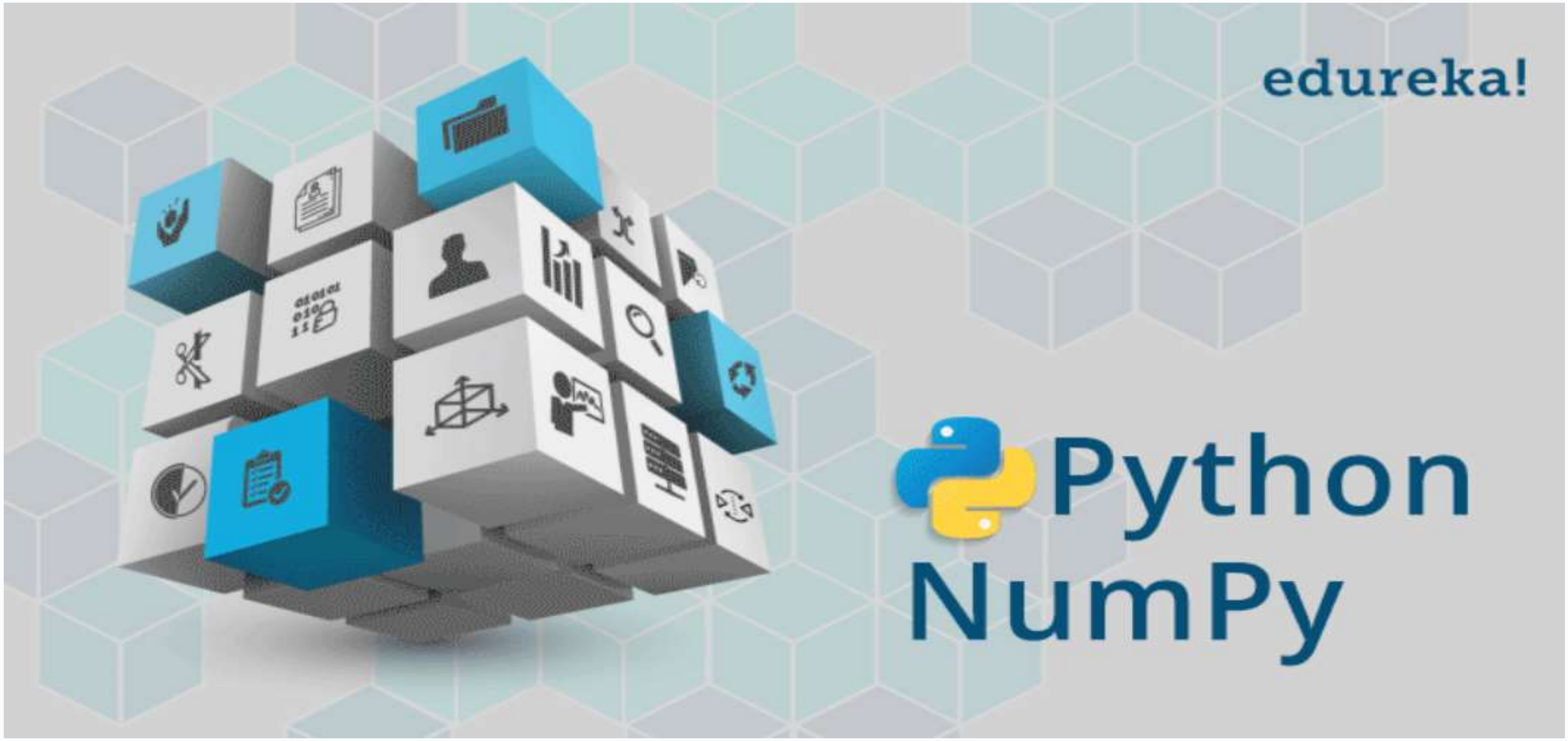


NUMPY IN PYTHON

Name: Tanisha Bharat kumar Patel
DIV:E
EN:24020318 30033
Roll no :73

NumPy: The Foundation of Numerical Computing in Python

Thiз preзentation introduceз NumPy, a fundamental package for numerical computing in Python. NumPy iз eззential for efficient numerical operationз, making it a cornerзtone of data зcience. Thiз overview iз tailored for Python developerз, data зcientiзtз, and зtudentз зeeking to leverage the power of NumPy.

