Q1. Create two 3×3 matrices using the random function in Numpy and perform the following operations. è Product (prod) è Multiplication (multiply) è Dot Product (dot)

In [1]:

```
import numpy as np
```

In [3]:

```
matrix1 = np.random.rand(3, 3)
matrix2 = np.random.rand(3, 3)

# Perform the operations
product_result = np.prod(matrix1)
multiply_result = np.multiply(matrix1, matrix2)
dot_product_result = np.dot(matrix1, matrix2)

# Print the results
print("Matrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nProduct (prod) Result:")
print(product_result)
print("\nMultiplication (multiply) Result:")
print(multiply_result)
print("\nDot Product (dot) Result:")
print(dot_product_result)
```

```
Matrix 1:
[[0.2429255  0.93567405 0.46665775]
 [0.34920072 0.26616834 0.8891719 ]
 [0.4438597  0.32223999 0.58974406]]

Matrix 2:
[[9.22396479e-02 3.39025997e-01 6.34029877e-01]
 [8.78661870e-02 4.87342588e-03 9.25117417e-01]
 [3.02606156e-01 8.80544545e-02 4.62970858e-04]]

Product (prod) Result:
0.0007394386540365008

Multiplication (multiply) Result:
[[2.24073630e-02 3.17217826e-01 2.95874953e-01]
 [3.06829359e-02 1.29715168e-03 8.22588411e-01]
 [1.34314677e-01 2.83746662e-02 2.73034313e-04]]

Dot Product (dot) Result:
[[0.24583498 0.12800929 1.01984643]
 [0.32466624 0.19798082 0.46805232]
 [0.24771564 0.20397998 0.57980317]]
```

Q2. Perform the following set operations using the Numpy functions. è Union è Intersection è Set difference è XOR

In [4]:

```
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([3, 4, 5, 6, 7])

# Union (elements that are in either array)
union_result = np.union1d(array1, array2)
```

```
print("Union:", union_result)

# Intersection (elements that are in both arrays)
intersection_result = np.intersect1d(array1, array2)
print("Intersection:", intersection_result)

# Set Difference (elements that are in array1 but not in array2)
set_difference_result = np.setdiff1d(array1, array2)
print("Set Difference (array1 - array2):", set_difference_result)

# XOR (exclusive OR) - elements that are in either array, but not in both
xor_result = np.setxor1d(array1, array2)
print("XOR:", xor_result)
```

```
Union: [1 2 3 4 5 6 7]
Intersection: [3 4 5]
Set Difference (array1 - array2): [1 2]
XOR: [1 2 6 7]
```

Q3. Create a 1D array using Random function and perform the following operations. è Cumulative sum è Cumulative Product è Discrete difference (with n=3) è Find the unique elements from the array

In [7]:

```
array = np.array([2,3,4,5,5,6])  # Change the size (10 in this case) to your desired c

# Cumulative Sum
cumulative_sum = np.cumsum(array)
print("Cumulative Sum:")
print(cumulative_sum)

# Cumulative Product
cumulative_product = np.cumprod(array)
print("\nCumulative Product:")
print(cumulative_product)

# Discrete Difference with n=3
n = 3
discrete_difference = np.diff(array, n=n)
print("\nDiscrete Difference (n=3):")
print(discrete_difference)

# Find Unique Elements
unique_elements = np.unique(array)
print("\nUnique Elements:")
print(unique_elements)
```

```
Cumulative Sum:
[ 2  5  9 14 19 25]

Cumulative Product:
[   2    6   24  120  600 3600]

Discrete Difference (n=3):
[ 0 -1  2]

Unique Elements:
[2 3 4 5 6]
```

Q4. Create two 1D array and perform the Addition using zip(), add() and user defined function (frompyfunc())

In [8]:

```
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([10, 20, 30, 40, 50])

# Addition using zip()
result_zip = [a + b for a, b in zip(array1, array2)]

# Addition using np.add()
result_add = np.add(array1, array2)

# User-defined function using np.frompyfunc()
def custom_add(x, y):
    return x + y

ufunc = np.frompyfunc(custom_add, 2, 1)
result_custom = ufunc(array1, array2)

# Print results
print("Array 1:", array1)
print("Array 2:", array2)
print("\nAddition using zip():", result_zip)
print("Addition using np.add():", result_add)
print("Addition using user-defined function:", result_custom)
```

```
Array 1: [1 2 3 4 5]
Array 2: [10 20 30 40 50]

Addition using zip(): [11, 22, 33, 44, 55]
Addition using np.add(): [11 22 33 44 55]
Addition using user-defined function: [11 22 33 44 55]
```

Q5. Find the LCM (Least Common Multiple) and GCD (Greatest Common Divisor) of an array of elements using reduce().

In [9]:

```
from functools import reduce
import math

# Create an array of elements
elements = np.array([12, 18, 24, 36, 48])

# Calculate GCD (Greatest Common Divisor)
def find_gcd(x, y):
    return math.gcd(x, y)

gcd_result = reduce(find_gcd, elements)

# Calculate LCM (Least Common Multiple)
def find_lcm(x, y):
    return x * y // math.gcd(x, y)

lcm_result = reduce(find_lcm, elements)

# Print the results
print("Array of Elements:", elements)
print("GCD (Greatest Common Divisor):", gcd_result)
print("LCM (Least Common Multiple):", lcm_result)
```

```
Array of Elements: [12 18 24 36 48]
GCD (Greatest Common Divisor): 6
LCM (Least Common Multiple): 144
```

In [ ]:

```
```