sagar-250-lab7

September 4, 2023

[14]:

```
# 1 Matrices
import numpy as np
matrix1=np.random.randint(1, 10, size=(3, 3))
matrix2=np.random.randint(1, 10, size=(3, 3))
matrix0= matrix1*matrix2
result_matrix = np.dot(matrix1,matrix2) #Dot product
result_matrix2 = np.multiply(matrix1,matrix2) #Multiplication
print(matrix0)
print(matrix1)
print(matrix2)
print(result_matrix)
```

```
[[63 63 18]
 [40 18  3]
 [12 15 56]]
[[9 7 9]
 [8 9 1]
 [3 3 7]]
[[7 9 2]
 [5 2 3]
 [4 5 8]]
[[134 140 111]
 [105  95  51]
 [ 64  68  71]]
```

[19]:

```
# 2. Set Operation
set1 = np.array([1,2,3,4,5,6,7])
set2 = np.array([3,4,5,6,9,11])

union_result = np.union1d(set1,set2) #Union
intersection_result = np.intersect1d(set1,set2) #Intersection
set_difference_result = np.setdiff1d(set1,set2) #set Difference
xor_result = np.setxor1d(set1,set2) #Xor operation
print(union_result)
print(intersection_result)
print(set_difference_result)
print(xor_result)
```

```
[ 1  2  3  4  5  6  7  9 11]
[3 4 5 6]
[1 2 7]
[ 1  2  7  9 11]
```

[21]:

```python
#3.
import numpy as np

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

# Calculate the cumulative sum
cumulative_sum = np.cumsum(random_array)


print(random_array)

print(cumulative_sum) #Cumulative sum

# Calculate
cumulative_product = np.cumprod(random_array)
print(cumulative_product) #Cumulative Product

arr = np.array([10, 15, 20, 25, 30, 35, 40, 45, 50])


diff_n3 = np.diff(arr, n=3) #Discrete difference


arr = np.array([1, 2, 2, 3, 4, 4, 5, 5, 6]) #Array with and duplicate elements


unique_elements = np.unique(arr) #Unique elements

print(unique_elements)
```

```
[0.40893745 0.1982575  0.4685946  0.97630113 0.78120091 0.88067753
 0.27681765 0.95685285 0.14349857 0.39257053]
[0.40893745 0.60719495 1.07578956 2.05209068 2.8332916  3.71396913
 3.99078678 4.94763963 5.0911382  5.48370873]
[4.08937449e-01 8.10749179e-02 3.79912689e-02 3.70909186e-02
 2.89754596e-02 2.55180363e-02 7.06384277e-03 6.75905811e-03
 9.69915150e-04 3.80760103e-04]
[1 2 3 4 5 6]
```

[22]:

```python
#4.
import numpy as np
```

```
array1 = np.array([1, 2, 3, 4, 5])
array2 = np.array([6, 7, 8, 9, 10])


result_zip = np.array([a + b for a, b in zip(array1, array2)]) #Addition using␣
↪zip()


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


def custom_add(x, y): #Define a user-defined addition function
    return x + y

ufunc_custom_add = np.frompyfunc(custom_add, 2, 1) #User defined function

# Addition using the user-defined function
result_custom_add = ufunc_custom_add(array1, array2)

# Printing  the results
print("Array 1:", array1)
print("Array 2:", array2)
print("Addition using zip():", result_zip)
print("Addition using np.add():", result_np_add)
print("Addition using user-defined function:", result_custom_add)
```

```
Array 1: [1 2 3 4 5]
Array 2: [ 6  7  8  9 10]
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]
```

[23]:
```
#5.
from functools import reduce
from math import gcd, lcm

# Creating an array of elements
elements = [12, 18, 24, 36, 48]


lcm_result = reduce(lambda x, y: lcm(x, y), elements) #LCM using reduce()


gcd_result = reduce(lambda x, y: gcd(x, y), elements) #Calculating the GCD␣
↪(Greatest Common Divisor) using reduce()
```

```python
print("LCM (Least Common Multiple):", lcm_result)
print("GCD (Greatest Common Divisor):", gcd_result)
```

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