Suyogya Sedhai
102367011
2Q21

LAB ASSIGNMENT 4

Q.1 Write a program to create a NumPy 1D-array with 5 elements and perform basic operations like:

a) Addition of 2 in all the element

b) Multiply 3 with all the elements

c) Divide every element by 2

Code:

```
import numpy as np


arr = np.array([10, 20, 30, 40, 50])


arr_addition = arr + 2

print("Array after adding 2 to each element:", arr_addition)


arr_multiplication = arr * 3

print("Array after multiplying each element by 3:", arr_multiplication)


arr_division = arr / 2

print("Array after dividing each element by 2:", arr_division)
```

Ouptut:

```
Array after adding 2 to each element: [12 22 32 42 52]
Array after multiplying each element by 3: [ 30  60  90 120 150]
Array after dividing each element by 2: [ 5. 10. 15. 20. 25.]
```

Q.2 Questions on Basic NumPy Array:

a) Reverse the NumPy array: arr = np.array([1, 2, 3, 6, 4, 5])

b) Find the most frequent value and their indice(s) in the following arrays:

i. x = np.array([1,2,3,4,5,1,2,1,1,1])

ii. y = np.array([1, 1, 1, 2, 3, 4, 2, 4, 3, 3,])

Suyogya Sedhai
102367011
2Q21

Code:

```
import numpy as np


# Part (a): Reverse the NumPy array
arr = np.array([1, 2, 3, 6, 4, 5])
reversed_arr = arr[::-1]
print(reversed_arr)


# Part (b): Find the most frequent value and their indices
def most_frequent(arr):
    values, counts = np.unique(arr, return_counts=True)
    max_count = np.max(counts)
    most_frequent_value = values[np.argmax(counts)]
    indices = np.where(arr == most_frequent_value)[0]
    return most_frequent_value, indices

x = np.array([1,2,3,4,5,1,2,1,1,1])
y = np.array([1, 1, 1, 2, 3, 4, 2, 4, 3, 3])

value_x, indices_x = most_frequent(x)
print(value_x, indices_x)

value_y, indices_y = most_frequent(y)
print(value_y, indices_y)
```

Output:

```
[5 4 6 3 2 1]
1 [0 5 7 8 9]
1 [0 1 2]
```

Suyogya Sedhai
102367011
2Q21

Q.3 For the given 2-D array arr=np.array([10, 20, 30], [40, 50, 60], [70, 80, 90]), access elements using row and column indices as follows:

a) Access 1st row, 2nd column

b) Access 3rd row, 1st column

Code:

```
import numpy as np


# Define the 2D array
arr = np.array([[10, 20, 30],
                [40, 50, 60],
                [70, 80, 90]])


# a) Access 1st row, 2nd column (row index 0, column index 1)
element_a = arr[0, 1]


# b) Access 3rd row, 1st column (row index 2, column index 0)
element_b = arr[2, 0]


print("Element at 1st row, 2nd column:", element_a)
print("Element at 3rd row, 1st column:", element_b)
```

Output:

```
Element at 1st row, 2nd column: 20
Element at 3rd row, 1st column: 70
```

Q.4 Write program to create an 1-D NumPy array named <<Your Name>> with evenly spaced 25 numbers from 10 to 100 using linspace(). Print the dimensions of the array, shape, total elements, the data type of each element and total number of bytes consumed by the array. Find the transpose of this array using reshape() attribute. Can we do the same with T attribute?

