FDA Lab-9

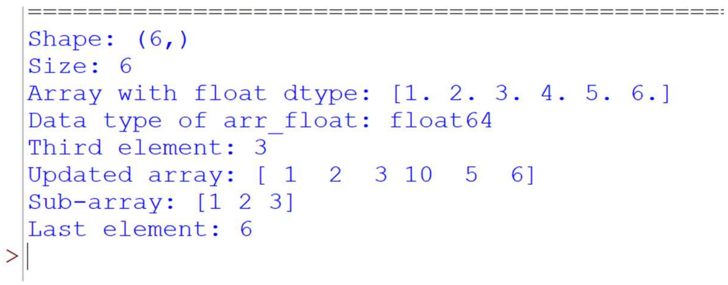
KHAN MOHD OWAIS RAZA
20BCD7138

Q1] Consider the following NumPy array:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])

1. NumPy Array Attributes:
a) Determine the shape and size of the array `arr` using the appropriate NumPy array attributes.
b) Change the dtype of the array `arr` to float and explain the purpose of the dtype attribute in a NumPy array.
2. Array Indexing: Accessing Single Elements:
a) Access and print the third element of the array `arr`.
b) Update the value of the fourth element in the array `arr` to 10.
c) Access and print a sub-array containing the first three elements of the array `arr`.
d) Using negative indexing, access and print the last element of the array `arr`.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
# 1. NumPy Array Attributes
# a) Determine the shape and size of the array `arr`
shape = arr.shape
size = arr.size
print("Shape:", shape)
print("Size:", size)
# b) Change the dtype of the array `arr` to float
arr_float = arr.astype(float)
print("Array with float dtype:", arr_float)
print("Data type of arr_float:", arr_float.dtype)
# 2. Array Indexing: Accessing Single Elements
# a) Access and print the third element of the array
`arr`
third_element = arr[2]
print("Third element:", third_element)
# b) Update the value of the fourth element in the array
`arr` to 10
arr[3] = 10
print("Updated array:", arr)
# c) Access and print a sub-array containing the first
three elements of the array `arr`
```

```
sub_array = arr[:3]
print("Sub-array:", sub_array)
# d) Using negative indexing, access and print the last
element of the array `arr`
last_element = arr[-1]
print("Last element:", last_element)
```



```
==============================================
Shape: (6,)
Size: 6
Array with float dtype: [1. 2. 3. 4. 5. 6.]
Data type of arr_float: float64
Third element: 3
Updated array: [ 1  2  3 10  5  6]
Sub-array: [1 2 3]
Last element: 6
>
```

Q2] Consider the following NumPy array:
import numpy as np
arr = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 10, 11, 12]])

1. Array Slicing: Accessing Subarrays:
a) Access and print the subarray consisting of the first two rows of the array `arr`.
b) Access and print the subarray consisting of the last two columns of the array `arr`.
c) Access and print a subarray consisting of the elements in the second row, starting from the second column, up to and including the third column.
2. Reshaping of Arrays:
a) Reshape the array `arr` into a 2x6 array. Print the reshaped array and explain the result.
b) Reshape the array `arr` into a 1D array. Print the reshaped array and discuss the purpose of reshaping arrays.

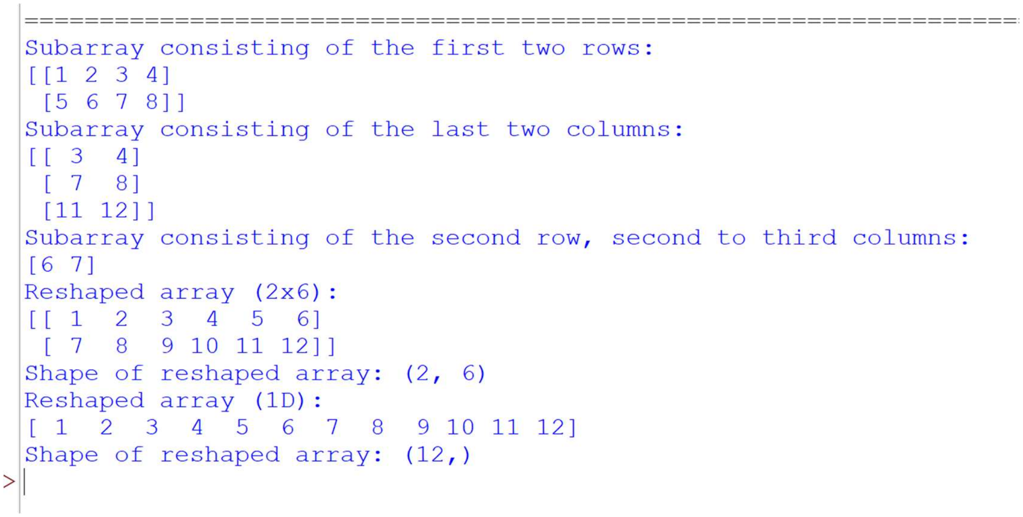
```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import numpy as np
arr = np.array([[1, 2, 3, 4],
                [5, 6, 7, 8],
                [9, 10, 11, 12]])
# 1. Array Slicing: Accessing Subarrays
# a) Access and print the subarray consisting of the
first two rows of the array `arr`
subarray_1 = arr[:2, :]
print("Subarray consisting of the first two rows:")
print(subarray_1)
```

