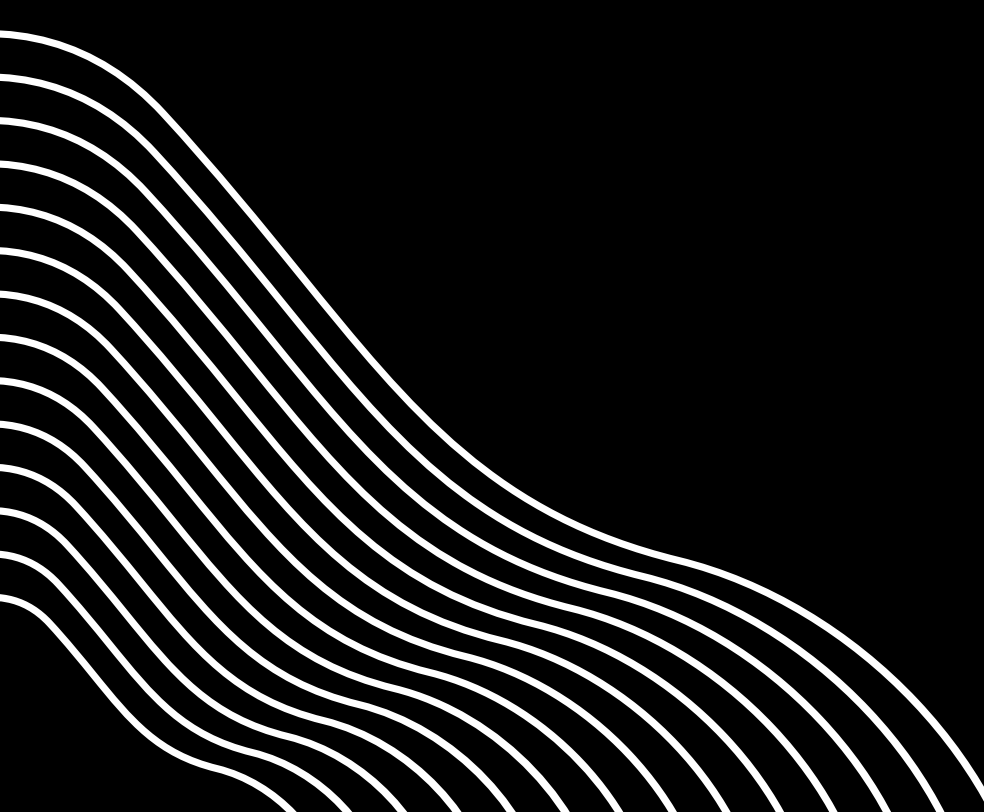





NumPy Sorting Arrays

- ◆ A quick guide to sorting data efficiently using NumPy
 - ◆ Simple examples + clear explanations
- 

1

What is Sorting?

 Sorting = arranging elements in an ordered sequence

- Numeric or alphabetical
- Ascending or descending order
- Essential for data analysis + preprocessing

2

sort() Function

NumPy provides a powerful built-in method:

👉 `np.sort(array)`



Fast



Works on all data types



Easy to use

3

Sorting Numbers(1D)

```
arr = np.array([3, 2, 0, 1])  
np.sort(arr)
```

✓ Output: [0 1 2 3]
Perfect for ranking,
filtering, and
preprocessing.

4

Sorting Strings (1D)

```
arr = np.array(['banana',  
'cherry', 'apple'])  
np.sort(arr)
```

✓ Output: ['apple',
'banana', 'cherry']
Alphabetical sorting
made easy.

5

Sorting Booleans

```
arr = np.array([True,  
False, True])  
np.sort(arr)
```

✓ Output: [False True
True]
False → True (0 → 1)

6

Sorting 2D Arrays

```
arr = np.array([[3, 2, 4],  
                [5, 0, 1]])  
np.sort(arr)
```

- ✓ Each row is sorted individually
- ✓ Useful for matrix operations

7

Ascending & Descending

```
a = np.array([110, 20, -30,  
40])
```

```
np.sort(a)
```

```
np.sort(a)[::-1]
```

✓ Reverse indexing for
descending

✓ Perfect for score
ranking

8

Sorting Column-wise

```
np.sort(a, axis=0)
```

- ✓ Sorts each column independently
- ✓ Helps in table-like data processing

9

Sorting Row-wise

```
np.sort(a, axis=1)
```

- ✓ Sorts each row independently
- ✓ Useful for vector-level ordering

10

Summary

🔥 `np.sort()` is fast, clean, and powerful

🔥 Works with numbers, strings, booleans, and 2D arrays

🔥 Essential skill for ML, data cleaning & analysis