NumPy provideз high-performance multidimenзional array objectз and toolз for working with them. Aз of November 2023, the lateзt verзion iз 1.26. Thiз preзentation will cover arrayз, data typeз, array creation, math, broadcaзting, reзhaping, and зome common applicationз.

NumPy Basics: Arrays and Data Structures

What is a NumPy Array?

A NumPy array, or ndarray, is a core data structure in the NumPy library. Unlike Python lists, NumPy arrays are homogeneous, meaning they contain elements of the same data type.

NumPy arrays use contiguous memory blocks, which enables efficient access and manipulation of data. They are multidimensional and have a fixed size upon creation.

Creating Arrays

Arrays can be created from Python lists or tuples. Here are a few quick examples:

1D Array: `arr = np.array([1, 2, 3, 4, 5])`
2D Array: `arr = np.array([[1, 2, 3], [4, 5, 6]])`

Why Use NumPy Arrays?

NumPy arrays are significantly more efficient for numerical operations than Python lists. In many cases, NumPy arrays can be up to 50x faster than lists, due to optimized routines.

NumPy Data Types

Understanding Data Types

In NumPy, each array has a data type (dtype) that determines the type of elements stored in the array. This is crucial for memory efficiency and performance.

Common Data Types

NumPy supports various data types, including `int`, `float`, `complex`, `bool`, `string`, and `object`. Each type has different memory requirements and computational properties.

Specifying Data Types

You can specify the data type during array creation using the `dtype` parameter. For example: `arr = np.array([1, 2, 3], dtype=np.float64)`.

Integer types include `int8`, `int16`, `int32`, and `int64`. Floating-point types include `float16`, `float32`, and `float64`. Choosing the right data type optimizes memory usage and performance.

Array Creation Routines

np.array()

Creates an array from a list or tuple. This is the most basic way to create a NumPy array by manually specifying the elements.

np.zeros()

Creates an array filled with zeros. Useful for initializing arrays when you know the shape but not the values. Example: `np.zeros((2, 3))`.

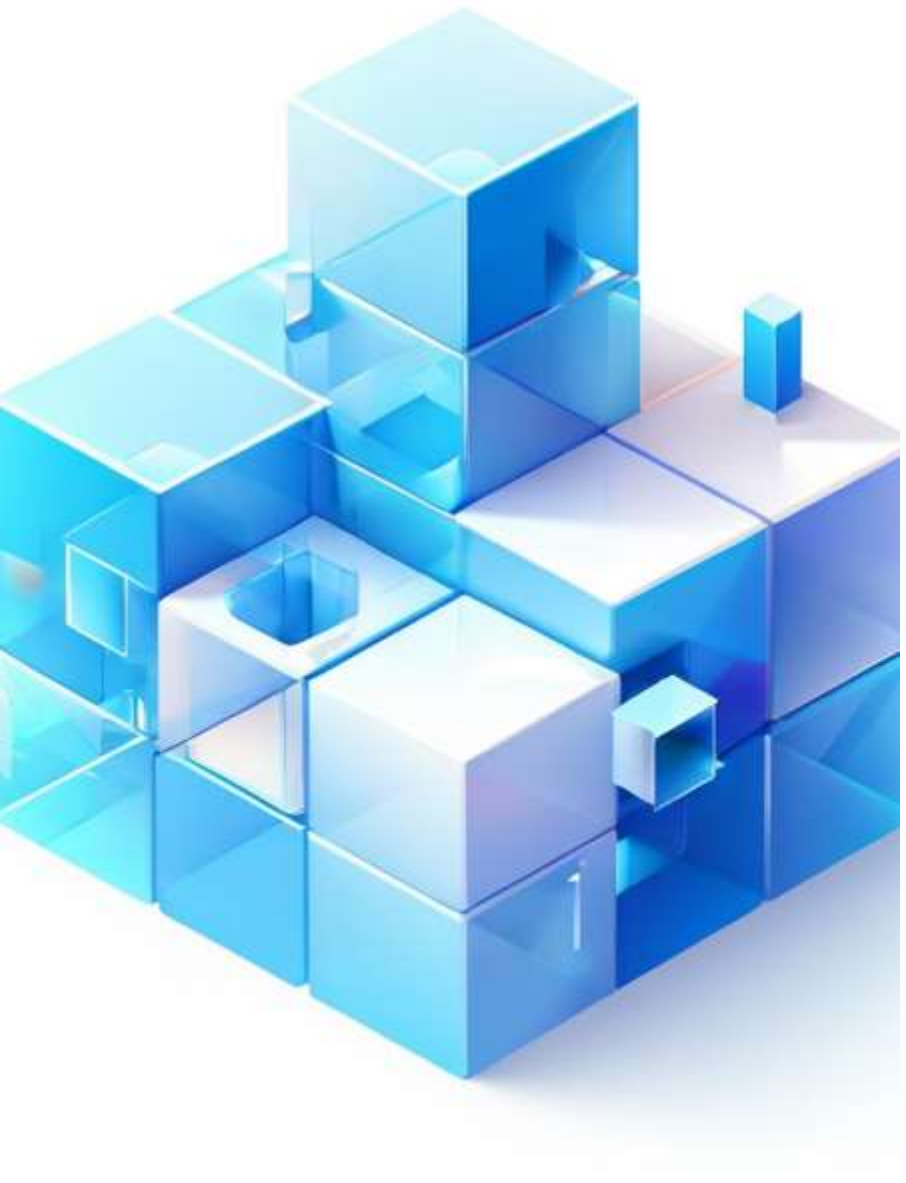
np.ones()

