```python
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("\nMatrix 2:")
print(matrix2)

# Calculate the product of the two matrices
product_result = np.prod([matrix1, matrix2], axis=0)
print("\nProduct of matrices:")
print(product_result)

# Perform element-wise multiplication
multiply_result = np.multiply(matrix1, matrix2)
print("\nElement-wise multiplication:")
print(multiply_result)

# Calculate the dot product of the two matrices
dot_product_result = np.dot(matrix1, matrix2)
print("\nDot product of matrices:")
print(dot_product_result)
```

```
Matrix 1:
[[0.46269482 0.1605347  0.86142764]
 [0.22252666 0.82588103 0.76073336]
 [0.15082994 0.39135715 0.93855117]]

Matrix 2:
[[0.15937351 0.31768567 0.93514727]
 [0.06997297 0.70378267 0.09101898]
 [0.84182388 0.6595554  0.22394663]]

Product of matrices:
[[0.0737413  0.05099957 0.8055617 ]
 [0.01557085 0.58124075 0.06924117]
 [0.12697225 0.25812172 0.21018537]]

Element-wise multiplication:
[[0.0737413  0.05099957 0.8055617 ]
 [0.01557085 0.58124075 0.06924117]
 [0.12697225 0.25812172 0.21018537]]

Dot product of matrices:
[[0.81014474 0.82813231 0.64021332]
```

```
 [0.73365771 1.15368008 0.45362972]
 [0.8415175  0.94237338 0.38685451]]
```

```
import numpy as np

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

# Perform set operations
union_result = np.union1d(set1, set2)  # Union
intersection_result = np.intersect1d(set1, set2)  # Intersection
set_difference_result = np.setdiff1d(set1, set2)  # Set Difference
(set1 - set2)
xor_result = np.setxor1d(set1, set2)  # XOR

# Print the results
print("Union Result:", union_result)
print("Intersection Result:", intersection_result)
print("Set Difference Result (set1 - set2):", set_difference_result)
print("XOR Result:", xor_result)
```

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

```
import numpy as np

# Create a random 1D array with 10 elements
np.random.seed(42)  # Setting seed for reproducibility
random_array = np.random.rand(10)

# Cumulative sum
cumulative_sum = np.cumsum(random_array)

# Cumulative product
cumulative_product = np.cumprod(random_array)

# Discrete difference with n=3
discrete_difference = np.diff(random_array, n=3)

# Find unique elements in the array
unique_elements = np.unique(random_array)

# Print the results
print("Random Array:", random_array)
print("Cumulative Sum:", cumulative_sum)
print("Cumulative Product:", cumulative_product)
print("Discrete Difference (n=3):", discrete_difference)
print("Unique Elements:", unique_elements)
```

```
Random Array: [0.37454012 0.95071431 0.73199394 0.59865848 0.15601864
0.15599452
 0.05808361 0.86617615 0.60111501 0.70807258]
Cumulative Sum: [0.37454012 1.32525443 2.05724837 2.65590685
2.81192549 2.96792001
 3.02600362 3.89217977 4.49329478 5.20136736]
Cumulative Product: [3.74540119e-01 3.56080649e-01 2.60648878e-01
1.56039662e-01
 2.43450960e-02 3.79770157e-03 2.20584225e-04 1.91064794e-04
 1.14851916e-04 8.13234921e-05]
Discrete Difference (n=3): [ 0.88027946 -0.39468929  0.75192011 -
0.54050251  1.00389023 -1.97915711
  1.44517237]
Unique Elements: [0.05808361 0.15599452 0.15601864 0.37454012
0.59865848 0.60111501
 0.70807258 0.73199394 0.86617615 0.95071431]
```

```python
import numpy as np

# Create two 1D arrays
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([10, 20, 30, 40, 50])

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

# Addition using numpy.add()
result_add = np.add(array1, array2)
print("Result using numpy.add():", result_add)

# User-defined function for addition
def custom_add(x, y):
    return x + y

# Use numpy.frompyfunc() to create a ufunc
custom_add_func = np.frompyfunc(custom_add, 2, 1)

# Perform addition using the user-defined function
result_custom = custom_add_func(array1, array2)
print("Result using user-defined function:", result_custom)
```

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

```python
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
```