```
# b) Access and print the subarray consisting of the last
two columns of the array `arr`
subarray_2 = arr[:, -2:]
print("Subarray consisting of the last two columns:")
print(subarray_2)

# c) Access and print a subarray consisting of the
elements in the second row, starting from the second
column, up to and including the third column.
subarray_3 = arr[1, 1:3]
print("Subarray consisting of the second row, second to
third columns:")
print(subarray_3)

# 2. Reshaping of Arrays
# a) Reshape the array `arr` into a 2x6 array
reshaped_arr_1 = arr.reshape(2, 6)
print("Reshaped array (2x6):")
print(reshaped_arr_1)
print("Shape of reshaped array:", reshaped_arr_1.shape)

# b) Reshape the array `arr` into a 1D array
reshaped_arr_2 = arr.flatten()
print("Reshaped array (1D):")
print(reshaped_arr_2)
print("Shape of reshaped array:", reshaped_arr_2.shape)
```



```
==================================================================
Subarray consisting of the first two rows:
[[1 2 3 4]
 [5 6 7 8]]
Subarray consisting of the last two columns:
[[ 3  4]
 [ 7  8]
 [11 12]]
Subarray consisting of the second row, second to third columns:
[6 7]
Reshaped array (2x6):
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
Shape of reshaped array: (2, 6)
Reshaped array (1D):
[ 1  2  3  4  5  6  7  8  9 10 11 12]
Shape of reshaped array: (12,)
>
```

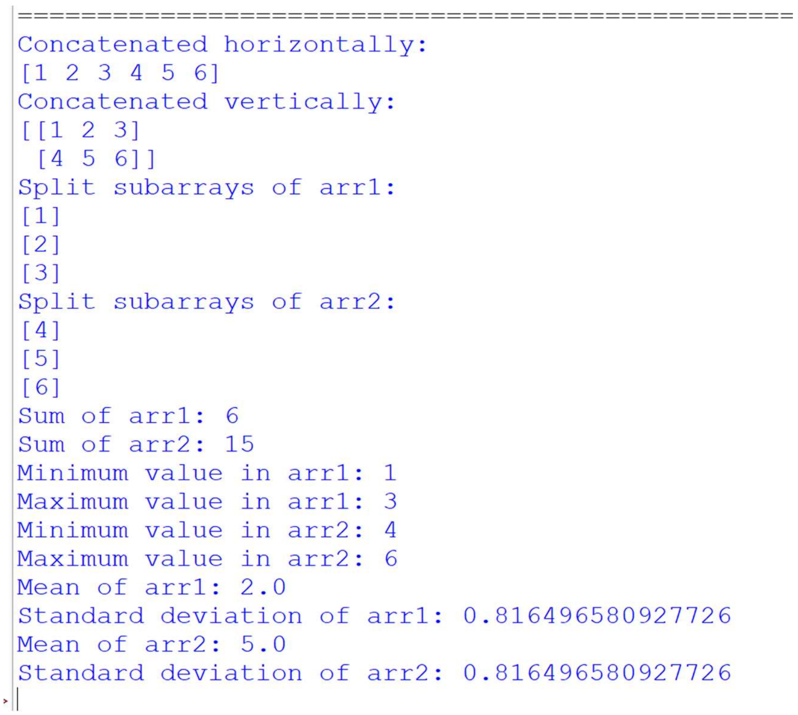
Q3] Consider the following NumPy arrays:
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

1. Array Concatenation and Splitting:
a) Concatenate `arr1` and `arr2` horizontally (column-wise) and print the result.
b) Concatenate `arr1` and `arr2` vertically (row-wise) and print the result.
c) Split the array `arr1` into three equal-sized subarrays. Print the resulting subarrays.
d) Split the array `arr2` at indices 1 and 2. Print the resulting subarrays.

2. Aggregations:
a) Calculate and print the sum of all elements in `arr1` and `arr2`.
b) Find and print the minimum and maximum values in `arr1` and `arr2`.
c) Calculate and print the mean and standard deviation of `arr1` and `arr2`.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