Suyogya Sedhai
102367011
2Q21

Code:

```
import numpy as np


my_array = np.linspace(10, 100, 25)


print("Array:", my_array)
print("Dimensions of array:", my_array.ndim)  # Number of dimensions
print("Shape of array:", my_array.shape)  # Shape of array
print("Total elements:", my_array.size)  # Number of elements
print("Data type of elements:", my_array.dtype)  # Data type of elements
print("Total bytes consumed:", my_array.nbytes)  # Total bytes used by array

# Transpose using reshape()
transposed_array = my_array.reshape(25, 1)  # Convert 1D to 2D column vector
print("\nTransposed array using reshape():\n", transposed_array)

# Check if we can use the T attribute
transposed_T = my_array.T  # Transpose using T attribute
print("\nTransposed array using T attribute:\n", transposed_T)
```

Output:

(See below in next page)

Suyogya Sedhai
102367011
2Q21

```
Array: [ 10.    13.75  17.5   21.25  25.    28.75  32.5   36.25  40.    43.75
  47.5   51.25  55.    58.75  62.5   66.25  70.    73.75  77.5   81.25
  85.    88.75  92.5   96.25 100.  ]
Dimensions of array: 1
Shape of array: (25,)
Total elements: 25
Data type of elements: float64
Total bytes consumed: 200

Transposed array using reshape():
 [[ 10.  ]
 [ 13.75]
 [ 17.5 ]
 [ 21.25]
 [ 25.  ]
 [ 28.75]
 [ 32.5 ]
 [ 36.25]
 [ 40.  ]
 [ 43.75]
 [ 47.5 ]
 [ 51.25]
 [ 55.  ]
 [ 58.75]
 [ 62.5 ]
 [ 66.25]
 [ 70.  ]
 [ 73.75]
 [ 77.5 ]
 [ 81.25]
 [ 85.  ]
 [ 88.75]
 [ 92.5 ]
 [ 96.25]
 [100.  ]]

Transposed array using T attribute:
 [ 10.    13.75  17.5   21.25  25.    28.75  32.5   36.25  40.    43.75
  47.5   51.25  55.    58.75  62.5   66.25  70.    73.75  77.5   81.25
  85.    88.75  92.5   96.25 100.  ]
```

Q5. Create a 2-D Array of three rows and four columns, named ucs420_<your_name>> with following values – 10, 20, 30, 40, 50, 60, 70, 80, 90, 15, 20, 35. Compute the mean, median, max, min, unique elements. Reshape the array to four rows and three columns and name it as reshaped_ ucs420_<your_name>>. Resize the array to two rows and three columns and name it as resized_ ucs420_<your_name>>.

Suyogya Sedhai
102367011
2Q21

Code:

```
import numpy as np

# Creating the 2-D array (3 rows, 4 columns)
ucs420_myname = np.array([[10, 20, 30, 40],
                          [50, 60, 70, 80],
                          [90, 15, 20, 35]])

# Compute statistical properties
mean_value = np.mean(ucs420_myname)
median_value = np.median(ucs420_myname)
max_value = np.max(ucs420_myname)
min_value = np.min(ucs420_myname)
unique_elements = np.unique(ucs420_myname)

# Reshape to 4 rows and 3 columns
reshaped_ucs420_myname = ucs420_myname.reshape(4, 3)

# Resize to 2 rows and 3 columns (Note: Resize modifies the original array if not copied)
resized_ucs420_myname = np.resize(ucs420_myname, (2, 3))

# Printing results
print("Original 2D Array:\n", ucs420_myname)
print("\nMean:", mean_value)
print("Median:", median_value)
print("Max:", max_value)
print("Min:", min_value)
print("Unique Elements:", unique_elements)

print("\nReshaped Array (4x3):\n", reshaped_ucs420_myname)
```

Suyogya Sedhai
102367011
2Q21

print("\nResized Array (2x3):\n", resized_ucs420_myname)

Output:

```
Original 2D Array:
 [[10 20 30 40]
 [50 60 70 80]
 [90 15 20 35]]

Mean: 43.333333333333336
Median: 37.5
Max: 90
Min: 10
Unique Elements: [10 15 20 30 35 40 50 60 70 80 90]

Reshaped Array (4x3):
 [[10 20 30]
 [40 50 60]
 [70 80 90]
 [15 20 35]]

Resized Array (2x3):
 [[10 20 30]
 [40 50 60]]
```