NumPy Riferimento Rapido

Creazione array, matematica, algebra lineare e altro

Creazione di Array

Da Liste

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

Costruttori Integrati

```
np.zeros((2, 3))     # 2x3 di zeri
np.ones((3, 3))      # 3x3 di uni
np.eye(4)            # matrice identità 4x4
np.arange(0, 10, 2)  # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5) # 5 valori equidistanti
```

Proprietà dell'Array

`a.shape`	Dimensioni come tupla: `(3, 4)`
`a.ndim`	Numero di dimensioni
`a.size`	Numero totale di elementi
`a.dtype`	Tipo di dato: `float64`, `int32`, ecc.

Indicizzazione e Slicing

Indicizzazione di Base

```
a = np.array([[1, 2, 3], [4, 5, 6]])
a[0, 1]    # 2 (riga 0, col 1)
a[1]       # [4, 5, 6] (riga 1)
a[:, 0]    # [1, 4] (tutte le righe, col 0)
```

Slicing

```
a[0, 1:]   # [2, 3] (riga 0, col 1 in poi)
a[:, :2]   # prime 2 colonne
a[::2]     # ogni altra riga
```

Indicizzazione Booleana

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

Operazioni sugli Array

Operazioni Elemento per Elemento

```
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]
```

Confronto

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

Aggregazione

`a.sum()`	Somma di tutti gli elementi
`a.mean()`	Media aritmetica
`a.std()`	Deviazione standard
`a.min() / a.max()`	Valore minimo / massimo
`a.argmin() / a.argmax()`	Indice del minimo / massimo
`a.cumsum()`	Somma cumulativa

Aggiungi `axis=0` (colonne) o `axis=1` (righe) per risultati per asse

Funzioni Matematiche

Funzioni Comuni

`np.sqrt(a)`	Radice quadrata di ogni elemento
`np.abs(a)`	Valore assoluto
`np.exp(a)`	e^x per ogni elemento
`np.log(a)`	Logaritmo naturale (ln)
`np.log10(a)`	Logaritmo in base 10
`np.sin(a) / np.cos(a)`	Funzioni trigonometriche (radianti)
`np.round(a, 2)`	Arrotonda a 2 decimali
`np.clip(a, lo, hi)`	Limita i valori a [lo, hi]

Algebra Lineare

Operazioni su Matrici

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
A @ B              # moltiplicazione matriciale
np.dot(A, B)       # uguale a A @ B
A.T                # trasposta
```

Decomposizione e Risoluzione

```
np.linalg.inv(A)       # inversa
np.linalg.det(A)       # determinante
np.linalg.eig(A)       # autovalori/vettori
np.linalg.solve(A, b)  # risolve Ax = b
```

Numeri Casuali

Generazione di Numeri Casuali

```
rng = np.random.default_rng(42)  # con seme
rng.random((2, 3))     # uniforme [0, 1)
rng.integers(1, 10, 5) # 5 interi in [1, 10)
rng.normal(0, 1, 100)  # 100 da N(0,1)
rng.choice([1, 2, 3], size=2)  # campione
```

API Legacy

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

Reshaping

Manipolazione della Forma

```
a = np.arange(12)
a.reshape(3, 4)    # matrice 3x4
a.reshape(3, -1)   # inferisce le colonne
a.flatten()        # torna a 1D (copia)
a.ravel()          # torna a 1D (vista)
```

Stack e Split

```
np.vstack([a, b])  # impila verticalmente
np.hstack([a, b])  # impila orizzontalmente
np.concatenate([a, b], axis=0)
np.split(a, 3)     # divide in 3 parti
```

Broadcasting

Come Funziona il Broadcasting

```
a = np.array([[1, 2, 3],
              [4, 5, 6]])  # forma (2,3)
b = np.array([10, 20, 30]) # forma (3,)
a + b  # b si espande a (2,3)
```

Regole

`Regola 1`	Aggiunge 1 alla forma più corta finché i rank coincidono
`Regola 2`	Le dimensioni coincidono se uguali o una è 1
`Regola 3`	Le dimensioni di dimensione 1 si espandono per coincidere

I/O su File

Formato Binario NumPy

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

File di Testo

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

Pattern Comuni

Normalizza a [0, 1]

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

Distanza Euclidea

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

Valori Unici e Conteggi

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

Ordinamento

```
np.sort(a)            # copia ordinata
idx = np.argsort(a)   # indici che ordinano
a[idx]                # applica l'ordine
```