# 1. Array Concatenation and Splitting
# a) Concatenate `arr1` and `arr2` horizontally (column-wise)
concatenated_horizontal = np.concatenate((arr1, arr2), axis=0)
print("Concatenated horizontally:")
print(concatenated_horizontal)
# b) Concatenate `arr1` and `arr2` vertically (row-wise)
concatenated_vertical = np.vstack((arr1, arr2))
print("Concatenated vertically:")
print(concatenated_vertical)
# c) Split the array `arr1` into three equal-sized subarrays
subarrays_arr1 = np.array_split(arr1, 3)
print("Split subarrays of arr1:")
for subarray in subarrays_arr1:
    print(subarray)
# d) Split the array `arr2` at indices 1 and 2
subarrays_arr2 = np.split(arr2, [1, 2])
print("Split subarrays of arr2:")
for subarray in subarrays_arr2:
    print(subarray)
# 2. Aggregations
# a) Calculate and print the sum of all elements in `arr1` and `arr2`
sum_arr1 = np.sum(arr1)
```

```
sum_arr2 = np.sum(arr2)
print("Sum of arr1:", sum_arr1)
print("Sum of arr2:", sum_arr2)
# b) Find and print the minimum and maximum values in
`arr1` and `arr2`
min_arr1 = np.min(arr1)
max_arr1 = np.max(arr1)
min_arr2 = np.min(arr2)
max_arr2 = np.max(arr2)
print("Minimum value in arr1:", min_arr1)
print("Maximum value in arr1:", max_arr1)
print("Minimum value in arr2:", min_arr2)
print("Maximum value in arr2:", max_arr2)
# c) Calculate and print the mean and standard deviation
of `arr1` and `arr2`
mean_arr1 = np.mean(arr1)
std_arr1 = np.std(arr1)
mean_arr2 = np.mean(arr2)
std_arr2 = np.std(arr2)
print("Mean of arr1:", mean_arr1)
print("Standard deviation of arr1:", std_arr1)
print("Mean of arr2:", mean_arr2)
print("Standard deviation of arr2:", std_arr2)
```



```
=================================================
Concatenated horizontally:
[1 2 3 4 5 6]
Concatenated vertically:
[[1 2 3]
 [4 5 6]]
Split subarrays of arr1:
[1]
[2]
[3]
Split subarrays of arr2:
[4]
[5]
[6]
Sum of arr1: 6
Sum of arr2: 15
Minimum value in arr1: 1
Maximum value in arr1: 3
Minimum value in arr2: 4
Maximum value in arr2: 6
Mean of arr1: 2.0
Standard deviation of arr1: 0.816496580927726
Mean of arr2: 5.0
Standard deviation of arr2: 0.816496580927726
```

