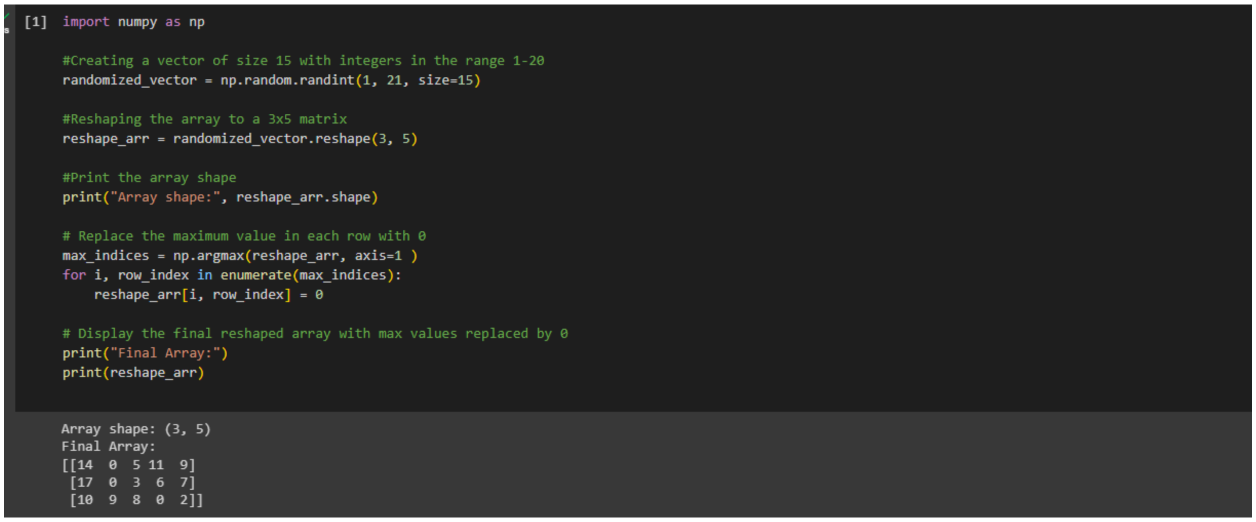
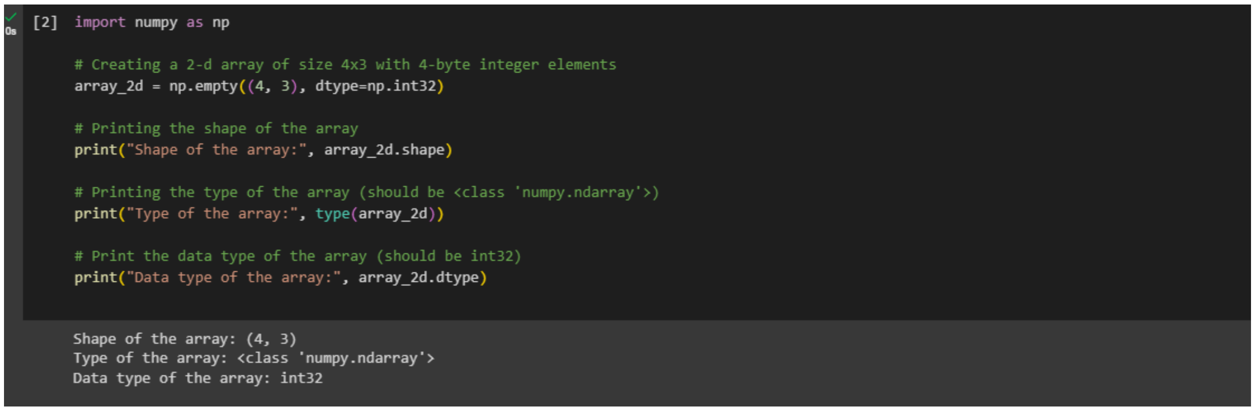
ICP_3 REPORT



