REFERENSI CEPAT NUMPY

Pembuatan array, matematika, aljabar linear, dan lainnya

Pembuatan Array

Dari List

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

Konstruktor Bawaan

```
np.zeros((2, 3))     # 2x3 berisi nol
np.ones((3, 3))      # 3x3 berisi satu
np.eye(4)            # matriks identitas 4x4
np.arange(0, 10, 2)  # [0, 2, 4, 6, 8]
np.linspace(0, 1, 5) # 5 titik berspasi merata
```

Properti Array

`a.shape`	Dimensi sebagai tuple: `(3, 4)`
`a.ndim`	Jumlah dimensi
`a.size`	Total jumlah elemen
`a.dtype`	Tipe data: `float64`, `int32`, dll.

Indexing & Slicing

Indexing Dasar

```
a = np.array([[1, 2, 3], [4, 5, 6]])
a[0, 1]    # 2 (baris 0, kolom 1)
a[1]       # [4, 5, 6] (baris 1)
a[:, 0]    # [1, 4] (semua baris, kolom 0)
```

Slicing

```
a[0, 1:]   # [2, 3] (baris 0, kolom 1 ke atas)
a[:, :2]   # 2 kolom pertama
a[::2]     # setiap baris lainnya
```

Boolean Indexing

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

Operasi Array

Operasi Element-wise

```
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]
```

Perbandingan

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

Agregasi

`a.sum()`	Jumlah semua elemen
`a.mean()`	Rata-rata aritmetika
`a.std()`	Standar deviasi
`a.min() / a.max()`	Nilai min / maks
`a.argmin() / a.argmax()`	Indeks min / maks
`a.cumsum()`	Jumlah kumulatif

Tambahkan `axis=0` (kolom) atau `axis=1` (baris) untuk hasil per sumbu

Fungsi Matematika

Fungsi Umum

`np.sqrt(a)`	Akar kuadrat setiap elemen
`np.abs(a)`	Nilai absolut
`np.exp(a)`	e^x untuk setiap elemen
`np.log(a)`	Logaritma natural (ln)
`np.log10(a)`	Logaritma basis-10
`np.sin(a) / np.cos(a)`	Fungsi trigonometri (radian)
`np.round(a, 2)`	Bulatkan ke 2 desimal
`np.clip(a, lo, hi)`	Batasi nilai ke [lo, hi]

Aljabar Linear

Operasi Matriks

```
A = np.array([[1, 2], [3, 4]])
B = np.array([[5, 6], [7, 8]])
A @ B             # perkalian matriks
np.dot(A, B)      # sama dengan A @ B
A.T               # transpose
```

Dekomposisi & Penyelesaian

```
np.linalg.inv(A)       # invers
np.linalg.det(A)       # determinan
np.linalg.eig(A)       # nilai/vektor eigen
np.linalg.solve(A, b)  # selesaikan Ax = b
```

Random

Pembangkitan Bilangan Acak

```
rng = np.random.default_rng(42)  # dengan seed
rng.random((2, 3))     # seragam [0, 1)
rng.integers(1, 10, 5) # 5 int dalam [1, 10)
rng.normal(0, 1, 100)  # 100 dari N(0,1)
rng.choice([1, 2, 3], size=2)  # sampel
```

API Lama

```
np.random.seed(42)
np.random.rand(3, 3)     # seragam 3x3
np.random.randn(3, 3)    # normal standar
np.random.shuffle(arr)   # acak di tempat
```

Reshaping

Manipulasi Bentuk

```
a = np.arange(12)
a.reshape(3, 4)    # matriks 3x4
a.reshape(3, -1)   # inferensi kolom
a.flatten()        # kembali ke 1D (salinan)
a.ravel()          # kembali ke 1D (view)
```

Penggabungan & Pemisahan

```
np.vstack([a, b])  # susun vertikal
np.hstack([a, b])  # susun horizontal
np.concatenate([a, b], axis=0)
np.split(a, 3)     # pisah menjadi 3 bagian
```

Broadcasting

Cara Kerja Broadcasting

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

Aturan

`Aturan 1`	Tambahkan 1 di depan shape yang lebih pendek hingga rank sama
`Aturan 2`	Dimensi cocok jika sama atau salah satunya adalah 1
`Aturan 3`	Dimensi berukuran 1 diregangkan untuk cocok dengan yang lain

File I/O

Binary NumPy

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

File Teks

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

Pola Umum

Normalisasi ke [0, 1]

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

Jarak Euclidean

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

Nilai Unik & Hitungan

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

Pengurutan

```
np.sort(a)            # salinan terurut
idx = np.argsort(a)   # indeks yang mengurutkan
a[idx]                # terapkan urutan sort
```