assignnment8-ui22cs03

October 20, 2023

NUMPY: NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

[2]:
```
#Let's Start with Importing NUMPY first
import numpy as np
```

The N-dimensional array (ndarray) An ndarray is a (usually fixed-size) multidimensional container of items of the same type and size. The number of dimensions and items in an array is defined by its shape, which is a tuple of N non-negative integers that specify the sizes of each dimension. The type of items in the array is specified by a separate data-type object (dtype), one of which is associated with each ndarray.

[ ]:
```
#Let's first create 3 x 3 Numpy multidimental array
x = np.array([[1,3,5],[2,4,6],[6,8,9]])
print("Here is the 3x3 Numpy matrix:\n",x)
#Note in Numpy array index start from 0
#Let's start Array Indexing part and access each elements below
print("\n Here is 1st Row elements:",x[0])
print("\n Here is 1st Row, 1st element:",x[0][0])
print("\n Here is last Row, last element:",x[2][2])
print("\n Here is second Row, second column element:",x[1,1])
print("\n Data type of created N dimenstional array",type(x))
print("\n Numpy Shape/Dimention size: ",x.shape)
print("\n Numpy Datatype: ",x.dtype)
```

```
Here is the 3x3 Numpy matrix:
 [[1 3 5]
 [2 4 6]
 [6 8 9]]

 Here is 1st Row elements: [1 3 5]

 Here is 1st Row, 1st element: 1

 Here is last Row, last element: 9
 Here is second Row, second column element: 4

 Data type of created N dimenstional array <class 'numpy.ndarray'>

 Numpy Shape/Dimention size:  (3, 3)

 Numpy Datatype:  int64
```

[ ]:
```
#We can trasnfor the array to data type
x.astype(float)
```

[ ]:
```
array([[1., 3., 5.],
       [2., 4., 6.],
       [6., 8., 9.]])
```

[ ]:
```
#Lets slice the above x array:
#Note : [1:2] means we access the 1st row and 2nd column index (rows,column)␣
 ↪format
print("Lets print the first Column all elements\n",x[:,0:1])
print("Let's print the 2nd Row all elements\n",x[1:2])
```

```
Lets print the first Column all elements
 [[1]
 [2]
 [6]]
Let's print the 2nd Row all elements
 [[2 4 6]]
```

[ ]:
```
#Negative slicing
print("Here we perfomed negative slicing by putting index -1 which access the␣
 ↪last element of numpy array: ",x[-1])
```

```
Here we perfomed negative slicing by putting index -1 which access the last
element of numpy array:  [6 8 9]
```

[ ]:
```
#Start Stop Step
series = np.array([1,3,5,8,12])
print(series[0:8:3])
print(series[::-1])
```

```
[1 8]
[12  8  5  2  1]
```

[ ]:
```
#lets generate the sequence of array by "np/arrange" attribute
a = np.arange(20)
print("Below is the new generated array from numpy\n")
print(a)
s = slice(1,15,3)
```

```
print("Let's use Start:Stop:Step Slicing on it")
print(a[s])
print("Sequence:",a[2:10])
```

```
Below is the new generated array from numpy

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
Let's use Start:Stop:Step Slicing on it
[ 1  4  7 10 13]
```

Creating a 1-D array

[3]:
```
l=[0,1,2,3,4,5]
arr1=np.array(l)
print(arr1)
type(arr1)
```

```
[0 1 2 3 4 5]
```

[3]:
```
numpy.ndarray
```

Arrays store only one kind of datatype

[4]:
```
l2=['s',0,2,3,4,8]
arr2=np.array(l2)
print(arr2)
```

```
['s' '0' '2' '3' '4' '8']
```

Creating a 2-D array

[5]:
```
l=[[1,2,3],[4,5,6]]
arr2=np.array(l)
print(arr2)
```

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

Creating a 3-D array

[6]:
```
l=[[[41,11,21],[23,22,32],[31,33,43]],[[4,44,54],[5,55,65],[6,66,76]],[[7,77,87],[8,88,98],[9,
arr3=np.array(l)
print(arr3)
```

```
[[[ 41  11  21]
  [ 23  22  32]
  [ 31  33  43]]

 [[  4  44  54]
  [  5  55  65]
  [   6   66   76]]

 [[   7   77   87]
  [   8   88   98]
  [   9   99 1300]]]
```