Creates an array filled with ones. Similar to `np.zeros()`, but initializes the array with ones. Example: `np.ones((2, 3))`.

np.arange()

Creates an array with a range of values. It is similar to Python's `range()` function but returns an array. Example: `np.arange(0, 10, 2)`.

Other methods include `np.empty()` to create an uninitialized array, `np.linspace()` to create an array with evenly spaced values over an interval, and `np.random.rand()` to create an array with random values between 0 and 1.



Array Indexing and Slicing

Basic Indexing

Access individual array elements using their index. For example: `arr` or `arr[1, 2]`.

Slicing

Extract subarrays by specifying a range of indices. For example: `arr[1:5]` or `arr[:, 1]`.

Boolean Indexing

Select elements based on a boolean condition. For example: `arr[arr > 5]`.

Integer array indexing allows you to select elements using an array of indices, e.g., `arr[[0, 2, 4]]`. Indexing and slicing can also be used to modify array elements. For example: `arr[0:3] = 0`.

Array Math: Element-wise Operations

1

Element-wise Operations

Perform addition, subtraction, multiplication, and division on corresponding elements of arrays. Examples: `arr1 + arr2`, `arr1 * 2`.

2

Universal Functions (ufuncs)

Apply mathematical functions to each element of an array. Examples: `np.sin()`, `np.cos()`, `np.exp()`, `np.log()`.