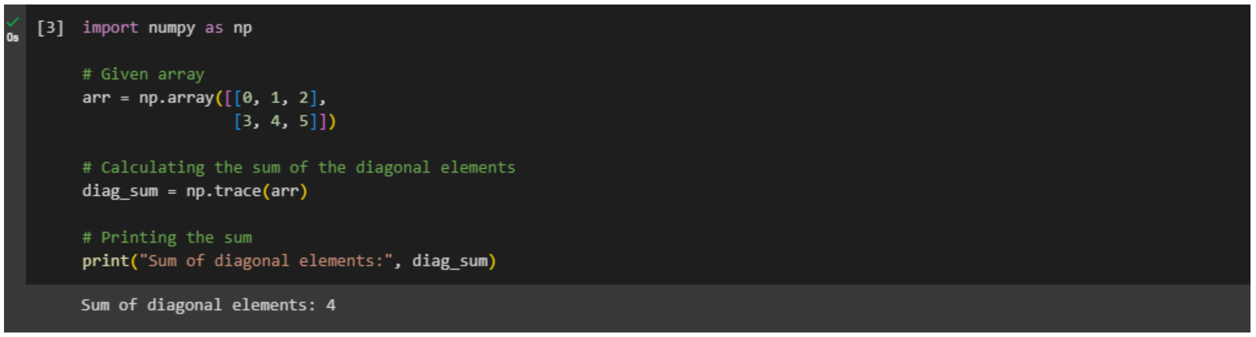
```
[1] import numpy as np

    #Creating a vector of size 15 with integers in the range 1-20
    randomized_vector = np.random.randint(1, 21, size=15)

    #Reshaping the array to a 3x5 matrix
    reshape_arr = randomized_vector.reshape(3, 5)

    #Print the array shape
    print("Array shape:", reshape_arr.shape)

    # Replace the maximum value in each row with 0
    max_indices = np.argmax(reshape_arr, axis=1 )
    for i, row_index in enumerate(max_indices):
        reshape_arr[i, row_index] = 0

    # Display the final reshaped array with max values replaced by 0
    print("Final Array:")
    print(reshape_arr)
```

```
Array shape: (3, 5)
Final Array:
[[14  0  5 11  9]
 [17  0  3  6  7]
 [10  9  8  0  2]]
```



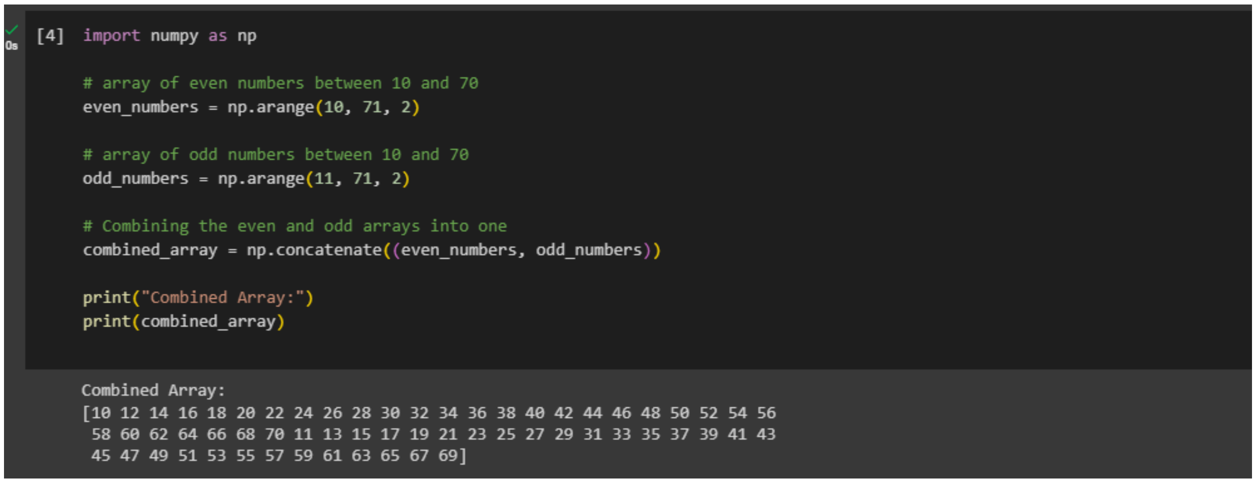
```
[2] import numpy as np

    # Creating a 2-d array of size 4x3 with 4-byte integer elements
    array_2d = np.empty((4, 3), dtype=np.int32)

    # Printing the shape of the array
    print("Shape of the array:", array_2d.shape)

    # Printing the type of the array (should be <class 'numpy.ndarray'>)
    print("Type of the array:", type(array_2d))

    # Print the data type of the array (should be int32)
    print("Data type of the array:", array_2d.dtype)
```

```
Shape of the array: (4, 3)
Type of the array: <class 'numpy.ndarray'>
Data type of the array: int32
```



