In [1]:
```
import numpy as np
```

In [2]:
```
d1 = np.array([1, 2, 3, 4, 5])
print(d1)
```

```
[1 2 3 4 5]
```

In [15]:
```
d2 = np.array([[1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]])
print(d2)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

In [17]:
```
d3= np.array([
    [[1, 2, 3],
    [4, 5, 6]],
    [[7, 8, 9],
    [10, 11, 12]]])
print(d3)
```

```
[[[ 1  2  3]
  [ 4  5  6]]

 [[ 7  8  9]
  [10 11 12]]]
```

In [18]:
```
d4 = np.array([[[[1, 2],
    [3, 4]],
    [[5, 6],
    [7, 8]]],
    [[[9, 10],
    [11, 12]],
    [[13, 14],
    [15, 16]]]])
print(d4)
```

```
[[[[ 1  2]
   [ 3  4]]

  [[ 5  6]
   [ 7  8]]]


 [[[ 9 10]
   [11 12]]

  [[13 14]
   [15 16]]]]
```

In [6]:
```
d5 = np.array([
    [
        [
            [
                [
                    [1, 2],
                    [3, 4]
                ],
                [
                    [5, 6],
                    [7, 8]
                ]
            ],
            [
                [
                    [9, 10],
                    [11, 12]
                ],
                [
                    [13, 14],
                    [15, 16]
                ]
            ]
        ],
        [
            [
                [
                    [17, 18],
                    [19, 20]
                ],
                [
                    [21, 22],
                    [23, 24]
                ]
            ],
            [
                [
                    [25, 26],
                    [27, 28]
                ],
                [
                    [29, 30],
                    [31, 32]
                ]
            ]
        ]
    ]
])
print(d5)
```

```
[[[[[[ 1  2]
      [ 3  4]]

     [[ 5  6]
      [ 7  8]]]


    [[[ 9 10]
      [11 12]]

     [[13 14]
      [15 16]]]]



   [[[[17 18]
      [19 20]]

     [[21 22]
      [23 24]]]


    [[[25 26]
      [27 28]]

     [[29 30]
      [31 32]]]]]]
```

In [7]:
```
da1 = np.array([1, 2, 3, 4, 5])
da2 = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
da3 = np.array([
    [
        [1, 2, 3],
        [4, 5, 6]
    ],
    [
        [7, 8, 9],
        [10, 11, 12]
    ]
])
da4 = np.array([
    [
        [
            [1, 2],
            [3, 4]
        ],
        [
            [5, 6],
            [7, 8]
        ]
    ],
    [
        [
            [9, 10],
            [11, 12]
        ],
        [
            [13, 14],
            [15, 16]
        ]
    ]
])
da5 = np.array([
    [
        [
            [
                [
                    [1, 2],
                    [3, 4]
                ],
                [
                    [5, 6],
                    [7, 8]
                ]
            ],
            [
                [
                    [9, 10],
                    [11, 12]
                ],
                [
```

```
                    [13, 14],
                    [15, 16]
                ]
            ]
        ],
        [
            [
                [
                    [17, 18],
                    [19, 20]
                ],
                [
                    [21, 22],
                    [23, 24]
                ]
            ],
            [
                [
                    [25, 26],
                    [27, 28]
                ],
                [
                    [29, 30],
                    [31, 32]
                ]
            ]
        ]
    ]
])

#determinant_1d = np.linalg.det(da1)
determinant_2d = np.linalg.det(da2)
#determinant_3d = np.linalg.det(da3)
determinant_4d = np.linalg.det(da4)
determinant_5d = np.linalg.det(da5)

#print("Determinant of 1-D Matrix (Vector):")
#print(determinant_1d)

print("\nDeterminant of 2-D Matrix:")
print(determinant_2d)

#print("\nDeterminant of 3-D Matrix (Tensor):")
#print(determinant_3)

print("\nDeterminant of 4-D Matrix:")
print(determinant_4d)

print("\nDeterminant of 5-D Matrix:")
print(determinant_5d)
```

```
Determinant of 2-D Matrix:
0.0

Determinant of 4-D Matrix:
[[-2. -2.]
 [-2. -2.]]