Random Function

```
[8]: np.random.seed(100)
print("\n 1-D \n")
a1=np.random.uniform(1,10,(2))
print("Random samples from uniform distribution: ", a1)
arr=np.random.rand(2)
print("Random values ", arr)
arr=np.random.random(2)
print("Random values from 0-1", arr)
arr=np.random.normal(0,5,2)
print("Random values with mean=0 & Standard Deviation=5: ", arr)
arr=np.random.randint(10,size=4)
print("Random integer values from 0-10", arr)
print("\n 2-D \n")
a1=np.random.uniform(1,10,(2,2))
print("Random samples from uniform distribution: \n ", a1)
arr=np.random.rand(2,2)
print("Random values \n", arr)
arr=np.random.random((2,2))
print("Random values from 0-1 \n", arr)
arr=np.random.normal(0,5,(2,2))
print("Random values with mean=0 & Standard Deviation=5: \n ", arr)
arr=np.random.randint(10,size=(4,4))
print("Random integer values from 0-10 \n", arr)
print("\n 3-D \n")
a1=np.random.uniform(1,10,(2,2,2))
print("Random samples from uniform distribution: \n", a1)
arr=np.random.rand(2,2,2)
print("Random values from \n", arr)
arr=np.random.random ((2,2,2))
print("Random values from 0-1 \n", arr)
arr=np.random.normal(0,5,(2,2,2))
print("Random values with mean=0 & Standard Deviation=5: \n ", arr)
arr=np.random.randint(10,size=(4,4,4))
print("Random integer values from 0-10 \n", arr)
```

```
 1-D

Random samples from uniform distribution:  [5.89064448 3.50532447]
Random values  [0.42451759 0.84477613]
Random values from 0-1 [0.00471886 0.12156912]
Random values with mean=0 & Standard Deviation=5:  [4.90660393 2.57109421]
Random integer values from 0-10 [1 0 8 4]

 2-D

Random samples from uniform distribution:
  [[6.61880699 5.7101297 ]
 [6.06066638 1.05235472]]
Random values
 [[0.30742321 0.95018431]
 [0.12665424 0.07898787]]
Random values from 0-1
 [[0.31135313 0.63238359]
 [0.69935892 0.64196495]]
Random values with mean=0 & Standard Deviation=5:
  [[-1.15142543  2.40462058]
 [-6.3345529   1.35502546]]
Random integer values from 0-10
 [[2 9 9 3]
 [2 5 8 1]
 [0 7 6 2]
 [0 8 2 5]]

 3-D

Random samples from uniform distribution:
 [[[4.42966064 5.76339204]
  [9.61189211 2.58239178]]

 [[2.06472553 8.76651103]
  [1.67380192 8.42585376]]]
Random values from
 [[[0.83613181 0.07539491]
  [0.01140079 0.04842057]]

 [[0.35712271 0.66569338]
  [0.01138961 0.10791777]]]
Random values from 0-1
 [[[0.9010131  0.79487876]
  [0.81146098 0.64027806]]

 [[0.62477951 0.14550751]
  [0.5702159  0.0651125 ]]]
Random values with mean=0 & Standard Deviation=5:
  [[[ -5.20566852  -3.43299676]
  [-12.79723791   4.73155275]]
 [[  3.95335826   3.40000947]
  [  3.5830291    2.38852423]]]
Random integer values from 0-10
 [[[9 0 0 5]
  [9 6 6 5]
  [6 4 7 3]
  [9 2 3 8]]

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

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

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

Array Analysis

[9]:
```
print("Number of Dimensions of arr1",arr1.ndim)
print("Size of arr1",arr1.size)
print("Shape of arr1",arr1.shape)
print("Shape of arr1",arr1.dtype)
print("Number of Dimensions of arr2",arr2.ndim)
print("Size of arr2",arr2.size)
print("Shape of arr2",arr2.shape)
print("Shape of arr2",arr2.dtype)
print("Number of Dimensions of arr3",arr3.ndim)
print("Size of arr3",arr3.size)
print("Shape of arr3",arr3.shape)
print("Shape of arr3",arr3.dtype)
```

```
Number of Dimensions of arr1 1
Size of arr1 6
Shape of arr1 (6,)
Shape of arr1 int64
Number of Dimensions of arr2 2
Size of arr2 6
Shape of arr2 (2, 3)
Shape of arr2 int64
Number of Dimensions of arr3 3
Size of arr3 27
```

```
Shape of arr3 (3, 3, 3)
Shape of arr3 int64
```

Array Indexing & Slicing

```
[10]: print("\n Array Indexing \n")
print(arr1[1])
print("\n")
print(arr2[1,2])
print("\n")
print(arr3[1,1,1])
print("\n Array Slicing \n")
print(arr1[0:1])
print("\n")
print(arr2[0:1,0:2])
print("\n")
print(arr3[0:1,0:2,0:2])
```

```

 Array Indexing 

1


6


55

 Array Slicing 

[0]


[[1 2]]


[[[41 11]
  [23 22]]]
```

Integer Indexing

```
[18]: print("\n Array Integer Indexing \n")
print(arr1[3])
print("\n")
print(arr2[0,1])
print("\n")
print(arr3[0,1,2])
```

```
 Array Integer Indexing

3



2



32
```

Filtering and Boolean Indexing

[19]:
```python
filter1= arr1 >3
print( filter1, "\n")
print(arr1[filter1])
filter2= arr2 >3
print(filter2, "\n")
print(arr2[filter2])
filter3= arr3 >30
print(filter3, "\n")
print(arr3[filter3])
```

```
[False False False False  True  True]

[4 5]
[[False False False]
 [ True  True  True]]

[4 5 6]
[[[ True False False]
  [False False  True]
  [ True  True  True]]

 [[False  True  True]
  [False  True  True]
  [False  True  True]]

 [[False  True  True]
  [False  True  True]
  [False  True  True]]]

[  41   32   31   33   43   44   54   55   65   66   76   77   87   88
   98   99 1300]
```

Broadcasting

[22]:
```python
array1 = np.array([1, 2, 3])
array2 = np.array([[1], [2], [3]])
```

```
result = array1 + array2 #broadcasting 1D array onto 2D array
num=10
print(num+array1) #broadcasting scalar onto 1D array
print (result)
```

```
[11 12 13]
[[2 3 4]
 [3 4 5]
 [4 5 6]]
```

[ ]:

```

```