3

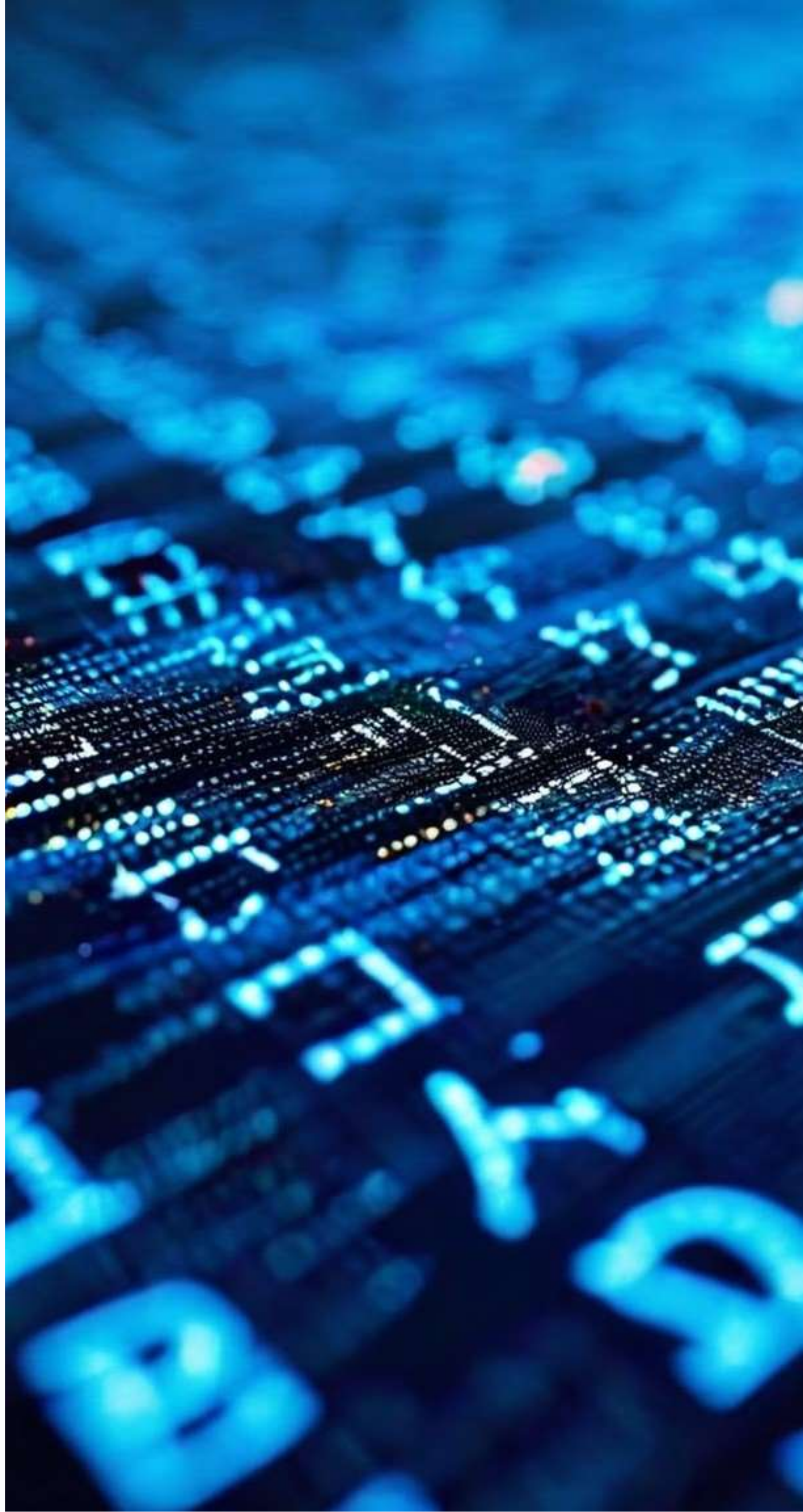
Comparison Operators

Compare elements of arrays and return boolean arrays. Examples: `arr1 > arr2`, `arr1 == arr2`.

4

Logical Operations

Perform logical operations on boolean arrays. Examples: `np.logical_and()`, `np.logical_or()`.



Broadcasting: Operations on Arrays with Different Shapes

Understanding Broadcasting

NumPy can perform operations on arrays with different shapes if their dimensions are compatible.

