Référence rapide NumPy

Création de tableaux, calcul, algèbre linéaire et plus

Création de tableaux

Depuis des listes

```
import numpy as np
a = np.array([1, 2, 3])        # 1D
b = np.array([[1, 2], [3, 4]])  # 2D
```

Constructeurs intégrés

```
np.zeros((2, 3))     # 2x3 de zéros
np.ones((3, 3))      # 3x3 de uns
np.eye(4)            # matrice identité 4x4
np.arange(0, 10, 2)  # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5) # 5 valeurs régulièrement espacées
```

Propriétés des tableaux

`a.shape`	Dimensions sous forme de tuple : **`(3, 4)`**
`a.ndim`	Nombre de dimensions
`a.size`	Nombre total d'éléments
`a.dtype`	Type de données : **`float64`**, **`int32`**, etc.

Indexation et découpage

Indexation de base

```
a = np.array([[1, 2, 3], [4, 5, 6]])
a[0, 1]    # 2 (ligne 0, col 1)
a[1]       # [4, 5, 6] (ligne 1)
a[:, 0]    # [1, 4] (toutes les lignes, col 0)
```

Découpage

```
a[0, 1:]   # [2, 3] (ligne 0, col 1 et suivantes)
a[:, :2]   # 2 premières colonnes
a[::2]     # une ligne sur deux
```

Indexation booléenne

```
a = np.array([10, 20, 30, 40])
a[a > 15]   # [20, 30, 40]
a[a % 20 == 0]  # [20, 40]
```

Opérations sur les tableaux

Opérations élément par élément

```
a = np.array([1, 2, 3])
a + 10    # [11, 12, 13]
a * 2     # [2, 4, 6]
a ** 2    # [1, 4, 9]
a + a     # [2, 4, 6]
```

Comparaison

```
a = np.array([1, 2, 3, 4])
a > 2          # [False, False, True, True]
np.where(a > 2, a, 0)  # [0, 0, 3, 4]
```

Agrégation

`a.sum()`	Somme de tous les éléments
`a.mean()`	Moyenne arithmétique
`a.std()`	Écart-type
`a.min() / a.max()`	Valeur min / max
`a.argmin() / a.argmax()`	Indice du min / max
`a.cumsum()`	Somme cumulée

Ajouter `axis=0` (colonnes) ou `axis=1` (lignes) pour des résultats par axe

Fonctions mathématiques

Fonctions courantes

`np.sqrt(a)`	Racine carrée de chaque élément
`np.abs(a)`	Valeur absolue
`np.exp(a)`	e^x pour chaque élément
`np.log(a)`	Logarithme naturel (ln)
`np.log10(a)`	Logarithme en base 10
`np.sin(a) / np.cos(a)`	Fonctions trigonométriques (radians)
`np.round(a, 2)`	Arrondir à 2 décimales
`np.clip(a, lo, hi)`	Limiter les valeurs à [lo, hi]

Algèbre linéaire

Opérations matricielles

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
A @ B              # multiplication matricielle
np.dot(A, B)       # identique à A @ B
A.T                # transposée
```

Décomposition et résolution

```
np.linalg.inv(A)       # inverse
np.linalg.det(A)       # déterminant
np.linalg.eig(A)       # valeurs/vecteurs propres
np.linalg.solve(A, b)  # résoudre Ax = b
```

Nombres aléatoires

Génération de nombres aléatoires

```
rng = np.random.default_rng(42)  # graine fixe
rng.random((2, 3))     # uniforme [0, 1)
rng.integers(1, 10, 5) # 5 entiers dans [1, 10)
rng.normal(0, 1, 100)  # 100 valeurs N(0,1)
rng.choice([1, 2, 3], size=2)  # échantillon
```

API héritée

```
np.random.seed(42)
np.random.rand(3, 3)     # uniforme 3x3
np.random.randn(3, 3)    # normale standard
np.random.shuffle(arr)   # mélange en place
```

Remodelage

Manipulation de forme

```
a = np.arange(12)
a.reshape(3, 4)    # matrice 3x4
a.reshape(3, -1)   # déduire les colonnes
a.flatten()        # retour en 1D (copie)
a.ravel()          # retour en 1D (vue)
```

Empilement et division

```
np.vstack([a, b])  # empiler verticalement
np.hstack([a, b])  # empiler horizontalement
np.concatenate([a, b], axis=0)
np.split(a, 3)     # diviser en 3 parties
```

Diffusion (Broadcasting)

Fonctionnement de la diffusion

```
a = np.array([[1, 2, 3],
              [4, 5, 6]])  # forme (2,3)
b = np.array([10, 20, 30]) # forme (3,)
a + b  # b est diffusé vers (2,3)
```

Règles

`Règle 1`	Ajouter des 1 au début de la forme la plus courte jusqu'à égalité des rangs
`Règle 2`	Les dimensions correspondent si elles sont égales ou si l'une vaut 1
`Règle 3`	Les dimensions de taille 1 s'étendent pour correspondre à l'autre

Entrées/Sorties de fichiers

Binaire NumPy

```
np.save("data.npy", arr)     # tableau unique
arr = np.load("data.npy")
np.savez("data.npz", a=x, b=y) # multiples
d = np.load("data.npz"); d["a"]
```

Fichiers texte

```
np.savetxt("data.csv", arr, delimiter=",")
arr = np.loadtxt("data.csv", delimiter=",")
arr = np.genfromtxt("data.csv", delimiter=",",
                    skip_header=1)
```

Motifs courants

Normaliser vers [0, 1]

```
normalized = (a - a.min()) / (a.max() - a.min())
```

Distance euclidienne

```
dist = np.sqrt(np.sum((a - b) ** 2))
# ou : np.linalg.norm(a - b)
```

Valeurs uniques et décomptes

```
vals, counts = np.unique(a, return_counts=True)
dict(zip(vals, counts))
```

Tri

```
np.sort(a)            # copie triée
idx = np.argsort(a)   # indices de tri
a[idx]                # appliquer l'ordre de tri
```