Q1

```python
import numpy as np

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

# Reshape the array to 3 by 5
reshaped_array = vector.reshape(3, 5)
print(f"Reshaped Array: {reshaped_array}")

# Print array shape
shape = reshaped_array.shape
print(f"Shape of Array: {shape}")

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

print(f"Array after replacing max by 0: {reshaped_array}")
```

```
Reshaped Array: [[ 7  6  9 13  6]
 [ 4 15  6  6  4]
 [18 12 19  4 19]]
Shape of Array: (3, 5)
Array after replacing max by 0: [[ 7  6  9  0  6]
 [ 4  0  6  6  4]
 [18 12  0  4 19]]
```

Q2

```python
# Create a 2-dimensional array of size 4x3 with 4-byte integer elements
array_2d = np.random.randint(-2147483648, 2147483647, size=(4, 3), dtype=np.int32)

# Get the shape, type, and data type of the array
shape_2d = array_2d.shape
type_2d = type(array_2d)
dtype_2d = array_2d.dtype

print(f"2D Array: {array_2d}")
print(f"Shape of array: {shape_2d}")
print(f"Type of array: {type_2d}")
print(f"Data type of array: {dtype_2d}")
```

```
2D Array: [[-1473741170  -119027319 -1026331127]
 [  375647621  -709209646   109918273]
 [ 1613723491  1509479639  2012054799]
 [ -517714807  -155514590 -1300385088]]
Shape of array: (4, 3)
Type of array: <class 'numpy.ndarray'>
Data type of array: int32
```

Q3

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

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

print(f"Diagonal Sum: {diagonal_sum}")
```

```
Diagonal Sum: 4
```

Q4

```python
# To create an array of odd and even numbers between 10 to 70
odd_numbers = np.arange(11, 70, 2)
even_numbers = np.arange(10, 70, 2)

# perform three element-wise mathematical operations using two arrays of the same size
addition = odd_numbers + even_numbers
multiplication = odd_numbers * even_numbers
division = odd_numbers / even_numbers

# sort a 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]])
sorted_by_row = np.sort(given_array, axis=1)
sorted_by_column_and_row = np.sort(sorted_by_row, axis=0)

print(f"Odd Numbers Array: {odd_numbers}")
print(f"Even Numbers Array: {even_numbers}")
print(f"Addition: {addition}")
print(f"Multiplication: {multiplication}")
print(f"Division: {division}")
print(f"Given Array after sorting by row and column: {sorted_by_column_and_row}")
```

```
Odd Numbers Array: [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]
Even Numbers 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]
Addition: [ 21  25  29  33  37  41  45  49  53  57  61  65  69  73  77  81  85  89
  93  97 101 105 109 113 117 121 125 129 133 137]
Multiplication: [ 110  156  210  272  342  420  506  600  702  812  930 1056 1190 1332
 1482 1640 1806 1980 2162 2352 2550 2756 2970 3192 3422 3660 3906 4160
 4422 4692]
Division: [1.1        1.08333333 1.07142857 1.0625     1.05555556 1.05
 1.04545455 1.04166667 1.03846154 1.03571429 1.03333333 1.03125
 1.02941176 1.02777778 1.02631579 1.025      1.02380952 1.02272727
 1.02173913 1.02083333 1.02       1.01923077 1.01851852 1.01785714
 1.01724138 1.01666667 1.01612903 1.015625   1.01515152 1.01470588]
Given Array after sorting by row and column: [[1.54 2.39 6.99]
 [3.38 4.38 7.99]
 [3.54 5.54 9.29]]
```

Q5

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

     # Find missing data by returning Boolean output
     missing_data = np.isnan(array_with_nan)

     print(f"Is Nan: {missing_data}")
```

```
Is Nan: [[False False  True False]
 [False False False False]
 [False  True False  True]]
```

Github Repo: https://github.com/dheerukarra/BigDataAnalytics/tree/main/ICP%203