# NumPy: The Ultimate Syntax Guide

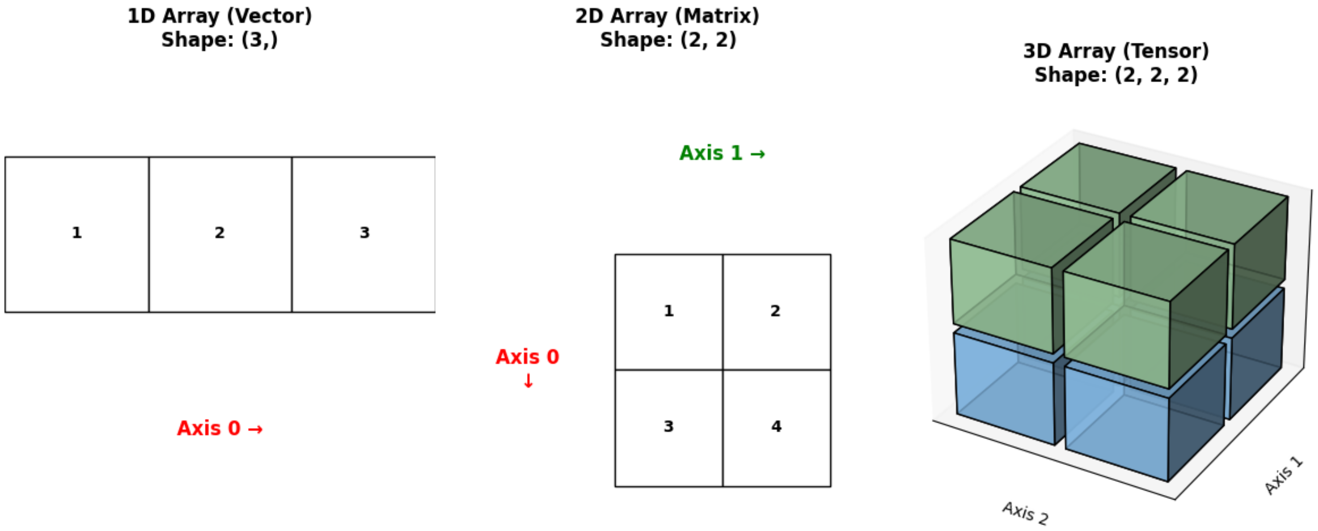
NumPy is the foundation of scientific computing in Python. To master it, you need to understand its **Mental Models** and **Syntax Patterns**.

## 1. The Mental Model

### The N-Dimensional Array

Think of arrays as grids of numbers with a specific **Shape**.

- **1D (Vector)**: A row of numbers. Shape: `(3,)`
- **2D (Matrix)**: A table of numbers. Shape: `(2, 3)` (Rows, Cols)
- **3D (Tensor)**: A cube of numbers. Shape: `(2, 2, 2)` (Depth, Rows, Cols)



### The Golden Rule: "No For Loops" (Vectorization)

In Python, loops are slow. In NumPy, operations apply to the **entire array at once**.

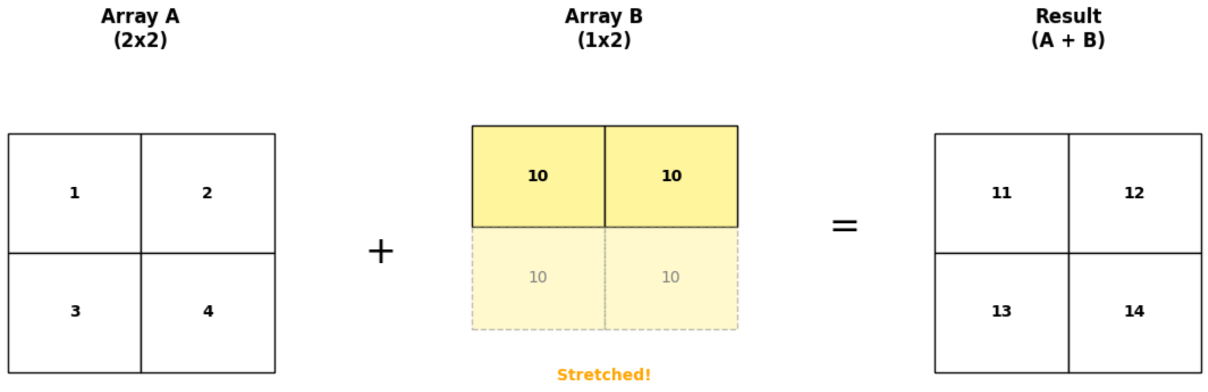
```
# SLOW (Python Loop)
result = []
for x in data:
    result.append(x * 2)

# FAST (NumPy Vectorization)
result = data * 2
```

### The "Magic Stretch" (Broadcasting)

NumPy automatically "stretches" smaller arrays to match larger ones during operations.

- **Scalar to Array**: `arr + 5` (5 is stretched to match `arr`).

- **Vector to Matrix**: Adding a row vector to every row of a matrix.