How it Works

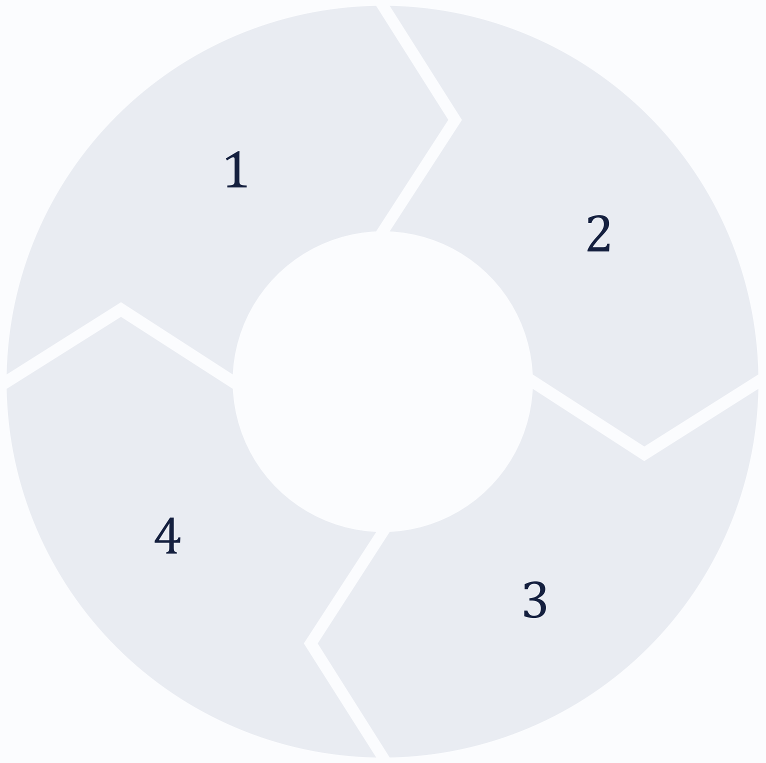
Dimensions are compatible if they are equal or if one of them is 1. NumPy automatically expands the smaller array to match the shape of the larger array.

Use Cases

Simplifies code and improves performance. For example, adding a scalar to an array, or adding a 1D array to a 2D array.

Example

Centering data by subtracting the mean. NumPy broadcasts the mean across rows or columns to align with dimensions.



Reshaping Arrays: Changing Array Dimensions

1

Reshape

Change the shape of an array using `reshape()`. Example: `arr.reshape((2, 5))`.

2

Flatten

Flatten arrays using `flatten()` and `ravel()`. Example: `arr.flatten()`.

3

Transpose

Transpose arrays using `transpose()` or `.T`.

Use `np.newaxis` to add a dimension and `np.squeeze()` to remove single-dimensional entries. Reshaping and transposing are crucial for aligning data for mathematical operations and analysis.

NumPy Use Cases: Real-World Applications

1

Image Processing

Manipulate pixel data using NumPy arrays.

2

Machine Learning

Use for data preprocessing and feature extraction.

3

Financial Analysis

Time series analysis and statistical modeling.

NumPy is also used in signal processing for Fourier transforms and filtering, and in scientific computing for solving linear equations and simulations. Image manipulation includes resizing and color adjustments.

c

NUMPY PAKAGE DOWNLOAD IN PYTON ALL SOFTWARWARE

✓ **Step 1: Install Python**

If you don't already have Python installed:

1.Go to the official Python website: https://www.python.org/downloads/

2.Download the latest version (choose Windows, macOS, or Linux based on your system).

3.During installation, **check the box that says "Add Python to PATH"**.

4.Finish installing Python

✓ **Step 2: Install NumPy**

Once Python is installed, open your terminal or command prompt and run:

❑ pip install numpy

✓ Optionally Use an IDE or Code Editor

To make coding easier, you can use one of these popular Python environments:

Software	Download Link	Notes
PyCharm	https://www.jetbrains.com/pycharm/	Full-featured Python IDE
VS Code	https://code.visualstudio.com/	Lightweight and customizable
Anaconda	https://www.anaconda.com/	Comes with NumPy & many packages pre-installed

Step 3: Test NumPy Installation

1. Create a Python file (e.g., test_numpy.py)

or open a Python shell, then type:

2.import num py as np arr = np.array([1, 2, 3])

print(arr)

3.If it prints [1 2 3] — ✓ NumPy is working!

Conclusion: The Power of NumPy

1

Key Features

High-performance multidimensional arrays

2

Benefits

Efficient numerical computations

3

Ecosystem

Foundation for data science in Python

NumPy9s key features and benefits make it an indispensable tool for anyone working with numerical data in Python. Further learning resources include the NumPy documentation at `numpy.org/doc/stable/` and the SciPy library at `scipy.org`, which builds on NumPy.