Determinant of 5-D Matrix:
[[[[-2. -2.]
   [-2. -2.]]

  [[-2. -2.]
   [-2. -2.]]]]
```

In [8]:
```
matrix_1d = np.array([1, 2, 3, 4, 5])
matrix_2d = np.array([
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
])
matrix_3d = np.array([
    [
        [1, 2, 3],
        [4, 5, 6]
    ],
    [
        [7, 8, 9],
        [10, 11, 12]
    ]
])
matrix_4d = np.array([
    [
        [
            [1, 2],
            [3, 4]
        ],
        [
            [5, 6],
            [7, 8]
        ]
    ],
    [
        [
            [9, 10],
            [11, 12]
        ],
        [
            [13, 14],
            [15, 16]
        ]
    ]
])
matrix_5d = np.array([
    [
        [
            [
                [
                    [1, 2],
                    [3, 4]
                ],
                [
                    [5, 6],
                    [7, 8]
                ]
            ],
            [
                [
                    [9, 10],
                    [11, 12]
                ],
                [
```

```
                    [13, 14],
                    [15, 16]
                ]
            ]
        ],
        [
            [
                [
                    [17, 18],
                    [19, 20]
                ],
                [
                    [21, 22],
                    [23, 24]
                ]
            ],
            [
                [
                    [25, 26],
                    [27, 28]
                ],
                [
                    [29, 30],
                    [31, 32]
                ]
            ]
        ]
    ]
])
try:
    inverse_2d = np.linalg.inv(matrix_2d)
    print("Inverse of 2-D Matrix:")
    print(inverse_2d)
except np.linalg.LinAlgError:
    print("The 2-D matrix is not invertible.")

try:
    inverse_3d = np.linalg.inv(matrix_3d)
    print("\nInverse of 3-D Matrix (Tensor):")
    print(inverse_3d)
except np.linalg.LinAlgError:
    print("The 3-D matrix is not invertible.")

try:
    inverse_4d = np.linalg.inv(matrix_4d)
    print("\nInverse of 4-D Matrix:")
    print(inverse_4d)
except np.linalg.LinAlgError:
    print("The 4-D matrix is not invertible.")

try:
    inverse_5d = np.linalg.inv(matrix_5d)
    print("\nInverse of 5-D Matrix:")
    print(inverse_5d)
except np.linalg.LinAlgError:
    print("The 5-D matrix is not invertible.")
```

```
The 2-D matrix is not invertible.
The 3-D matrix is not invertible.

Inverse of 4-D Matrix:
[[[[-2.   1. ]
   [ 1.5 -0.5]]

  [[-4.   3. ]
   [ 3.5 -2.5]]]


 [[[-6.   5. ]
   [ 5.5 -4.5]]

  [[-8.   7. ]
   [ 7.5 -6.5]]]]

Inverse of 5-D Matrix:
[[[[[[ -2.    1. ]
     [  1.5  -0.5]]

    [[ -4.    3. ]
     [  3.5  -2.5]]]


   [[[ -6.    5. ]
     [  5.5  -4.5]]

    [[ -8.    7. ]
     [  7.5  -6.5]]]]



  [[[[-10.    9. ]
     [  9.5  -8.5]]

    [[-12.   11. ]
     [ 11.5 -10.5]]]


   [[[-14.   13. ]
     [ 13.5 -12.5]]

    [[-16.   15. ]
     [ 15.5 -14.5]]]]]]
```

In [9]:
```
A = np.array([[3, 1],
              [1, 2]])
eigenvalues, eigenvectors = np.linalg.eig(A)
print("Eigenvalues:")
print(eigenvalues)
print("\nEigenvectors:")
print(eigenvectors)
```