---

## 2. Universal Syntax Patterns

### Pattern 1: Creation

**Syntax**: `np.function(shape_or_data)`

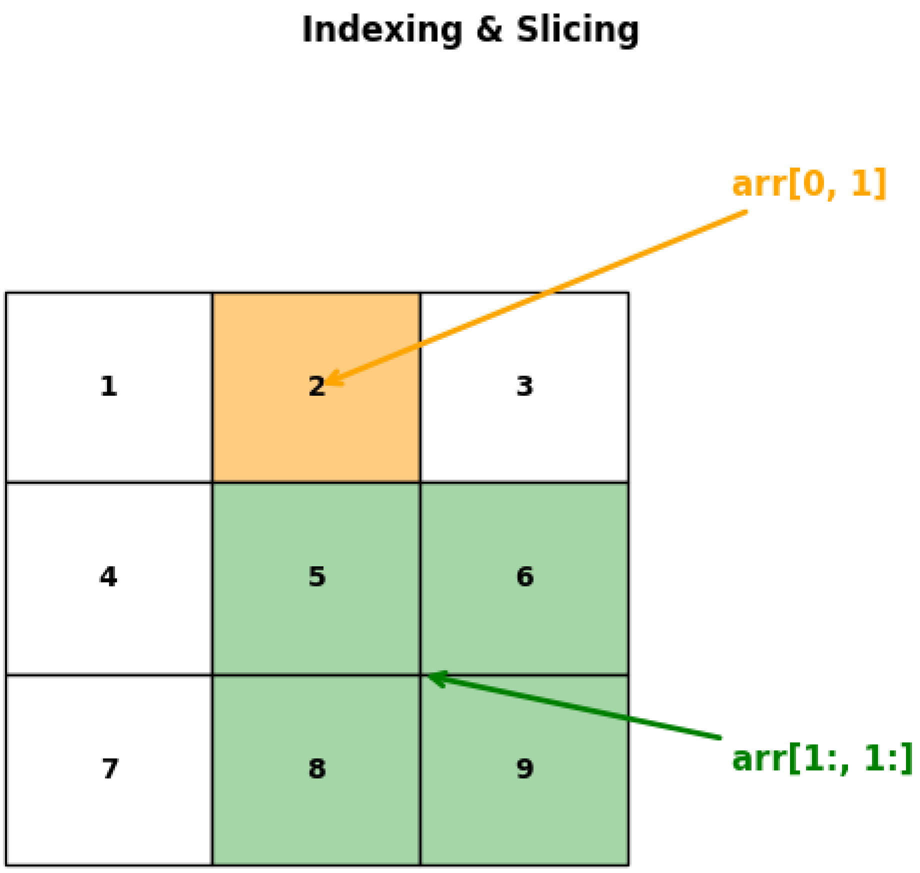
```
# From List
np.array([1, 2, 3])

# Zeros & Ones (Pass shape as tuple!)
np.zeros((3, 3))
np.ones((2, 4))

# Ranges
np.arange(0, 10, 2) # Start, Stop, Step
np.linspace(0, 1, 5) # Start, Stop, Num Points
```

### Pattern 2: The Comma Syntax (Indexing)

**Syntax**: `arr[row, col]`

Unlike Python lists (`list[row][col]`), NumPy uses a single bracket with commas.

```
# Row 0, Column 1
arr[0, 1]

# Slicing: Rows 0 to 2, Column 1 onwards
arr[0:2, 1:]
```



## 3. Essential Methods Cheat Sheet

### Inspection (Know your array)

| Attribute | Description |
|---|---|
| `arr.shape` | Dimensions (e.g., `(2, 3)`) |
| `arr.dtype` | Data type (e.g., `int64`, `float64`) |
| `arr.ndim` | Number of axes (1, 2, or 3) |

### Math (Aggregation)

**Key Concept**: The `axis` argument.

- `axis=0`: Collapse Rows (Move Down).
- `axis=1`: Collapse Columns (Move Across).

| Method | Description |
|---|---|
| `arr.sum()` | Sum of all elements |
| `arr.mean()` | Average |

| Method | Description |
|---|---|
| `arr.max()` | Maximum value |
| `arr.argmax()` | Index of the max value |

```
# Sum of each COLUMN (Collapse rows)
arr.sum(axis=0)
```

### Reshaping

Changing the shape without changing the data.

```
# 1D array of 6 elements -> 2x3 Matrix
arr = np.arange(6)
arr.reshape(2, 3)
```

---

## 4. Common Pitfalls

### 1. Copy vs. View

Slicing returns a **View** (reference), not a copy. Modifying the slice modifies the original!

```
subset = arr[0:2]
subset[:] = 0 # Modifies 'arr' too!

# Fix: Use .copy()
subset = arr[0:2].copy()
```

### 2. Shape Mismatch

Broadcasting only works if dimensions are compatible.

- Compatible: `(3, 2)` and `(2,)` (matches last dim)
- Incompatible: `(3, 2)` and `(3,)`

Always check `arr.shape` if you get a ValueError!