In [ ]:
```
'''Q1. Create two 3×3 matrices using the random function in Numpy and perform the f
 Product (prod)
 Multiplication (multiply)
 Dot Product (dot)'''


import numpy as np

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

print("Matrix 1:")
print(matrix1)

print("Matrix 2:")
print(matrix2)

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

print("Product Result:")
print(product_result)

print("Multiplication Result:")
print(multiplication_result)

print("Dot Product Result:")
print(dot_product_result)
```

```
Matrix 1:
[[0.8606337  0.607812   0.65018691]
 [0.16778852 0.12707266 0.09898154]
 [0.26363556 0.46498528 0.26851237]]
Matrix 2:
[[0.27110988 0.38274271 0.68339739]
 [0.08024701 0.72453999 0.9462554 ]
 [0.00853458 0.66922299 0.49427847]]
Product Result:
2.362663527421224e-05
Multiplication Result:
[[0.2333263  0.23263561 0.44433604]
 [0.01346453 0.09206923 0.09366181]
 [0.00225002 0.31117884 0.13271988]]
Dot Product Result:
[[0.28765047 1.2049054  1.4846736 ]
 [0.05653109 0.22252978 0.28383387]
 [0.11107953 0.61749967 0.75288257]]
```

In [ ]:
```
'''Q2. Perform the following set operations using the Numpy functions.
è Union
è Intersection
è Set difference
```

```
è XOR
'''
import numpy as np

# Create two NumPy arrays to represent sets
set1 = np.array([1, 2, 3, 4, 5])
set2 = np.array([3, 4, 5, 6, 7])

# Union
union_result = np.union1d(set1, set2)
print("Union Result:")
print(union_result)

# Intersection
intersection_result = np.intersect1d(set1, set2)
print("Intersection Result:")
print(intersection_result)

# Set Difference (set1 - set2)
set_difference_result = np.setdiff1d(set1, set2)
print("Set Difference Result (set1 - set2):")
print(set_difference_result)

# XOR (Symmetric Difference)
xor_result = np.setxor1d(set1, set2)
print("XOR (Symmetric Difference) Result:")
print(xor_result)
```

```
Union Result:
[1 2 3 4 5 6 7]
Intersection Result:
[3 4 5]
Set Difference Result (set1 - set2):
[1 2]
XOR (Symmetric Difference) Result:
[1 2 6 7]
```

In [ ]:

```
'''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'''


import numpy as np

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

print("Original Array:")
print(random_array)

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

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

# Discrete Difference (with n=3)
n = 3
discrete_difference = np.diff(random_array, n=n)
print(f"Discrete Difference (n={n}):")
print(discrete_difference)

# Find Unique Elements
unique_elements = np.unique(random_array)
print("Unique Elements:")
print(unique_elements)
```

```
Original Array:
[0.62750001 0.66669792 0.48353932 0.19466011 0.5751654  0.04829884
 0.95341591 0.45789676 0.10251576 0.19388719]
Cumulative Sum:
[0.62750001 1.29419793 1.77773725 1.97239736 2.54756275 2.5958616
 3.54927751 4.00717427 4.10969003 4.30357722]
Cumulative Product:
[6.27500007e-01 4.18352950e-01 2.02290103e-01 3.93778129e-02
 2.26487554e-02 1.09390871e-03 1.04294997e-03 4.77563411e-04
 4.89577756e-05 9.49228573e-06]
Discrete Difference (n=3):
[ 0.11663589  0.77510513 -1.57675635  2.33935546 -2.83261985  1.54077439
  0.30661427]
Unique Elements:
[0.04829884 0.10251576 0.19388719 0.19466011 0.45789676 0.48353932
 0.5751654  0.62750001 0.66669792 0.95341591]
```

In [ ]:
```
'''Q4. Create two 1D array and perform the Addition using zip(), add() and user def


import numpy as np

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

# Perform addition using zip()
addition_zip = [x + y for x, y in zip(array1, array2)]
print("Addition using zip():")
print(addition_zip)

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

# Define a user-defined addition function using numpy.frompyfunc()
def custom_add(x, y):
    return x + y
```

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

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

In [ ]:

```
'''Q5. Find the LCM (Least Common Multiple) and GCD (Greatest Common Divisor) of an

from functools import reduce
import math

# Define a function to find the LCM of two numbers
def lcm(x, y):
    return x * y // math.gcd(x, y)

# Define a function to find the GCD of two numbers
def gcd(x, y):
    return math.gcd(x, y)

# Example array of elements
array = [12, 18, 24, 36]

# Find the LCM of the elements in the array
lcm_result = reduce(lcm, array)

# Find the GCD of the elements in the array
gcd_result = reduce(gcd, array)

print("Array:", array)
print("LCM of the elements:", lcm_result)
print("GCD of the elements:", gcd_result)
```

```
Array: [12, 18, 24, 36]
LCM of the elements: 72
GCD of the elements: 6
```