```
Eigenvalues:
[3.61803399 1.38196601]

Eigenvectors:
[[ 0.85065081 -0.52573111]
 [ 0.52573111  0.85065081]]
```

In [10]:
```
matrix_A = np.array([[3, 1],
                     [1, 2]])

matrix_B = np.array([[5, 2],
                     [2, 8]])

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

matrix_D = np.array([[0, 1],
                     [1, 0]])

matrix_E = np.array([[6, 0],
                     [0, 3]])

def calculate_properties(matrix):
    rank = np.linalg.matrix_rank(matrix)
    diagonal_elements = np.diag(matrix)
    trace = np.trace(matrix)
    return rank, diagonal_elements, trace
matrices = [matrix_A, matrix_B, matrix_C, matrix_D, matrix_E]

for i, matrix in enumerate(matrices, start=1):
    rank, diagonal_elements, trace = calculate_properties(matrix)

    print(f"Matrix {i}:")
    print("Rank:", rank)
    print("Diagonal Elements:", diagonal_elements)
    print("Trace:", trace)
    print()
```

```
Matrix 1:
Rank: 2
Diagonal Elements: [3 2]
Trace: 5

Matrix 2:
Rank: 2
Diagonal Elements: [5 8]
Trace: 13

Matrix 3:
Rank: 2
Diagonal Elements: [4 3]
Trace: 7

Matrix 4:
Rank: 2
Diagonal Elements: [0 0]
Trace: 0

Matrix 5:
Rank: 2
Diagonal Elements: [6 3]
Trace: 9
```

In [11]:
```
from scipy import stats

def generate_random_dataset(size):
    return np.random.randint(1, 10, size)

datasets = [generate_random_dataset(10) for _ in range(5)]

for i, data in enumerate(datasets, start=1):
    print(f"Dataset {i}: {data}")
    print(f"Mean: {np.mean(data)}")
    print(f"Median: {np.median(data)}")

    try:
        mode_result = stats.mode(data)
        print(f"Mode: {mode_result.mode} (with a count of {mode_result.count[0
    except stats.StatisticError:
        print("No unique mode")

    print()
```

```
Dataset 1: [9 8 8 7 3 7 2 3 7 8]
Mean: 6.2
Median: 7.0
Mode: [7] (with a count of 3)

Dataset 2: [9 7 6 4 4 5 5 8 2 5]
Mean: 5.5
Median: 5.0
Mode: [5] (with a count of 3)

Dataset 3: [3 9 2 9 9 3 6 2 3 9]
Mean: 5.5
Median: 4.5
Mode: [9] (with a count of 4)

Dataset 4: [1 9 8 3 6 4 7 3 7 9]
Mean: 5.7
Median: 6.5
Mode: [3] (with a count of 2)

Dataset 5: [6 5 2 3 1 2 3 9 3 3]
Mean: 3.7
Median: 3.0
Mode: [3] (with a count of 4)
```

In [13]:
```
import scipy.stats
def generate_random_datasets(size):
    data1 = np.random.randint(1, 100, size)
    data2 = np.random.randint(1, 100, size)
    return data1, data2
data1, data2 = generate_random_datasets(20)

covariance = np.cov(data1, data2)[0, 1]
pearson_corr = np.corrcoef(data1, data2)[0, 1]

spearman_corr, _ = scipy.stats.spearmanr(data1, data2)

print("Data 1:", data1)
print("Data 2:", data2)
print("Covariance:", covariance)
print("Pearson Correlation:", pearson_corr)
print("Spearman Correlation:", spearman_corr)
```

```
Data 1: [58 62 56 59 82 58  3 23 44 99 19 84 39 55 76 80 13 26  7 44]
Data 2: [71 32 90 61 85 94 83 76 43 98 62 49 85 21 23 81 93 89 32 14]
Covariance: -8.299999999999997
Pearson Correlation: -0.010828280350380168
Spearman Correlation: 0.021084337349397592
```

In [ ]:
```
```