Aarya Arban

S11-07

Assignment No. 13 – Arrays Operation

```
import numpy as np
a = np.array([1, 2, 3, 4, 5,6,7,8])
print(a)
print(type(a))
print(len(a))
```

Output:

```
[1 2 3 4 5 6 7 8]
<class 'numpy.ndarray'>
8
```

```
#Use a tuple to create a NumPy array:
import numpy as np
a = np.array((1, 2, 3, 4, 5))
print(a)
print(type(a))
```

Output:

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

```
#Create a 0-D array with value 42
import numpy as np
a = np.array(42)
```

```
print(a)
print(type(a))
```

Output:

```
42
<class 'numpy.ndarray'>
```

```
#Create a 1-D array containing the values 1,2,3,4,5:
import numpy as np
a = np.array([1, 2, 3, 4, 5,6,7,8,9])
print(a)
print(len(a))
print(type(a))
```

Output:

```
[1 2 3 4 5 6 7 8 9]
9
<class 'numpy.ndarray'>
```

```
#Create a 2-D array containing two arrays with the values 1,2,3 and 4,5,6:
import numpy as np
a = np.array([[1, 2, 3,7], [4, 5, 6,9]])
print(a)
print(len(a))
```

Output:

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

#Create a 3-D array with two 2-D arrays, both containing two arrays with the values 1,2,3 and 4,5,6:

```
import numpy as np

a = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a)

print(len(a))
```

Output:

```
[[[1 2 3]

 [4 5 6]]

 [[1 2 3]

 [4 5 6]]]

2
```

#Check how many dimensions the arrays have:

```
import numpy as np

a = np.array(42)

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

c = np.array([[1, 2, 3], [4, 5, 6]])

d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)

print(b.ndim)

print(c.ndim)

print(d.ndim)
```

Output:

```
0

1

2
```

```
3
```

```
#Create an array with 5 dimensions and verify that it has 5 dimensions:
import numpy as np
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('number of dimensions:', arr.ndim)
```

Output:

```
[[[[[1 2 3 4]]]]]
number of dimensions: 5
```

```
#Access Array Elements
#Get the first element from the following array:
import numpy as np
a = np.array([1, 2, 3, 4])
print("Length of array")
print(len(a))
print(a[0])
print(a[1])
print(a[2])
```

Output:

```
Length of array
4
1
2
3
```

```
#Get third and fourth elements from the following array and add them.
import numpy as np
a = np.array([1, 2, 3, 4])
print(a[0] + a[1])
print(a[0] + a[2])
print(a[0] + a[3])
print(a[1] + a[2])
print(a[1] + a[3])
print("Sum Of Arrays")
print(a[2] + a[3])
```

Output:

```
Sum Of Arrays
3
4
5
5
6
7
```

```
import numpy as np
a = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print(a[0,0])
print('2nd element on 1st row: ', a[0, 1])
print(a[0,2])
print(a[0,3])
print(a[0,4])
print(a[1,0])
print(a[1,1])
print(a[1,2])
print(a[1,3])
print(a[1,4])
print(a)
```

Output:

```
1
2nd element on 1st row: 2
3
4
5
6
7
8
9
10
[[ 1 2 3 4 5]
 [ 6 7 8 9 10]]
```

```
#Print the last element from the 2nd dim:
import numpy as np
a = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Last element from 2nd dim: ', a[1, -1])
```

Output:

```
Last element from 2nd dim: 10
```

```
#Slice elements from index 1 to index 5 from the following array:

import numpy as np

a = np.array([1, 2, 3, 4, 5, 6, 7])

print(a[1:5])

print(a[0:6])

print(a[1:6])

#slice elements from particular index

print(a[3:])

print(a[4:])

print(a[2:])

print(a[:4])

print(a[:5])

#Negative Indexing

print(a[-3:-1])

#Return every other element from index 1 to index 5:

print(a[1:5:2])

#Return every other element from the entire array:

print(a[::2])

#2D array

a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

print(a)

print(a[1, 1:4])

print(a[0:2, 2])

#From both elements, slice index 1 to index 4 (not included), this will return a 2-D array:

print(a[0:2, 1:4])
```