Q4] Consider the following NumPy structured array:

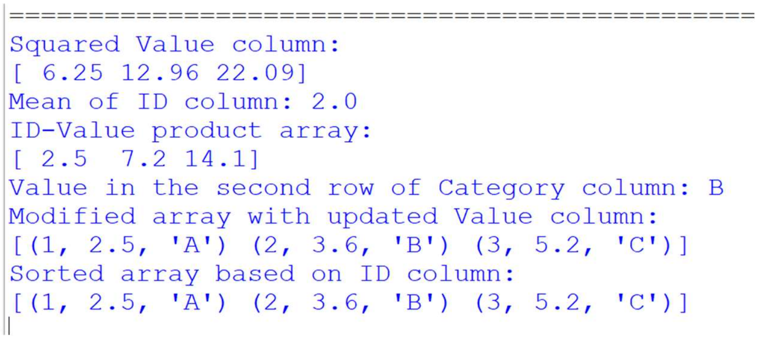
```
import numpy as np
data = np.array([(1, 2.5, 'A'), (2, 3.6, 'B'), (3, 4.7, 'C')],
            dtype=[('ID', int), ('Value', float), ('Category', 'U1')])
```

1. Computations on Arrays:
a) Compute the square of the 'Value' column in the structured array and assign it to a new variable. Print the new array.
b) Calculate the mean of the 'ID' column in the structured array. Print the result.
c) Perform element-wise multiplication of the 'ID' column with the 'Value' column and store the result in a new array. Print the new array.

2. NumPy's Structured Arrays:
a) Access and print the value in the second row of the 'Category' column.
b) Update the value in the third row of the 'Value' column to 5.2. Print the modified array to verify the change.
c) Sort the structured array based on the 'ID' column in ascending order. Print the sorted array.

```
#KHAN MOHD OWAIS RAZA
#20BCD7138
import numpy as np
data = np.array([(1, 2.5, 'A'), (2, 3.6, 'B'), (3, 4.7,
'C')],
                dtype=[('ID', int), ('Value', float),
('Category', 'U1')])
# 1. Computations on Arrays
# a) Compute the square of the 'Value' column
value_squared = data['Value'] ** 2
print("Squared Value column:")
print(value_squared)
# b) Calculate the mean of the 'ID' column
mean_id = np.mean(data['ID'])
print("Mean of ID column:", mean_id)
# c) Perform element-wise multiplication of 'ID' and
'Value' columns
id_value_product = data['ID'] * data['Value']
print("ID-Value product array:")
print(id_value_product)
# 2. NumPy's Structured Arrays
# a) Access and print the value in the second row of the
'Category' column
category_value = data[1]['Category']
print("Value in the second row of Category column:",
category_value)
```

```
# b) Update the value in the third row of the 'Value'
column to 5.2
data['Value'][2] = 5.2
print("Modified array with updated Value column:")
print(data)
# c) Sort the structured array based on the 'ID' column
in ascending order
sorted_data = np.sort(data, order='ID')
print("Sorted array based on ID column:")
print(sorted_data)
```



```
===============================================
Squared Value column:
[ 6.25 12.96 22.09]
Mean of ID column: 2.0
ID-Value product array:
[ 2.5  7.2 14.1]
Value in the second row of Category column: B
Modified array with updated Value column:
[(1, 2.5, 'A') (2, 3.6, 'B') (3, 5.2, 'C')]
Sorted array based on ID column:
[(1, 2.5, 'A') (2, 3.6, 'B') (3, 5.2, 'C')]
```