ASSIGNMENT/TASK 4

NAME - SHWETA JHA

REG. ID.:GO_STP_102574

Question on Numpy--

(Q.1) Import the numpy package under the name np and Print the numpy version and the configuration
ANS.

```
import numpy as np
print(np.version)
print(np.show_config())
```

```
<module 'numpy.version' from '/usr/local/lib/python3.7/dist-packages/numpy/version.py'>
blas_mkl_info:
  NOT AVAILABLE
blis_info:
  NOT AVAILABLE
openblas_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
blas_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
lapack_mkl_info:
  NOT AVAILABLE
openblas_lapack_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
lapack_opt_info:
    libraries = ['openblas', 'openblas']
    library_dirs = ['/usr/local/lib']
    language = c
    define_macros = [('HAVE_CBLAS', None)]
None
```

2. Create a null vector of size 10

```
np.zeros(10)
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

3. Create Simple 1-D array and check type and check data types in array

```
sample = np.array([1,2,3,4,5])
print('Type:',type(sample))
print('Elements Type',sample.dtype)

     Type: <class 'numpy.ndarray'>
     Elements Type int64
```

4. How to find number of dimensions, bytes per element and bytes of memory used?

```
arr=np.array([[1,2,3],[4,5,6]])
print("Dimensionso of array:",arr.ndim)
print("Bytes used pe element:",arr.itemsize)
print("Bytes used in array:",arr.nbytes)

     Dimensionso of array: 2
     Bytes used pe element: 8
     Bytes used in array: 48
```

5.Create a null vector of size 10 but the fifth value which is 1

```
nv=np.zeros((2,5))
nv[0,4]=1
print(nv)

     [[0. 0. 0. 0. 1.]
      [0. 0. 0. 0. 0.]]
```

6.Create a vector with values ranging from 10 to 49

```
v=np.arange(10,50)
print(v)

     [10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
      34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49]
```

7.Reverse a vector (first element becomes last)

```
arr1=np.arange(1,9)
print(arr1)
print("Reversed Vector:",arr1[::-1])
```

```
[1 2 3 4 5 6 7 8]
Reversed Vector: [8 7 6 5 4 3 2 1]
```

8.Create a 3x3 matrix with values ranging from 0 to 8

```
mt=np.arange(0,9).reshape((3,3))
print(mt)
```

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

9.Find indices of non-zeros elements of array

```
ar2=np.array([1,2,0,0,4,0])
print(ar2)
print("Non-zero Elements of Array:",ar2[np.nonzero(ar2)])
```

```
[1 2 0 0 4 0]
Non-zero Elements of Array: [1 2 4]
```

10. Create a 3x3 identity matrix

```
i=np.identity(3)
print(i)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

11.)Create a 3x3x3 array with random values

```
rv=np.random.rand(3,3,3)
print(rv)
```

```
[[[0.34458463 0.17320578 0.85545272]
  [0.07255253 0.94219082 0.5445534 ]
  [0.52243613 0.82737793 0.40760648]]

 [[0.86276974 0.68156783 0.78965367]
  [0.08484262 0.82081789 0.86963518]
  [0.42888759 0.06523481 0.08687838]]

 [[0.76558391 0.91144683 0.80007715]
  [0.24624117 0.1498704  0.00663562]
  [0.03905424 0.07903171 0.72741848]]]
```

12.)Create a 10x10 array with random values and find the minimum and maximum values

```
t=np.random.rand(10,10)
print(t)
```

```
[[0.92470132 0.51563458 0.71662979 0.28449572 0.35986267 0.22549032
  0.29985183 0.27770893 0.95087299 0.56839188]
 [0.35519681 0.36687015 0.58072899 0.46403315 0.25764408 0.55078326
  0.31300495 0.80987283 0.71008342 0.38994125]
 [0.14462273 0.99998881 0.06468517 0.5772886  0.5240118  0.36853363
  0.9353951  0.73937899 0.83449744 0.170029  ]
 [0.8809669  0.81669598 0.64861876 0.76053893 0.58831335 0.88845341
  0.3425759  0.08781624 0.7163088  0.18208993]
 [0.47009158 0.27450908 0.74071365 0.39255211 0.41106837 0.12015327
  0.92957055 0.80451207 0.3112458  0.06018128]
 [0.81329568 0.35532899 0.6202084  0.51504806 0.83085065 0.80819989
  0.6613289  0.97552795 0.62497668 0.69991145]
 [0.40204378 0.37464397 0.7422431  0.87433552 0.49597805 0.05767201
  0.93605547 0.54816228 0.03298624 0.81132966]
 [0.25153273 0.80765022 0.47927192 0.69238818 0.54932993 0.98067965
  0.02320664 0.8156931  0.3606839  0.33288927]
 [0.6051641  0.21679842 0.73349576 0.49033201 0.94391978 0.09998581
  0.48543769 0.73963919 0.82284191 0.12291117]
 [0.7874406  0.88175127 0.92461054 0.41188368 0.80736524 0.92270395
  0.3797048  0.42517513 0.56272666 0.86486346]]
```

