

Numerical Python (NumPy) is a powerful library for numerical computations in Python.

Table of Contents



- [1. Installing and Importing NumPy](#)
- [2. Creating Arrays](#)
 - [1D Array](#)
 - [2D Array \(Matrix\)](#)
 - [Zeros, Ones, and Identity Matrix](#)
 - [Range and Linspace](#)
- [3. Array Properties](#)
- [4. Reshaping and Flattening Arrays](#)
- [5. Indexing and Slicing](#)
 - [Basic Indexing](#)
 - [Slicing](#)
 - [Conditional Indexing](#)
- [6. Mathematical Operations](#)
 - [Element-wise Operations](#)
 - [Aggregate Functions](#)
- [7. Matrix Operations](#)
- [8. Random Numbers](#)
- [9. Copying and Broadcasting](#)
 - [Copying Arrays](#)
 - [Broadcasting](#)
- [10. Useful NumPy Functions](#)
- [11. Saving and Loading Data](#)
- [12. Handling Missing Values](#)
- [13. Boolean Operations](#)
- [14. Performance Tips](#)
 - [Tips for Learning NumPy](#)

1. Installing and Importing NumPy

```
pip install numpy  
  
import numpy as np
```

2. Creating Arrays

1D Array

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

2D Array (Matrix)

```
arr2d = np.array([[1, 2], [3, 4]])
```

Zeros, Ones, and Identity Matrix

```
np.zeros((3, 3)) # 3x3 array of zeros  
np.ones((2, 4))  # 2x4 array of ones  
np.eye(3)        # 3x3 Identity matrix
```

Range and Linspace

```
np.arange(0, 10, 2) # [0, 2, 4, 6, 8]  
np.linspace(0, 5, 10) # 10 points between 0 and 5
```

3. Array Properties

```
arr.shape      # Shape (rows, cols)  
arr.size       # Total number of elements  
arr.ndim       # Number of dimensions  
arr.dtype      # Data type of elements
```

4. Reshaping and Flattening Arrays

```
arr.reshape(2, 2)  # Reshape to 2x2
arr.flatten()      # Flatten to 1D array
```

5. Indexing and Slicing

Basic Indexing

```
arr[0]             # First element
arr[-1]            # Last element
arr2d[1, 1]        # Element at row 1, column 1
```

Slicing

```
arr[1:3]           # Elements from index 1 to 2
arr[:2]            # First two elements
arr[::2]           # Every second element
```

Conditional Indexing

```
arr[arr > 2]       # Filter elements greater than 2
```

6. Mathematical Operations

Element-wise Operations

```
arr + 2            # Add 2 to each element
arr * 3            # Multiply each element by 3
np.sqrt(arr)       # Square root of elements
np.exp(arr)        # Exponential of elements
```

Aggregate Functions

```
np.sum(arr)      # Sum of all elements
np.mean(arr)     # Mean
np.min(arr)      # Minimum
np.max(arr)      # Maximum
np.std(arr)      # Standard deviation
```

7. Matrix Operations

```
np.dot(arr1, arr2) # Matrix multiplication
np.transpose(arr2d) # Transpose matrix
np.linalg.inv(arr2d) # Inverse of matrix
```

8. Random Numbers

```
np.random.rand(3, 3) # Uniform distribution
np.random.randn(3, 3) # Standard normal distribution
np.random.randint(0, 10, (2, 2)) # Random integers between 0 and 10
```

9. Copying and Broadcasting

Copying Arrays

```
b = arr.copy() # Independent copy
```

Broadcasting

```
arr + np.array([1, 2, 3, 4]) # Broadcasting addition
```

10. Useful NumPy Functions

Function	Description
<code>np.sum(arr)</code>	Sum of elements
<code>np.prod(arr)</code>	Product of elements
<code>np.cumsum(arr)</code>	Cumulative sum
<code>np.cumprod(arr)</code>	Cumulative product
<code>np.sort(arr)</code>	Sort elements
<code>np.argsort(arr)</code>	Indices of sorted elements
<code>np.unique(arr)</code>	Unique elements
<code>np.argmax(arr)</code>	Index of max value
<code>np.argmin(arr)</code>	Index of min value

11. Saving and Loading Data

```
np.save('array.npy', arr)          # Save array to file
loaded_arr = np.load('array.npy')  # Load array from file
```

12. Handling Missing Values

```
np.nan                # Not-a-Number (NaN)
np.isnan(arr)         # Check for NaNs
np.nanmean(arr)       # Mean ignoring NaNs
np.nansum(arr)        # Sum ignoring NaNs
```

13. Boolean Operations

```
np.all(arr > 0)        # True if all elements > 0
np.any(arr < 0)        # True if any element < 0
```

14. Performance Tips

- Use Vectorized Operations - Faster than loops.
 - Avoid Copying Arrays Unnecessarily - Use views instead.
 - Preallocate Memory - Use `np.empty` or `np.zeros` to initialize large arrays.
-

Tips for Learning NumPy

- Practice Array Manipulation - Experiment with slicing, reshaping, and broadcasting.
- Explore NumPy Documentation - Learn more about advanced functions and optimizations.
- Apply to Real Projects - Use NumPy for data analysis, image processing, and scientific computing.

NumPy is the foundation of data science and machine learning in Python