**Introduction:**

Linear algebra provides powerful tools for manipulating and analyzing matrices, and three essential operations are determinant, inverse, and transpose. In this article, we'll explore these operations and demonstrate how to perform them efficiently using NumPy, a popular numerical computing library in Python.

**1. Determinant:**

The determinant of a square matrix is a scalar value that provides essential information about the matrix's properties. It determines whether the matrix is invertible and affects its behavior in linear transformations.

The determinant of a square matrix $A$ is a scalar value denoted as $|A|$ or $\det(A)$. For a 2×2 matrix:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The determinant is calculated as:

$$|A| = ad - bc$$

For higher-dimensional matrices, you can use the cofactor expansion or Laplace expansion along any row or column to compute the determinant.

Determinant

**Example 1:**

```
import numpy as np

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

determinant = np.linalg.det(A)

print(determinant)
```

**Output**

-2.0000000000000004

**Example 2:**

```
import numpy as np

B = np.array([[1,2,3], [4,5,6], [7,8,9]])

determinant = np.linalg.det(B)

print(determinant)
```

**Output**

0.0

**2. Transpose:**

The transpose of a matrix is obtained by swapping its rows and columns. It reflects the matrix across its main diagonal. Transposition is essential in various mathematical operations and transformations.

The transpose of a matrix $A$ is denoted as $A^T$. It is obtained by swapping the rows and columns of the original matrix. For a matrix $A$ with dimensions $m \times n$, its transpose $A^T$ will have dimensions $n \times m$. Mathematically:

$$(A^T)_{ij} = A_{ji}$$

Transpose

**Example 1:**

```
import numpy as np

B = np.array([[1,2,3], [4,5,6], [7,8,9]])

Transpose = np.transpose(B)

print(Transpose)
```

**Output**

```
[[1 4 7]

 [2 5 8]

 [3 6 9]]
```

**Example 2:**

```
import numpy as np

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

Transpose = np.transpose(C)

print(Transpose)
```

**Output**

```
[[1 4]
```

[2 5]

[3 6]]

**3.Identity Matrix**

**What is an Identity Matrix?**

An identity matrix is a square matrix in which all the elements on the main diagonal (from top-left to bottom-right) are 1, and all the other elements are 0. It is denoted by $I_n$, where n represents the order (size) of the matrix.

For example, the 3×3 identity matrix looks like this:

$I3=[[100010001]]$

**4. Inverse:**

The inverse of a square matrix *A* is another matrix denoted as $A^{-1}$ , such that the product of $A$ and $A^{-1}$ yields the identity matrix. It is essential for solving systems of linear equations and various transformations.

The inverse of a square matrix $A$, denoted as $A^{-1}$, is a matrix such that their product is the identity matrix $I$:

$$A \cdot A^{-1} = A^{-1} \cdot A = I$$

For a 2×2 matrix:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

The inverse of $A$ is calculated as:

$$A^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

For higher-dimensional matrices, you can use techniques like Gaussian elimination or matrix adjugate to compute the inverse.

Inverse

**Example 1:**

```
import numpy as np

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

Inverse = np.linalg.inv(A)

print(Inverse)
```

**Output**

[[-2.  1. ]

 [ 1.5 -0.5]]

**Conclusion:**

Understanding determinant, inverse, and transpose operations is crucial in linear algebra and various fields such as physics, engineering, and machine learning. With NumPy, performing these operations becomes efficient and straightforward, enabling practitioners to analyze and manipulate matrices effectively.