Output:

```
[2 3 4 5]
[1 2 3 4 5 6]
[2 3 4 5 6]
[4 5 6 7]
[5 6 7]
[3 4 5 6 7]
[1 2 3 4]
[1 2 3 4 5]
[5 6]
[2 4]
[1 3 5 7]
[[ 1 2 3 4 5]
 [ 6 7 8 9 10]]
[7 8 9]
[3 8]
[[2 3 4]
 [7 8 9]]
```

```
#Data Types
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr.dtype)
```

Output:

```
int64
```

```
#Create an array with data type string:
import numpy as np
arr = np.array([1, 2, 3, 4], dtype='S')
print(arr)
print(arr.dtype)
```

Output:

```
[b'1' b'2' b'3' b'4']
|S1
```

```
#Create an array with data type 4 bytes integer:
import numpy as np
arr = np.array([1, 2, 3, 4], dtype='i4')
print(arr)
print(arr.dtype)
```

Output:

```
[1 2 3 4]
int32
```

```
#Change data type from float to integer by using 'i' as parameter value:
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
print(arr)
newarr = arr.astype('i')
print(newarr)
print(newarr.dtype)
```

Output:

```
[1.1 2.1 3.1]
[1 2 3]
int32
```

```
#Change data type from float to integer by using int as parameter value:
import numpy as np
arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype(int)
print(newarr)
print(newarr.dtype)
```

Output:

```
[1 2 3]
int64
```

```
#Change data type from integer to boolean:
import numpy as np
arr = np.array([1, 0, 3])
newarr = arr.astype(bool)
print(newarr)
print(newarr.dtype)
```

Output:

```
[ True False True]
bool
```

```
#Make a copy, change the original array, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
arr[0] = 42
print(arr)
print(x)
```

Output:

```
[42 2 3 4 5]
[1 2 3 4 5]
```

```
#Make a view, change the original array, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
arr[0] = 42
print(arr)
print(x)
```

Output:

```
[42 2 3 4 5]
[42 2 3 4 5]
```

```
#Make a view, change the view, and display both arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr.view()
```

```
x[0] = 31
print(arr)
print(x)
```

Output:

```
[31 2 3 4 5]
[31 2 3 4 5]
```

```
#Print the value of the base attribute to check if an array owns it's data or not:
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr)
x = arr.copy()
print(x)
y = arr.view()
print(y)
print(x.base)
print(y.base)
```

Output:

```
[1 2 3 4 5]
[1 2 3 4 5]
[1 2 3 4 5]
None
[1 2 3 4 5]
```

```
#Print the shape of a 2-D array:

import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])

print(arr.shape)
```

Output:

```
(2, 4)
```

```
#Create an array with 5 dimensions using ndmin using a vector with values 1,2,3,4

#and verify that last dimension has value 4:

import numpy as np

arr = np.array([1, 2, 3, 4], ndmin=5)

print(arr)

print('shape of array:', arr.shape)
```

Output:

```
[[[[[1 2 3 4]]]]]

shape of array: (1, 1, 1, 1, 4)
```

```
#Convert the following 1-D array with 12 elements into a 3-D array.

#The outermost dimension will have 2 arrays that contains 3 arrays, each with 2 elements:

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(2, 3, 2)

print(newarr)

newarr = arr.reshape(2, 2, 3)

print(newarr)
```

Output:

```
[[[ 1 2]
 [ 3 4]
 [ 5 6]]
 [[ 7 8]
 [ 9 10]
 [11 12]]]
[[[ 1 2 3]
 [ 4 5 6]]
 [[ 7 8 9]
 [10 11 12]]]
```

```
#Iterate on the elements of the following 1-D array:
import numpy as np
arr = np.array([1, 2, 3])
for x in arr:
print(x)
```

Output:

```
1
2
3
```

```
#Iterate on the elements of the following 2-D array:
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
```

```
print(x)
Output:
[1 2 3]
[4 5 6]
```

```
#Iterate on each scalar element of the 2-D array:
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
for y in x:
print(y)
Output:
1
2
3
4
5
6
```

```
#Iterate on the elements of the following 3-D array:
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr:
print(x)
Output:
[[1 2 3]
```

```
 [4 5 6]]
 [[ 7 8 9]
 [10 11 12]]
```

```
#Iterate down to the scalars:
import numpy as np
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr:
for y in x:
for z in y:
print(z)
```