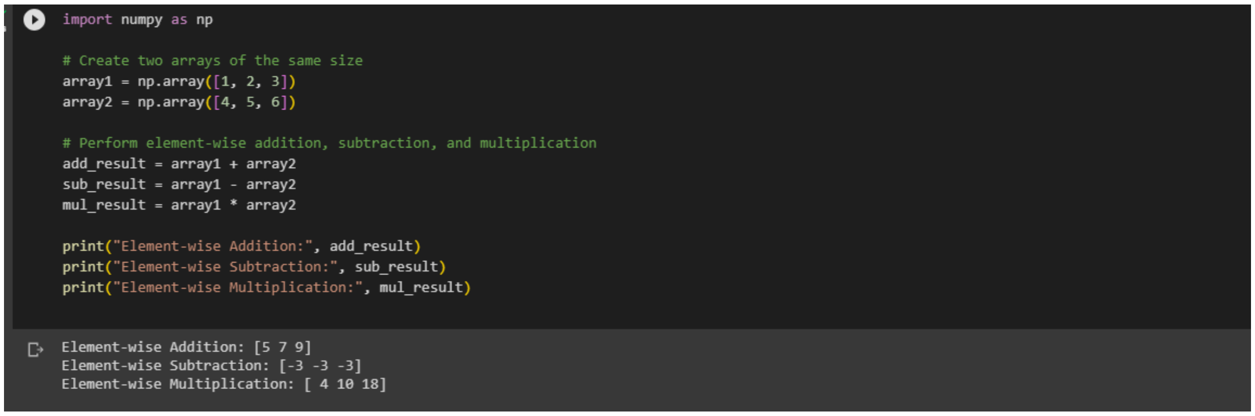
```
[3] import numpy as np

    # Given array
    arr = np.array([[0, 1, 2],
                    [3, 4, 5]])

    # Calculating the sum of the diagonal elements
    diag_sum = np.trace(arr)

    # Printing the sum
    print("Sum of diagonal elements:", diag_sum)
```

```
Sum of diagonal elements: 4
```



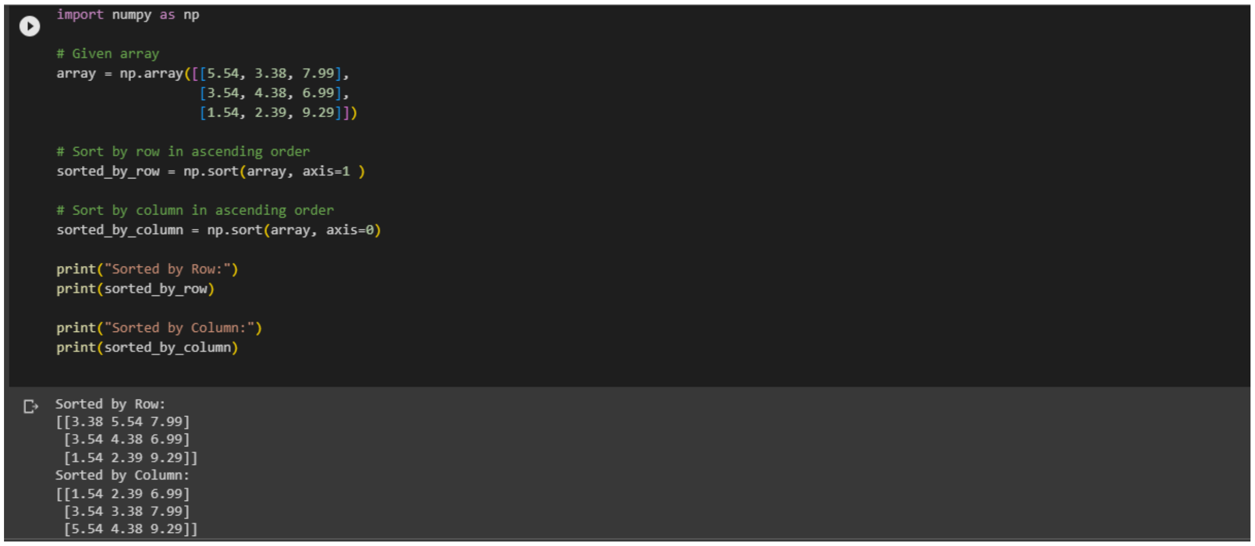
```
[4] import numpy as np

    # array of even numbers between 10 and 70
    even_numbers = np.arange(10, 71, 2)

    # array of odd numbers between 10 and 70
    odd_numbers = np.arange(11, 71, 2)

    # Combining the even and odd arrays into one
    combined_array = np.concatenate((even_numbers, odd_numbers))

    print("Combined Array:")
    print(combined_array)
```

```
Combined Array:
[10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56
 58 60 62 64 66 68 70 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43
 45 47 49 51 53 55 57 59 61 63 65 67 69]
```



