NumPy
Fonctions mathématiques

Par

OUHAB Abdelkrim

Opérations arithmétiques

```
import numpy as np
x = np.array([4, 5, 6])
y = np.array([1, 2, 3])
print("x + y = ", x + y)
print("x - y = ", x - y)
print("x * y = ", x * y)
print("x / y = ", x / y)
```

Output

```
x + y = [5 7 9]
x - y = [3 3 3]
x * y = [ 4 10 18]
x / y = [4. 2.5 2. ]
```

Autres fonctions mathématiques

```
import numpy as np
a = np.array([-3, -1, 0, 1])
b = np.array([9, 16, 2])
# valeur absolue
print(np.absolute(a))
# exponentielle
print(np.exp(b))
# racine carrée
print(np.sqrt(b))
```

Output

```
[3 1 0 1]
[8.10308393e+03 8.88611052e+06 7.38905610e+00]
[3. 4. 1.41421356]
```

Fonctions d'agrégation: somme

- *numpy.sum(arr, axis=None, dtype=None, out=None, initial=0, keepdims=False)*
- arr : Tableau dont les éléments doivent être additionnés
- axis : Axe de calcul de la somme. 0 : par colonne, 1 : par ligne, None : tableau entier
- dtype : Type de données de la somme retournée
- out : Tableau de sortie pour stocker le résultat
- initial : Valeur initiale de la somme
- keepdims : Conserve la dimension de l'axe réduit

Fonctions d'agrégation: somme

```
import numpy as np
a = np.array([10, 20, 30, 40, 50])
print("somme de tous les éléments = ",
 np.sum(a))
print("somme avec initial 100 = ",
np.sum(a, initial=100))
```

Output

```
somme de tous les éléments = 150
somme avec initial 100 = 250
```

Fonctions d'agrégation: somme

```
import numpy as np
arr = np.array([ [10, 10, 10, 10], [20,  20, 20, 20],
  [30,  30, 30,  30] ])
print(np.sum(arr))
print(np.sum(arr, axis=0))
print(np.sum(arr, axis=1))
print(np.sum(arr, axis=1, keepdims=True))
```

Output

```
240
[60 60 60 60]
[ 40 80 120]
[[ 40] [ 80] [120]]
```

Fonctions d'agrégation: moyenne

- *numpy.mean(arr, axis=None, dtype=None, out=None)*

```
import numpy as np
arr = [[10, 10, 10], [20, 20, 20], [30,  30, 30]]
print(np.mean(arr))
print(np.mean(arr, axis=0))
print(np.mean(arr, axis=1))
```

Output

```
20.0
[20. 20. 20.]
[10. 20. 30.]
```

Fonctions d'agrégation: maximum et minimum

- *numpy.maximum(arr1, arr2, out=None, where=True, casting='same_kind', order='K', dtype=None)*
- *numpy.minimum(arr1, arr2, out=None, where=True, casting='same_kind', order='K', dtype=None)*
  - *arr1 : Premier tableau (ou scalaire) d'entrée*
  - *arr2 : Deuxième tableau (ou scalaire) d'entrée*
  - *out (optionnel) : Tableau de destination du résultat*
  - *where : Masque booléen ; les positions sont calculées*
  - *dtype (optionnel) : Type de données de la sortie*
  - *casting / order : Contrôle la conversion des données et l'organisation de la mémoire (rarement utilisé)*

Fonctions d'agrégation: maximum et minimum

```
a = np.array([2, 8, 125])
b = np.array([3, 3, 15])
print("minimum entre a et b :")
print(np.minimum(a, b))
c = np.array([[1, 4, 7], [55, 5, 8]])
d = np.array([3, 0, 1])
print("maximum entre c et d :")
print(np.maximum(c, d))
```

Output

```
minimum entre a et b :
[ 2 3 15]
maximum entre c et d :
[[ 3 4 7] [55 5 8]]
```

Produit de matrices

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
B = np.array([[2, 0], [1, 2]])
produit = A @ B
print("Poduit entre A et B:")
print(produit)
```

Output

```
Poduit entre A et B:
 [[ 4 4]
  [10 8]]
```

Autres opérations sur les matrices

```
import numpy as np
A = np.array([[1, 2], [3, 4]])
transpose_A = A.T
inverse_A = np.linalg.inv(A)
determinant_A = np.linalg.det(A)
print(" transposée_A:")
print(transpose_A)
print("inverse_A:")
print(inverse_A)
print("déterminant _A:")
print(determinant_A)
```

Output

```
transposée_A:
[[1 3]
 [2 4]]
inverse_A:
[[-2.  1. ]
 [ 1.5 -0.5]]
déterminant_A:
 -2.0000000000000004
```