```python
#1
# Using NumPy create random vector of size 15 having only Integers in the range 1-20.
# a. Reshape the array to 3 by 5.
# b. Print array shape.
# c. Replace the max in each row by 0

import numpy as np

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

# Reshaping the array to 3x5
reshaped_array = random_vector.reshape(3, 5)

# Printing array shape
array_shape = reshaped_array.shape

# Replacing the max in each row by 0
for row in reshaped_array:
    row[row.argmax()] = 0

print("RESHAPED ARRAY:\n" , reshaped_array)
print("ARRAY SHAPE:\n", array_shape)
```

```
RESHAPED ARRAY:
 [[ 0 17  3 11  2]
 [ 0  8 14  6 12]
 [10  1  0  7 13]]
ARRAY SHAPE:
 (3, 5)
```

```python
#2
# Create a 2-dimensional array of size 4 x 3 (composed of 4-byte integer elements), also print the shape,
# type, and data type of the array.

# Creating a 2-dimensional array of size 4x3 with 4-byte integer elements
arr_2d = np.random.randint(1, 100, size=(4, 3), dtype=np.int32)

# Printing the shape, type, and data type of the array
arr_shape = arr_2d.shape
arr_type = type(arr_2d)
arr_dtype = arr_2d.dtype

print("ARRAY:\n" , arr_2d)

print("ARRAY SHAPE:\n",  arr_shape)

print("ARRAY TYPE:\n", arr_type)

print("ARRAY DATA TYPE:\n", arr_dtype)
```

```
ARRAY:
 [[28 95 55]
 [66 73 42]
 [ 2 19 85]
 [15 93 14]]
ARRAY SHAPE:
 (4, 3)
ARRAY TYPE:
 <class 'numpy.ndarray'>
ARRAY DATA TYPE:
 int32
```

```python
#3
# Compute the sum of the diagonal element of a given array.
# [[0 1 2]
# [3 4 5]]

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

# Compute the sum of the diagonal elements
diagonal_sum = np.trace(given_array)

print("DIAGONAL SUM:\n", diagonal_sum)
```

```
DIAGONAL SUM:
 4
```

```python
#4
# Write a NumPy code:
# a. To create an array of odd and even numbers between 10 to 70.
# b. To perform at-least three element-wise mathematical operations using two arrays of the same size.
# c. To sort a given array by row and column in ascending order.
# array = [[5.54, 3.38, 7.99],
# [3.54, 4.38, 6.99],
# [1.54, 2.39, 9.29]]

# a. Creating an array of odd and even numbers between 10 to 70
odd_numbers = np.arange(10, 70, 2)
even_numbers = np.arange(10, 70, 2)

# b. Creating two arrays of the same size for element-wise operations
array1 = np.random.randint(1, 10, size=(3, 3))
array2 = np.random.randint(1, 10, size=(3, 3))

# Performing at least three element-wise mathematical operations
elementwise_sum = array1 + array2
elementwise_diff = array1 - array2
elementwise_product = array1 * array2

# c. Sorting the given array by row and column in ascending order
given_array = np.array([[5.54, 3.38, 7.99],
                        [3.54, 4.38, 6.99],
                        [1.54, 2.39, 9.29]])

# Sorting by row
sorted_by_row = np.sort(given_array, axis=1)
```

```python
# Sorting by column
sorted_by_column = np.sort(given_array, axis=0)

print("ODD NUMBERS:\n", odd_numbers)
print("EVEN NUMBERS:\n", even_numbers)
print("ARRAY 1:\n", array1)
print("ARRAY 2:\n", array2)
print("ELEMENTWISE SUM:\n", elementwise_sum)
print("ELEMENTWISE DIFF:\n", elementwise_diff)
print("ELEMENTWISE PRODUCT:\n", elementwise_product)
print("SORTED BY ROW:\n", sorted_by_row)
print("SORTED BY COLUMN:\n", sorted_by_column)
```

```
ODD NUMBERS:
 [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]
EVEN NUMBERS:
 [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]
ARRAY 1:
 [[4 5 8]
 [8 5 2]
 [4 3 3]]
ARRAY 2:
 [[7 1 5]
 [7 7 6]
 [7 6 8]]
ELEMENTWISE SUM:
 [[11  6 13]
 [15 12  8]
 [11  9 11]]
ELEMENTWISE DIFF:
 [[-3  4  3]
 [ 1 -2 -4]
 [-3 -3 -5]]
ELEMENTWISE PRODUCT:
 [[28  5 40]
 [56 35 12]
 [28 18 24]]
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]]
```

```python
#5
# Declare an array given below using NumPy and find missing data by returning Boolean output
# (True/False).
# Array = [[ 4 2 nan 1]
# [11 12 14 9]
# [5 nan 1 nan]]

# Importing NumPy and declaring the array with missing data (nan values)
import numpy as np

# Declaring the array
given_array = np.array([[4, 2, np.nan, 1],
                        [11, 12, 14, 9],
                        [5, np.nan, 1, np.nan]])

# Finding missing data by returning Boolean output (True for missing, False for present)
missing_data = np.isnan(given_array)

print("Missing Data:\n", missing_data)
```

```
Missing Data:
 [[False False  True False]
 [False False False False]
 [False  True False  True]]
```

Github Repo Link: https://github.com/Krypton0626/Bigdata/tree/main/ICP%203