adhish-203-lab7

September 4, 2023

Adhish Bahl

2347203

Python Lab7

Q1) Create two 3×3 matrices using the random function in Numpy and perform the following operations.

- Product (prod)
- Multiplication (multiply)
- Dot Product (dot)

[25]:
```
import numpy as np

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

productResult = np.multiply(matrix1, matrix2)
multiplyResult = np.dot(matrix1, matrix2)
dotResult = np.dot(matrix1.flatten(), matrix2.flatten())

print("Matrix 1:")
print(matrix1)
print("\nMatrix 2:")
print(matrix2)
print("================================================================================")
print("\n\nProduct (Element-wise multiplication):")
print(productResult)
print("\n\nMatrix Multiplication (Dot Product):")
print(multiplyResult)
print("\n\nDot Product (Flattened):", round(dotResult, 4))
```

```
Matrix 1:
[[0.521 0.203 0.287]
 [0.465 0.874 0.37 ]
 [0.893 0.599 0.537]]

Matrix 2:
[[0.196 0.796 0.032]
```

```
 [0.196 0.198 0.383]
 [0.681 0.747 0.891]]
================================================================================


Product (Element-wise multiplication):
[[0.102116 0.161588 0.009184]
 [0.09114  0.173052 0.14171 ]
 [0.608133 0.447453 0.478467]]


Matrix Multiplication (Dot Product):
[[0.337351 0.669299 0.350138]
 [0.514414 0.819582 0.679292]
 [0.658129 1.230569 0.73646 ]]


Dot Product (Flattened): 2.2128
```

Q2) Perform the following set operations using the Numpy functions.

- Union
- Intersection
- Set difference
- XOR

[26]:
```python
import numpy as np

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

unionResult = np.union1d(array1, array2)
intersectionResult = np.intersect1d(array1, array2)
differenceResult = np.setdiff1d(array1, array2)
xorResult = np.setxor1d(array1, array2)

print("Array 1: ", array1)
print("Array 2: ", array2)

print("\nUnion: ", unionResult)
print("Intersection: ", intersectionResult)
print("Set Difference: ", differenceResult)
print("XOR: ", xorResult)
```

```
Array 1: [1 2 3 4 5]
Array 2: [3 4 5 6 7]

Union: [1 2 3 4 5 6 7]
```

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

Q3) Create a 1D array using Random function and perform the following operations.

- Cumulative sum
- Cumulative Product
- Discrete difference (with n=3)
- Find the unique elements from the array

[27]:
```
import numpy as np

array = np.random.randint(1, 11, 10)

cumulativeSum = np.cumsum(array)
cumulativeProduct = np.cumprod(array)
discreteDifference = np.diff(array, n=3)
uniqueElements = np.unique(array)

print("Original Array: ", array)
print("Cumulative Sum: ", cumulativeSum)
print("Cumulative Product: ", cumulativeProduct)
print("Discrete Difference: ", discreteDifference)
print("Unique Elements: ", uniqueElements)
```

```
Original Array: [ 7  7  4  2  2 10  7  4  6  9]
Cumulative Sum: [ 7 14 18 20 22 32 39 43 49 58]
Cumulative Product: [       7       49      196      392      784     7840
54880   219520
  1317120 11854080]
Discrete Difference (n=3): [  4   1   6 -19  11   5  -4]
Unique Elements: [ 2  4  6  7  9 10]
```

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

[28]:
```
import numpy as np

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

additionZip = [a1 + a2 for a1, a2 in zip(array1, array2)]

additionAdd = np.add(array1, array2)

def add(x, y):
    return x + y
```

```
additionCustom = np.frompyfunc(add, 2, 1)(array1, array2)

print("Array 1:", array1)
print("Array 2:", array2)

print("Addition using zip:", additionZip)
print("Addition using np.add():", additionAdd)
print("Addition using custom function:", additionCustom)
```

```
Array 1: [1 2 3 4 5]
Array 2: [5 4 3 2 1]
Addition using zip: [6, 6, 6, 6, 6]
Addition using np.add(): [6 6 6 6 6]
Addition using custom function: [6 6 6 6 6]
```

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

[29]:
```
import numpy as np
from functools import reduce
import math

array = np.array([12, 18, 24, 36])

def lcm(x, y):
    return x * y // math.gcd(x, y)

lcmResult = reduce(lcm, array)

gcdResult = reduce(math.gcd, array)

print("Array: ", array)
print("LCM of the array: ", lcmResult)
print("GCD of the array: ", gcdResult)
```

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