NumPy Arrays for Data Analysis

1. Introduction

NumPy arrays are the backbone of numerical and data analysis in Python. Beyond efficient storage and fast computation, NumPy provides a rich set of **mathematical and statistical methods** for data summarization, transformation, and analysis.

2. Mathematical and Statistical Methods

NumPy offers built-in aggregation functions that operate over entire arrays or along a specified axis. These methods are vectorized and implemented in C, ensuring high performance on large datasets.

programming language

2.1 Aggregation Methods

Method	Description
`sum()`	Sum of all elements
`mean()`	Arithmetic mean
`std()` / `var()`	Standard deviation / variance
`min()` / `max()`	Minimum / maximum value
`argmin()` / `argmax()`	Index of min / max value
`cumsum()`	Cumulative sum
`cumprod()`	Cumulative product

2.2 Using Aggregations on Arrays

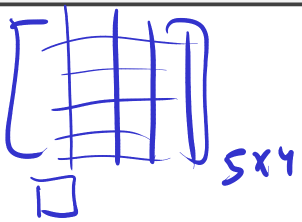
random numbers from the standard normal population

```
import numpy as np
data = np.random.randn(5, 4)  # 5 rows, 4 columns

# Overall mean
print(data.mean())
```

5 rows 4 columns

Col1 Col2 Col3 Col4 N(0,1)



```
# Column-wise sum
print(data.sum(axis=0))

# Row-wise standard deviation
print(data.std(axis=1))
```

axis=0 for column-wise operation

axis=1 for row-wise operation

Note: `axis=0` aggregates down columns, `axis=1` aggregates across rows.

2.3 Cumulative Methods

These are useful for running totals or progressive calculations.

```
arr = np.array([2, 3, 5])

print(arr.cumsum())   # [ 2  5 10]
print(arr.cumprod())  # [ 2  6 30]
```

2+3 2+3+5

2x3 2x3x5

2.4 Boolean Aggregations

Boolean values are treated as `1` (True) and `0` (False).

```
arr = np.random.randn(100)

print((arr > 0).sum())   # Count positives
print((arr < 0).any())   # Check if any negative
print((arr > 2).all())   # Check if all > 2
```

True or False

1 0

3. Why These Methods Matter in Data Analysis

- Fast descriptive statistics on large datasets.
- Efficient computation without explicit loops.
- Integration with `pandas` and machine learning workflows.
- Foundation for exploratory data analysis (EDA).

4. Practical Tip

Use NumPy's vectorized aggregations whenever possible — they are much faster than Python `for` loops and scale to millions of elements efficiently.