NumPy for Data Science

Master numerical computing with Python

What is NumPy?

NumPy (Numerical Python) is the fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently.

Why Use NumPy?

- Faster than traditional Python lists for numerical operations
- Memory efficient - stores data more compactly
- Provides vectorized operations for clean, fast code
- Foundation for other data science libraries (Pandas, SciPy, Scikit-learn)
- Supports broadcasting for operations on different sized arrays

Installation

```
pip install numpy
```

Creating NumPy Arrays

There are several ways to create NumPy arrays:

```
import numpy as np

# From a Python list
arr1 = np.array([1, 2, 3, 4, 5])

# Array of zeros
zeros = np.zeros((3, 4))  # 3 rows, 4 columns

# Array of ones
ones = np.ones((2, 3))

# Range of values
range_arr = np.arange(0, 10, 2)  # [0, 2, 4, 6, 8]

# Evenly spaced values
linspace = np.linspace(0, 1, 5)  # 5 values from 0 to 1
```

Array Operations

NumPy supports element-wise operations:

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

# Element-wise operations
print(arr + 10)      # [11 12 13 14 15]
print(arr * 2)       # [2 4 6 8 10]
print(arr ** 2)      # [1 4 9 16 25]

# Array operations
print(arr.sum())     # 15
print(arr.mean())    # 3.0
print(arr.std())     # Standard deviation
print(arr.max())     # 5
print(arr.min())     # 1
```

Indexing and Slicing

```
arr = np.array([10, 20, 30, 40, 50])

# Basic indexing
print(arr[0])        # 10
print(arr[-1])       # 50

# Slicing
print(arr[1:4])      # [20 30 40]
print(arr[::2])      # [10 30 50]

# 2D array indexing
arr2d = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2d[0, 1])   # 2
print(arr2d[:, 1])   # [2 5] (all rows, column 1)
```

Best Practices

- Always use vectorized operations instead of loops
- Understand broadcasting rules for efficient operations
- Use appropriate data types (int32, float64) to save memory
- Leverage built-in NumPy functions instead of writing your own
- Use axis parameter in aggregation functions for multi-dimensional arrays