Output:

```
1
2
3
4
5
6
7
8
9
10
11
12
```

```
#Iterate through the following 3-D array:
import numpy as np
arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in np.nditer(arr):
print(x)
Output:
1
2
3
4
5
6
7
8
```

```
#Join two arrays
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.concatenate((arr1, arr2))
print(arr)
Output:
[1 2 3 4 5 6]
```

```
#Join two 2-D arrays along rows (axis=1):
import numpy as np
```

```
arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
arr = np.concatenate((arr1, arr2), axis=1)
print(arr)
```

Output:

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

```
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.stack((arr1, arr2), axis=1)
print(arr)
```

Output:

```
[[1 4]
 [2 5]
 [3 6]]
```

```
#Stacking along rows
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.hstack((arr1, arr2))
print(arr)
```

Output:

```
[1 2 3 4 5 6]
```

```
#Stacking along Columns
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.vstack((arr1, arr2))
print(arr)
```

Output:

```
[[1 2 3]
 [4 5 6]]
```

```
#Stacking along Height (Depth)
import numpy as np
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
arr = np.dstack((arr1, arr2))
print(arr)
```

Output:

```
[[[1 4]
 [2 5]
 [3 6]]]
```

```
#Split the array in 3 parts:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr)
```

```
#Split the array in 4 parts:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 4)
print(newarr)
```

```
#Access the splitted arrays:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6])
newarr = np.array_split(arr, 3)
print(newarr[0])
print(newarr[1])
print(newarr[2])
```

```
#Split the 2-D array into three 2-D arrays.
import numpy as np
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11, 12]])
newarr = np.array_split(arr, 3)
print(newarr)
```

```
#Split the 2-D array into three 2-D arrays.
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3)
print(newarr)
```

```
#Split the 2-D array into three 2-D arrays along rows.
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.array_split(arr, 3, axis=1)
print(newarr)
```

```
#Use the hsplit() method to split the 2-D array into three 2-D arrays along rows.
import numpy as np
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12], [13, 14, 15], [16, 17, 18]])
newarr = np.hsplit(arr, 3)
print(newarr)
```

Output:

```
[array([1, 2]), array([3, 4]), array([5, 6])]
[array([1, 2]), array([3, 4]), array([5]), array([6])]
[1 2]
[3 4]
[5 6]
[array([[1, 2],
 [3, 4]]), array([[5, 6],
 [7, 8]]), array([[ 9, 10],
 [11, 12]])]
[array([[1, 2, 3],
 [4, 5, 6]]), array([[ 7, 8, 9],
 [10, 11, 12]]), array([[13, 14, 15],
 [16, 17, 18]])]
```

```
[array([[ 1],
 [ 4],
 [ 7],
 [10],
 [13],
 [16]]), array([[ 2],
 [ 5],
 [ 8],
 [11],
 [14],
 [17]]), array([[ 3],
 [ 6],
 [ 9],
 [12],
 [15],
 [18]])]
[array([[ 1],
 [ 4],
 [ 7],
 [10],
 [13],
 [16]]), array([[ 2],
 [ 5],
 [ 8],
 [11],
 [14],
 [17]]), array([[ 3],
 [ 6],
 [ 9],
 [12],
 [15],
 [18]])]
```

```
#Find the indexes where the value is 4:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 4, 4])
x = np.where(arr == 4)
print(x)
```

```
#Find the indexes where the values are even:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 0)
print(x)
```

```
#Find the indexes where the values are odd:
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8])
x = np.where(arr%2 == 1)
print(x)
```

```
#Find the indexes where the value 7 should be inserted:
import numpy as np
arr = np.array([6, 7, 8, 9])
x = np.searchsorted(arr, 7)
print(x)
```

```
#Find the indexes where the value 7 should be inserted, starting from the right:
import numpy as np
arr = np.array([6, 7, 8, 9])
x= np.searchsorted(arr, 7, side='right')
print(x)
```

```
#Find the indexes where the values 2, 4, and 6 should be inserted:
import numpy as np
arr = np.array([1, 3, 5, 7])
x = np.searchsorted(arr, [2, 4, 6])
print(x)
```

Output:

```
(array([3, 5, 6]),)
(array([1, 3, 5, 7]),)
(array([0, 2, 4, 6]),)
1
2
[1 2 3]
```