.15,0.8,0.05] , [0.25,0.25,0.5] ])
puissance = np.linalg.matrix_power(m, 100)
print("\nPuissance d'une matrice' :")
print(puissance)
```

Résultats affichés par le code précèdent :

```
Résultat de l'addition de matrices :
[[ 6  8]
 [10 12]]

Résultat de la soustraction de matrices :
[[-4 -4]
 [-4 -4]]

Résultat de la multiplication de matrices :
[[19 22]
 [43 50]]

Inverse de la matrice :
[[-2.   1. ]
 [ 1.5 -0.5]]

Déterminant de la matrice :
-2.0000000000000004

Transposée de la matrice :
[[1 3]
 [2 4]]

Solution du système linéaire :
[1. 2.]

Puissance d'une matrice' :
[[0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]
 [0.625  0.3125 0.0625]]
```

1.2. Classe Matrix

La classe **matrix** de **NumPy** est une classe spécialisée pour représenter les matrices. Elle est dérivée de la classe **ndarray**, qui est la classe de base pour les tableaux multidimensionnels. La classe **matrix** offre une série de méthodes et d'attributs spécifiques aux matrices, ce qui la rend plus adaptée aux calculs matriciels.

1.2.1. *Création d'une matrice (matrix)*

```python
import numpy as np

# Création de matrice
m = np.matrix([[1, 2], [3, 4]])
```

1.2.2. *Opérations sur les matrices (matrix)*

Opération	Description
*	Multiplication de matrices.
**	Puissance.
.I	Renvoie l'inverse de la matrice.
.T	Renvoie la transposée de la matrice.

Exemple :

```
import numpy as np

# Création dematrices avec la classematrix
m1 = np.matrix([[1, 2], [3, 4]])
m2 = np.matrix([[5, 6], [7, 8]])

# Affichage desmatrices
print("Matrice 1 :")
print(m1)
print("Matrice 2 :")
print(m2)

#multiplication dematrices avec la classematrix
produit = m1 * m2
print("Produit dematrices :")
print(produit)

# Élévation à la puissance avec la classematrix
puissance_deux = m1 ** 10
print("Matrice élevée à la puissance 2 :")
print(puissance_deux)

# Inverse d'unematrice avec laméthode .I
inverse_m1 = m1.I
print("Inverse de lamatrice 1 :")
print(inverse_m1)

# Transposée d'unematrice avec laméthode .T
transpose_m2 = m2.T
print("Transposée de lamatrice 2 :")
print(transpose_m2)
```

Résultats affichés par le code précèdent :

