In [ ]:
```python
#Mul
import numpy as np

# Generate two random 3x3 matrices
matrix1 = np.random.rand(3, 3)
matrix2 = np.random.rand(3, 3)

# Perform matrix multiplication using np.matmul()
result_matrix = np.matmul(matrix1, matrix2)

print("Matrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nResultant Matrix (Matrix 1 * Matrix 2):")
print(result_matrix)
```

```
Matrix 1:
[[0.71075868 0.18992859 0.51167572]
 [0.70340858 0.74552935 0.53992014]
 [0.35487615 0.9121694  0.68482791]]

Matrix 2:
[[0.22485917 0.24979747 0.16382187]
 [0.84304419 0.8293213  0.86812121]
 [0.30305832 0.1090219  0.13263896]]

Resultant Matrix (Matrix 1 * Matrix 2):
[[0.47500639 0.39084141 0.34918699]
 [0.95030935 0.85285618 0.83405799]
 [1.05633906 0.91978992 0.94084493]]
```

In [ ]:
```python
#Dot
import numpy as np

# Generate two random 3x3 matrices
matrix1 = np.random.rand(3, 3)
matrix2 = np.random.rand(3, 3)

# Perform matrix multiplication (dot product)
result_matrix = np.dot(matrix1, matrix2)

print("Matrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("\nResultant Matrix (Matrix 1 * Matrix 2):")
print(result_matrix)
```

```
Matrix 1:
[[0.58084804 0.22526263 0.63405106]
 [0.77308625 0.77643803 0.21486257]
 [0.15630325 0.20921994 0.15965918]]

Matrix 2:
[[0.43231608 0.95548449 0.14550815]
 [0.09842538 0.28414301 0.4079085 ]
 [0.05682639 0.75848684 0.62702694]]

Resultant Matrix (Matrix 1 * Matrix 2):
[[0.30931235 1.09991748 0.57397176]
 [0.42284869 1.12226179 0.56393065]
 [0.09723782 0.3298931  0.20819659]]
```

In [ ]:
```python
#Q2
import numpy as np

# Create two NumPy arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([3, 4, 5, 6, 7])

# Union
union_result = np.union1d(array1, array2)
print("Union:", union_result)

# Intersection
intersection_result = np.intersect1d(array1, array2)
print("Intersection:", intersection_result)

# Set difference (elements in array1 but not in array2)
difference_result1 = np.setdiff1d(array1, array2)
print("Set Difference (array1 - array2):", difference_result1)

# Set difference (elements in array2 but not in array1)
difference_result2 = np.setdiff1d(array2, array1)
print("Set Difference (array2 - array1):", difference_result2)

# XOR (Symmetric Difference)
xor_result = np.setxor1d(array1, array2)
print("XOR (Symmetric Difference):", xor_result)
```

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

In [ ]:
```python
#Q3
import numpy as np

# Create a random 1D array with 10 elements
random_array = np.random.rand(10)

# Cumulative sum
cumulative_sum = np.cumsum(random_array)
print("Cumulative Sum:", cumulative_sum)

# Cumulative product
cumulative_product = np.cumprod(random_array)
print("Cumulative Product:", cumulative_product)

# Discrete difference with n=3
discrete_diff = np.diff(random_array, n=3)
print("Discrete Difference (n=3):", discrete_diff)

# Find unique elements in the array
unique_elements = np.unique(random_array)
print("Unique Elements:", unique_elements)
```

```
Cumulative Sum: [0.20885331 0.88802188 1.23373469 1.24823308 1.26227292 2.20978721
 2.5901551  2.68496669 3.54651473 3.65749361]
Cumulative Product: [2.08853314e-01 1.41846606e-01 4.90381882e-02 7.10974837e-04
 9.98197646e-06 9.45806533e-06 3.59754438e-06 3.41088894e-07
 2.93864469e-07 3.26127482e-08]
Discrete Difference (n=3): [ 0.80601235  0.32851453  0.60317712 -2.43455383  1.78221093  0.77070267
 -2.56959838]
Unique Elements: [0.01403984 0.01449839 0.09481159 0.11097888 0.20885331 0.34571281
 0.38036789 0.67916856 0.86154804 0.94751429]
```

In [ ]:
```python
#Q4

import numpy as np

# Create two 1D arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([6, 7, 8, 9, 10])

# Addition using zip()
addition_zip = np.array([a + b for a, b in zip(array1, array2)])
print("Addition using zip():", addition_zip)

# Addition using np.add()
addition_np = np.add(array1, array2)
print("Addition using np.add():", addition_np)

# User-defined addition function using np.frompyfunc()
def add_elements(a, b):
    return a + b

addition_func = np.frompyfunc(add_elements, 2, 1)
addition_custom = addition_func(array1, array2).astype(int)
print("Addition using user-defined function:", addition_custom)
```

```
Addition using zip(): [ 7  9 11 13 15]
Addition using np.add(): [ 7  9 11 13 15]
Addition using user-defined function: [ 7  9 11 13 15]
```

In [ ]:
```python
#Q5

from functools import reduce
import numpy as np
import math

# Function to calculate the GCD (Greatest Common Divisor) of two numbers
def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

# Function to calculate the LCM (Least Common Multiple) of two numbers
def lcm(a, b):
    return a * b // gcd(a, b)

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

# Calculate the GCD of the array using reduce()
gcd_result = reduce(gcd, array)
print("GCD of the array:", gcd_result)

# Calculate the LCM of the array using reduce() and the GCD function
lcm_result = reduce(lcm, array)
print("LCM of the array:", lcm_result)
```

```
GCD of the array: 6
LCM of the array: 72
```

In [ ]: