

Referência Rápida de NumPy

Criação de arrays, matemática, álgebra linear e mais

Criação de Arrays

A Partir de Listas

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

Construtores Integrados

```
np.zeros((2, 3)) # 2x3 of zeros
np.ones((3, 3)) # 3x3 of ones
np.eye(4) # 4x4 identity matrix
np.arange(0, 10, 2) # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5) # 5 evenly spaced
```

Propriedades do Array

a.shape Dimensões como tupla: (3, 4)
a.ndim Número de dimensões
a.size Total de elementos
a.dtype Tipo de dado: float64, int32, etc.

Indexação e Fatiamento

Indexação Básica

```
a = np.array([[1, 2, 3], [4, 5, 6]])
a[0, 1] # 2 (row 0, col 1)
a[1] # [4, 5, 6] (row 1)
a[:, 0] # [1, 4] (all rows, col 0)
```

Fatiamento

```
a[0, 1:] # [2, 3] (row 0, col 1 onward)
a[:, :2] # first 2 columns
a[:, :2] # every other row
```

Indexação Booleana

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

Operações com Arrays

Operações Elemento a 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]
```

Comparação

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

Agregação

a.sum() Soma de todos os elementos
a.mean() Média aritmética
a.std() Desvio padrão
a.min() / **a.max()** Valor mínimo / máximo
a.argmin() / **a.argmax()** Índice do mínimo / máximo
a.cumsum() Soma cumulativa

Adicione `axis=0` (colunas) ou `axis=1` (linhas) para resultados por eixo

Funções Matemáticas

Funções Comuns

np.sqrt(a) Raiz quadrada de cada elemento
np.abs(a) Valor absoluto
np.exp(a) e^x para cada elemento
np.log(a) Logaritmo natural (ln)
np.log10(a) Logaritmo base 10
np.sin(a) / **np.cos(a)** Funções trigonométricas (radianos)
np.round(a, 2) Arredondar para 2 casas decimais
np.clip(a, lo, hi) Limitar valores ao intervalo [lo, hi]

Álgebra Linear

Operações com Matrizes

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
A @ B # matrix multiply
np.dot(A, B) # same as A @ B
A.T # transpose
```

Decomposição e Resolução

```
np.linalg.inv(A) # inverse
np.linalg.det(A) # determinant
np.linalg.eig(A) # eigenvalues/vectors
np.linalg.solve(A, b) # solve Ax = b
```

Números Aleatórios

Geração de Números Aleatórios

```
rng = np.random.default_rng(42) # seeded
rng.random((2, 3)) # uniform [0, 1)
rng.integers(1, 10, 5) # 5 ints in [1, 10)
rng.normal(0, 1, 100) # 100 from N(0,1)
rng.choice([1, 2, 3], size=2) # sample
```

API Legada

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

Reformatação

Manipulação de Forma

```
a = np.arange(12)
a.reshape(3, 4) # 3x4 matrix
a.reshape(3, -1) # infer columns
a.flatten() # back to 1D (copy)
a.ravel() # back to 1D (view)
```

Empilhamento e Divisão

```
np.vstack([a, b]) # stack vertically
np.hstack([a, b]) # stack horizontally
np.concatenate([a, b], axis=0)
np.split(a, 3) # split into 3 parts
```

Broadcasting

Como o Broadcasting Funciona

```
a = np.array([[1, 2, 3],
              [4, 5, 6]]) # shape (2,3)
b = np.array([10, 20, 30]) # shape (3,)
a + b # b broadcasts to (2,3)
```

Regras

- Regra 1** Preencher com 1s à esquerda até que os ranks coincidam
Regra 2 Dimensões compatíveis se iguais ou se uma for 1
Regra 3 Dimensões de tamanho 1 se expandem para coincidir

E/S de Arquivos

Binário NumPy

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

Arquivos de Texto

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

Padrões Comuns

Normalizar para [0, 1]

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

Distância Euclidiana

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

Valores Únicos e Contagens

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

Ordenação

```
np.sort(a) # sorted copy
idx = np.argsort(a) # indices that sort
a[idx] # apply sort order
```