```
Matrice 1 :
[[1 2]
 [3 4]]
Matrice 2 :
[[5 6]
 [7 8]]
Produit dematrices :
[[19 22]
 [43 50]]
Matrice élevée à la puissance 2 :
[[ 4783807  6972050]
 [10458075 15241882]]
Inverse de lamatrice 1 :
[[-2.   1. ]
 [ 1.5 -0.5]]
Transposée de lamatrice 2 :
[[5 7]
 [6 8]]
```

2. Création de graphiques

`Matplotlib.pyplot` est un module de la bibliothèque **`Matplotlib`** qui offre une interface simple pour créer des graphiques en Python. Il est largement utilisé pour générer une variété de visualisations de données, notamment des graphiques en ligne, des diagrammes en boîtes, des nuages de points, des histogrammes et bien plus encore.

Liste de fonctions **`Matplotlib.pyplot`** :

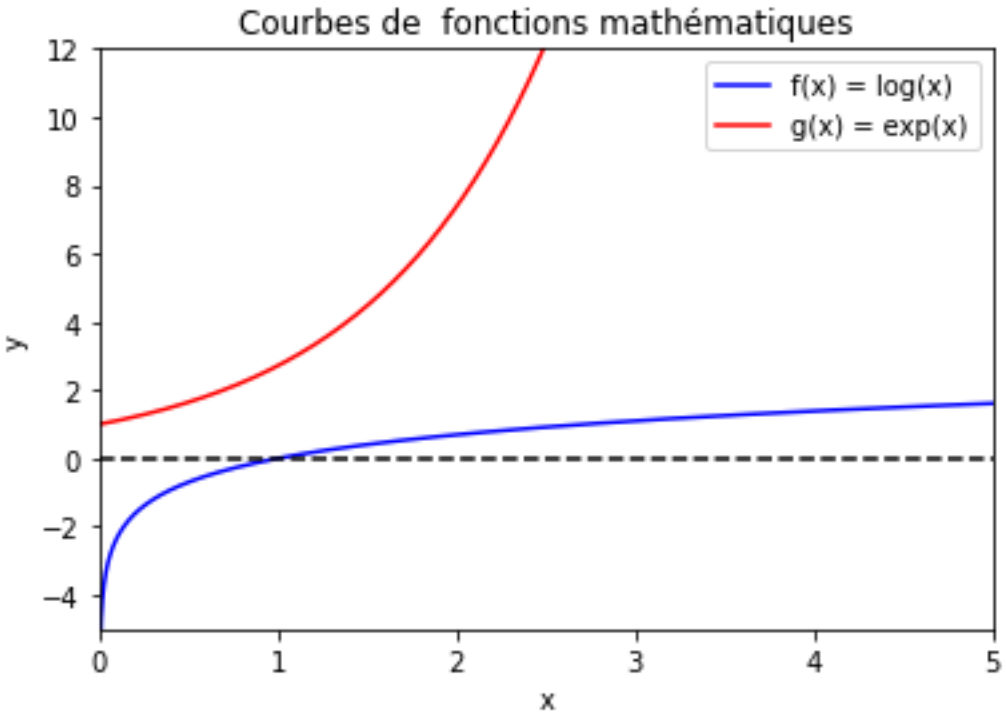
Fonction	Description
plt.plot(x, y, label, linestyle, color, marker)	Trace un graphique en utilisant les valeurs de x et y avec des options supplémentaires pour étiqueter, le style de ligne, la couleur et le marqueur.
scatter(x, y, color, marker)	Trace un nuage de points en utilisant les valeurs de x et y avec des options pour la couleur et le marqueur.
bar(x, height, width, color)	Trace un graphique à barres en utilisant les valeurs de x pour la position horizontale des barres et height pour les hauteurs des barres, avec des options pour la largeur et la couleur.
hist(x, bins, color)	Trace un histogramme en utilisant les valeurs de x avec des options pour les intervalles de classe (bins) et la couleur.
boxplot(data, notch, sym, vert, whis, labels)	Trace un diagramme en boîtes pour représenter la distribution de données numériques.
xlabel(s)	Définit l'étiquette de l'axe des abscisses (axe x).
ylabel(s)	Définit l'étiquette de l'axe des ordonnées (axe y).
title(s)	Définit le titre du graphique.
legend()	Ajoute une légende au graphique.
xlim(left, right)	Définit les limites de l'axe des abscisses (axe x).
ylim(bottom, top)	Définit les limites de l'axe des ordonnées (axe y).
xticks(ticks, labels)	Définit les emplacements des marques sur l'axe des abscisses (axe x).
yticks(ticks, labels)	Définit les emplacements des marques sur l'axe des ordonnées (axe y).
grid(bool)	Ajoute une grille au graphique si la valeur est True.
show()	Affiche le graphique.
plt.plot(x, y, label, linestyle, color, marker)	Enregistre le graphique dans un fichier image avec le nom de fichier spécifié.

Paramètres de la fonction **`pyplot.plot(x, y, label, linestyle, color, marker) :`**

Paramètre	Description
x	Une séquence de valeurs numériques représentant les valeurs de l'axe des abscisses (ou de l'axe x) du graphique.
y	Une séquence de valeurs numériques représentant les valeurs de l'axe des ordonnées (ou de l'axe y) du graphique.
label (optionnel)	Une étiquette pour la série de données, utile pour la légende du graphique. Exemple : label=' y = f(x)'
linestyle (optionnel)	Le style de la ligne. Il peut être spécifié sous forme de chaîne, par exemple, '-' pour une ligne pleine, ':' pour une ligne pointillée, ou '--' pour une ligne en tirets.
color (optionnel)	La couleur de la ligne, qui peut être spécifiée de différentes manières, comme des noms de couleur ou des codes hexadécimaux.
marker (optionnel)	Le marqueur à utiliser sur les points de données, comme 'o' pour des cercles, 's' pour des carrés, 's' pour des carrés, '.' pour des points.

Exemple

```
import matplotlib.pyplot as plt
import numpy as np

# Utilisation de np.arange pour créer un tableau
x = np.arange(0, 10, 0.001)
y = np.log(x)
z = np.exp(x)

# Création d'un graphique
plt.plot(x, y, color='blue', label = 'f(x) = log(x)')

# Création d'un graphique
plt.plot(x, z, linestyle='-', color='red', label = 'g(x) = exp(x)')

# Personnalisation des limites d'axe :
plt.xlim(0, 5)
plt.ylim(-5, 12)

# Ajout des étiquettes auxd'axes :
plt.xlabel('x')
plt.ylabel('y')

# Ajout d'une legende
plt.legend()

# Ajout du titre :
plt.title('Courbes de  fonctions mathématiques')

# Tracer une ligne horizontale à l'ordonnée 0
plt.axhline(y=0, color='black', linestyle='--')
# Affichage du graphique
plt.show()
```


Courbes de fonctions mathématiques
f(x) = log(x)
g(x) = exp(x)
12
10
8
6
4
2
0
−2
−4
y
0
1
2
3
4
5
x