**Day-10 Evening Assessment**

**Numpy**

import numpy as np  
# 1. Create a NumPy array of integers from 10 to 50 (inclusive)  
arr1 = np.arange(10, 51)  
print("1:", arr1)  
# 2. Create a 3x3 NumPy array of all True values  
arr2 = np.ones((3, 3), dtype=bool)  
print("\n2:\n", arr2)  
# 3. Create a 5x5 identity matrix  
arr3 = np.eye(5)  
print("\n3:\n", arr3)  
# 4. Generate an array of 10 random float numbers between 0 and 1  
arr4 = np.random.rand(10)  
print("\n4:", arr4)  
# 5. Create a 1D array of 15 numbers equally spaced between 0 and 5  
arr5 = np.linspace(0, 5, 15)  
print("\n5:", arr5)  
# 6. Reshape an array of 12 elements into a 3x4 matrix  
arr6 = np.arange(12).reshape(3, 4)  
print("\n6:\n", arr6)  
# 7. Replace all even numbers in the array [1, 2, 3, 4, 5, 6] with -1  
arr7 = np.array([1, 2, 3, 4, 5, 6])  
arr7[arr7 % 2 == 0] = -1  
print("\n7:", arr7)  
# 8. Extract all odd numbers from a 1D array ranging from 0 to 20  
arr8 = np.arange(0, 21)  
odds = arr8[arr8 % 2 == 1]  
print("\n8:", odds)  
# 9. Create a 2D array of shape (4, 5) and calculate the sum of each column  
arr9 = np.random.randint(1, 10, size=(4, 5))  
col\_sum = arr9.sum(axis=0)  
print("\n9 Array:\n", arr9)  
print("Column-wise sum:", col\_sum)  
# 10. Create two 3x3 arrays and perform element-wise multiplication  
a = np.random.randint(1, 10, size=(3, 3))  
b = np.random.randint(1, 10, size=(3, 3))  
result = a \* b  
print("\n10 Array A:\n", a)  
print("Array B:\n", b)  
print("Element-wise multiplication:\n", result)  
# 11. Create an array from 1 to 100 and count how many numbers are divisible by both 3 and 5  
arr11 = np.arange(1, 101)  
count = np.sum((arr11 % 3 == 0) & (arr11 % 5 == 0))  
print("\n11: Numbers divisible by both 3 and 5:", count)  
# 12. Normalize a NumPy array: subtract its mean and divide by its standard deviation  
arr12 = np.random.randint(1, 100, size=10)  
normalized = (arr12 - np.mean(arr12)) / np.std(arr12)  
print("\n12 Original:", arr12)  
print("Normalized:", normalized)

o/p:

1: [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33

34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50]

2:

[[ True True True]

[ True True True]

[ True True True]]

3:

[[1. 0. 0. 0. 0.]

[0. 1. 0. 0. 0.]

[0. 0. 1. 0. 0.]

[0. 0. 0. 1. 0.]

[0. 0. 0. 0. 1.]]

4: [0.12530669 0.08449092 0.77240856 0.7140684 0.52720126 0.1652475

0.55553632 0.57403909 0.57401636 0.18848514]

5: [0. 0.35714286 0.71428571 1.07142857 1.42857143 1.78571429

2.14285714 2.5 2.85714286 3.21428571 3.57142857 3.92857143

4.28571429 4.64285714 5. ]

6:

[[ 0 1 2 3]

[ 4 5 6 7]

[ 8 9 10 11]]

7: [ 1 -1 3 -1 5 -1]

8: [ 1 3 5 7 9 11 13 15 17 19]

9 Array:

[[9 7 9 6 5]

[5 6 5 1 1]

[4 7 9 2 9]

[8 6 5 7 9]]

Column-wise sum: [26 26 28 16 24]

10 Array A:

[[7 2 1]

[1 4 6]

[9 5 5]]

Array B:

[[4 1 8]

[2 1 7]

[1 1 9]]

Element-wise multiplication:

[[28 2 8]

[ 2 4 42]

[ 9 5 45]]

11: Numbers divisible by both 3 and 5: 6

12 Original: [81 22 69 42 33 70 32 9 13 53]

Normalized: [ 1.62184874 -0.85714286 1.11764706 -0.01680672 -0.39495798 1.15966387

-0.43697479 -1.40336134 -1.23529412 0.44537815]