NumPy Notes

Introduction to NumPy

NumPy (Numerical Python) is a fundamental library in Python for scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays efficiently. NumPy is widely used in data science, machine learning, and numerical computations.

Installing and Importing NumPy

To use NumPy, you need to install it first. You can install NumPy using pip:

```
!pip install numpy
```

This command installs NumPy in your Python environment.

After installation, import NumPy into your Python script:

```
import numpy as np
```

This command imports the NumPy library and assigns it the alias np for easier usage.

Creating NumPy Arrays

1D, 2D, and 3D Arrays

1D Array (Vector)

A one-dimensional array is a simple array with elements arranged in a single row.

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

print(a)
```

Explanation: The np.array() function is used to create an array in NumPy.

2D Array (Matrix)

A two-dimensional array is a matrix with rows and columns.

```
b = np.array([[1, 2, 3], [4, 5, 6]])

print(b)
```

Explanation: This creates a 2D array (matrix) where each inner list represents a row.

3D Array (Matrix of Matrices)

A three-dimensional array consists of multiple 2D arrays.

```
c = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print(c)
```

Explanation: Here, each 2D array is stacked inside a larger 3D structure.

Creating Arrays with Specific Values

Arrays with Zeroes and Ones

```
zero = np.zeros((1,3))  # 1x3 array of zeroes

one = np.ones((1,3))    # 1x3 array of ones

print(zero)

print(one)
```

Explanation:

- np.zeros(shape): Creates an array of the given shape filled with zeros.
- np.ones(shape): Creates an array filled with ones.

Array with a Specific Value

```
arr = np.full((3,3), 7)  # 3x3 array filled with 7

print(arr)
```

Explanation: np.full(shape, value) creates an array of a specified shape, filled with the given value.

Identity Matrix

```
identity = np.eye(4)  # 4x4 identity matrix

print(identity)
```

Explanation: np.eye(n) creates an identity matrix of size n x n with ones on the diagonal and zeros elsewhere.

Random Number Generation

Uniform Distribution (0,1)

```
rand_array = np.random.rand(4,4)

print(rand_array)
```

Explanation: np.random.rand(shape) generates random values from a uniform distribution between 0 and 1.

Standard Normal Distribution

```
rand_normal = np.random.randn(3,3)

print(rand_normal)
```

Explanation: np.random.randn(shape) generates random values from a normal distribution with mean 0 and variance 1.

Random Integers within a Range

```
rand_ints = np.random.randint(10, 100, (2,3))

print(rand_ints)
```

Explanation: np.random.randint(low, high, shape) generates random integers between low and high with the specified shape.

Creating Sequences and Linearly Spaced Values

Using arange()

```
arr1 = np.arange(0, 10, 2)  # Generates numbers from 0 to 10 with step size 2

print(arr1)
```

Explanation: np.arange(start, stop, step) generates evenly spaced values within a given range.

Using linspace()

```
arr2 = np.linspace(0, 10, 100)  # Generates 100 linearly spaced values from 0 to 10

print(arr2)
```

Explanation: np.linspace(start, stop, num) creates an array with num evenly spaced values between start and stop.

Array Shape, Reshaping, and Transpose

Reshaping an Array

```
b = a.reshape(3,2)  # Changing shape

print(b)
```

Explanation: reshape(new_shape) changes the shape of an array without changing its data.

Flattening an Array

```
c = b.ravel()  # Converts multi-dimensional array to 1D
print(c)
```

Explanation: ravel() returns a 1D version of the array.

Transpose of a Matrix

```
transpose = arr.T  # Transposes the array
print(transpose)
```

Explanation: T swaps rows and columns in an array.

Mathematical and Statistical Operations

```
print(arr + 10)  # Adding 10 to each element
print(arr * 2)   # Multiplication with scalar
print(arr ** 2)  # Squaring elements
```

Aggregate Functions

```
print(arr.sum())  # Sum of all elements
print(arr.mean())  # Mean of elements
print(arr.max(axis=0))  # Max in each column
print(arr.max(axis=1))  # Max in each row
print(arr.min(axis=0))  # Min in each column
print(arr.min(axis=1))  # Min in each row
print(np.sort(arr))  # Sorting array
```

Joining and Splitting Arrays

Concatenation

```
arr_combined = np.concatenate((arr1, arr2))
print(arr_combined)
```

Explanation: np.concatenate(arrays, axis) joins multiple arrays along the specified axis.

Stacking

```
stacked = np.stack((arr1, arr2))

stacked_axis = np.stack((arr1, arr2), axis=1)

print(stacked)
```

Explanation: np.stack(arrays, axis) stacks arrays along a new dimension.

Splitting

```
split_arr = np.split(arr, 3)  # Splitting array into three parts

print(split_arr)
```

Explanation: np.split(array, sections) divides an array into equal parts.

Conclusion

NumPy is a powerful library that simplifies numerical computations with efficient array operations. It provides functions for array creation, reshaping, mathematical operations, and data manipulation, making it an essential tool for data scientists and engineers.