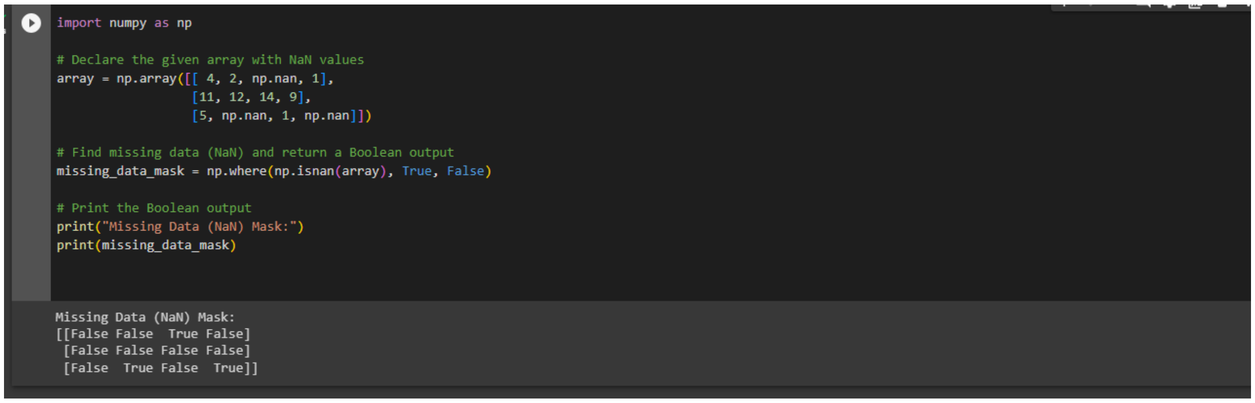
```
import numpy as np

# Create two arrays of the same size
array1 = np.array([1, 2, 3])
array2 = np.array([4, 5, 6])

# Perform element-wise addition, subtraction, and multiplication
add_result = array1 + array2
sub_result = array1 - array2
mul_result = array1 * array2

print("Element-wise Addition:", add_result)
print("Element-wise Subtraction:", sub_result)
print("Element-wise Multiplication:", mul_result)
```

```
Element-wise Addition: [5 7 9]
Element-wise Subtraction: [-3 -3 -3]
Element-wise Multiplication: [ 4 10 18]
```



```
import numpy as np

# Given array
array = np.array([[5.54, 3.38, 7.99],
                  [3.54, 4.38, 6.99],
                  [1.54, 2.39, 9.29]])

# Sort by row in ascending order
sorted_by_row = np.sort(array, axis=1 )

# Sort by column in ascending order
sorted_by_column = np.sort(array, axis=0)

print("Sorted by Row:")
print(sorted_by_row)

print("Sorted by Column:")
print(sorted_by_column)
```

```
Sorted by Row:
[[3.38 5.54 7.99]
 [3.54 4.38 6.99]
 [1.54 2.39 9.29]]
Sorted by Column:
[[1.54 2.39 6.99]
 [3.54 3.38 7.99]
 [5.54 4.38 9.29]]
```



```
import numpy as np

# Declare the given array with NaN values
array = np.array([[ 4, 2, np.nan, 1],
                  [11, 12, 14, 9],
                  [5, np.nan, 1, np.nan]])

# Find missing data (NaN) and return a Boolean output
missing_data_mask = np.where(np.isnan(array), True, False)

# Print the Boolean output
print("Missing Data (NaN) Mask:")
print(missing_data_mask)
```

```
Missing Data (NaN) Mask:
[[False False  True False]
 [False False False False]
 [False  True False  True]]
```

Github Repo Link: https://github.com/sxk7912/Bigdata

youtube link: https://youtu.be/q2KD3vZ1k_A