13.)Create a random vector of size 30 and find the mean value

```
print("Minimum Valu:",t.min())
print("maximum value:",t.max())
```

```
Minimum Valu: 0.023206644274859856
maximum value: 0.9999888142763228
```

14.)Create a 2d array with 1 on the border and 0 inside

```
rv1=np.random.rand(30)
print(rv1)
print("Mean is:",rv1.mean())
```

```
[0.17537152 0.19020569 0.96836888 0.00556677 0.51241538 0.61219438
 0.30878196 0.8348295  0.05071955 0.81129906 0.22762351 0.0743054
 0.66165621 0.3922207  0.61853995 0.93870522 0.01222345 0.28289056
 0.78865808 0.03197609 0.52787353 0.28875909 0.34572001 0.84095154
 0.91535487 0.71558522 0.14043949 0.80481188 0.72847286 0.93712   ]
Mean is: 0.4914546786146666
```

*15.)How to add a border (filled with 0's) around an existing array? *

```
arr2=np.ones((6,6))
arr2[1:-1,1:-1]=0
print(arr2)
```

```
[[1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

```
arr3=np.random.randint(5, size=(4,4))
arr4=np.pad(arr3, pad_width=1, constant_values=0)
print(arr3)
```

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

16.)How to Accessing/Changing specific elements, rows, columns, etc in Numpy array?

Example - [[1 2 3 4 5 6 7] [8 9 10 11 12 13 14]]

Get 13, get first row only, get 3rd column only, get [2, 4, 6], replace 13 by 20

```
ar=np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(ar[1,5])
print(ar[1,2])
print(ar[0,[1,3,5]])
ar[1,5]=20
print(ar)
```

```
13
10
[2 4 6]
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 20 14]]
```

17.)How to Convert a 1D array to a 2D array with 2 rows

```
ar1=np.linspace(1,6,6)
print(ar1)
print("Dimension of array:",ar1.ndim)

ar2=ar1.reshape((2,3))
print(ar2)
print("Dimension of array after reshaping:",ar2.ndim)
```

```
[1. 2. 3. 4. 5. 6.]
Dimension of array: 1
[[1. 2. 3.]
 [4. 5. 6.]]
Dimension of array after reshaping: 2
```

18.)Create the following pattern without hardcoding. Use only numpy functions and the below input array a.

Input:

a = np.array([1,2,3])` Desired Output:

> array([1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3])

```
a=np.array([1,2,3])
a2=a.repeat(3),np.tile(a,3)
np.concatenate(a2)
```

```
array([1, 1, 1, 2, 2, 2, 3, 3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3])
```

19.)Write a program to show how Numpy taking less memory compared to Python List?

```
import sys

a=1
list1=range(1000)
arr4=np.arange(1000)
print("Memory used by lists:",sys.getsizeof(a)*len(list1))
print("Memory used by Numpy array:",arr4.size*arr4.itemsize)
```

```
Memory used by lists: 28000
Memory used by Numpy array: 8000
```

20.)Write a program to show how Numpy taking less time compared to Python List?

```
import datetime
size=1000000
ll1=range(size)
ll2=range(size)

a1 = np.arange(size)
a2 = np.arange(size)

tic = datetime.datetime.now()
for i in range(size):
    list_product = li1[i]*li2[i]
  toc = datetime.datetime.now()
  print('Time Consumed by list:',toc-tic)

  tic=datetime.datetime.now()
  array_product = a1*a2
  toc = datetime.datetime.now()

  print('Time consumed by array:',toc-tic)

       Time Consumed by list: 0:00:00.360508
       Time consumed by array: 0:00:00.003667
```

✓ 0s completed at 10:23 AM