Task 4 (GOEDUHUB TECHNOLOGIES)

Reg ID: GO_STP_4

1. Import the numpy package under the name np and Print the numpy version and the configuration

```
import numpy as np
print("Version: {}".format(np.__version__))
print(np.show_config())
```

```
Version: 1.19.5
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

```
x= np.zeros(10)
x
```

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

```
a= np.arange(1,10,2)
a
```

```
array([1, 3, 5, 7, 9])
```

```
a.dtype
```

```
dtype('int64')
```

```
type(a)
```

```
numpy.ndarray
```

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

```
import numpy as np
a = np.array([5,10,15,20])
print(a)
print(type(a))
print(a.dtype)
```

```
[ 5 10 15 20]
<class 'numpy.ndarray'>
int64
```

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

```
import sys
a = np.arange(1,10).reshape(3,3)
print(f"dimension: {a.shape}")
print("Bytes per Element: {}".format(a.itemsize))
print("Memory size of numpy array is " +str(a.size*a.itemsize))


```

```
dimension: (3, 3)
Bytes per Element: 8
Memory size of numpy array is 72
```

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

```
a = (np.arange(10)==4).astype(int)
a
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0])
```

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

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

# Reverse a vector (first element becomes last)

print("\n Reversed vector\n",np.flipud(a))

```

```
[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]

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

7. .Reverse a vector (first element becomes last)

```
import numpy as np
a = np.array([10,20,30,40,24,17,4,8,13,12])
print(a[::-1])

```

```
[12 13  8  4 17 24 40 30 20 10]
```

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

```
a=np.arange(9).reshape(3,3)
a
```

```
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
```

9. Find indices of non-zero elements from [1,2,0,0,4,0]

```
a= np.array([1,2,0,0,4,0])

print("indexes of non-zero elements are:", np.nonzero(a))
```

```
indexes of non-zero elements are: (array([0, 1, 4]),)
```

10. Create a 3x3 identity matrix

```
np.identity(3)
```

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

11. Create a 3x3x3 array with random values

```
np.random.random((3,3,3))
```

```
array([[[0.14848343, 0.83829901, 0.54882122],
        [0.85029632, 0.84949129, 0.88719636],
        [0.28726921, 0.57028415, 0.50640523]],

       [[0.39536693, 0.73740667, 0.29897758],
        [0.85556182, 0.10424584, 0.53146837],
        [0.04283542, 0.14012119, 0.54116717]],

       [[0.48451911, 0.19203141, 0.06660058],
        [0.58111668, 0.53116524, 0.48680913],
        [0.60816985, 0.32154264, 0.15131149]]])
```

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

```
a=np.random.random((10,10))
print('max value',a.max())
print('min value',a.min())
# Create a random vector of size 30 and find the mean value
vector = np.random.random(30)
print("Mean value is",vector.mean())
vector
```

```
max value 0.9949897349014095
min value 7.106285170377369e-05
Mean value is 0.5004466700634077
array([0.47505155, 0.51896492, 0.51019581, 0.75079132, 0.84834351,
       0.36069626, 0.9053525 , 0.42523894, 0.02232105, 0.20449035,
       0.24544454, 0.61983763, 0.09408652, 0.65048023, 0.79260854,
       0.27294997, 0.39615947, 0.56874599, 0.85171174, 0.5926592 ,
       0.44851654, 0.61748994, 0.31317186, 0.36416273, 0.31564511,
       0.44763428, 0.73553166, 0.69921924, 0.67507359, 0.2908251 ])
```

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

```
import numpy as np
a = np.random.random(30)
print(a)
print("Mean value:",a.mean())
```

```
[0.83965197 0.43277093 0.82015169 0.97109402 0.69538056 0.7652756
 0.00434207 0.57214755 0.11990943 0.4416521  0.3517761  0.96535382
 0.45718063 0.44101876 0.30244776 0.75784625 0.60936128 0.33927153
 0.75830598 0.59823401 0.29682009 0.61491921 0.67366059 0.44154984
 0.99583125 0.75157108 0.81847429 0.00909895 0.09998633 0.41276754]
Mean value: 0.5452617062285429
```

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

```
a=np.ones((10,10))
print(a)
print(" \nArray with 1 on the border and 0 inside:")
a[1:-1, 1:-1] = 0
print(a)
```

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

Array with 1 on the border and 0 inside:
[[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]]
```

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

```
arr = np.arange(1,10).reshape(3,3)
np.pad(arr, pad_width=1, mode='constant', constant_values=0 )
```

```
array([[0, 0, 0, 0, 0],
       [0, 1, 2, 3, 0],
       [0, 4, 5, 6, 0],
       [0, 7, 8, 9, 0],
       [0, 0, 0, 0, 0]])
```

16. How to Accessing/Changing specic 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 rst row only, get 3rd column only, get [2, 4, 6], replace 13 by 20

```
import numpy as np
a = np.arange(1,15).reshape(2,7)
print(a)
print(a[1,5])
print(a[0,0:])
print(a[0:2,2])
print(a[0,1:6:2])
print(np.where(a==13, 20, a))
```

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

```
import numpy as np
a = np.array([10,20,30,40,50,60]).reshape(2,3)
print(a)
```

```
[[10 20 30]
 [40 50 60]]
```

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])

```
import numpy as np
a = np.array([1,2,3])
np.r_[np.repeat(a, 3), np.tile(a, 3)]
```

```
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 numpy as np
l = range(1000)
import sys
a = 10
print(sys.getsizeof(a)) #memory allocated to a
print(sys.getsizeof(a)*len(l))
a1 = np.arange(1000)
print(a1.size)
print(a1.size*a1.itemsize)
```

```
28
28000
1000
8000
```

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

```
import numpy as np
import sys
size = 1000000
l1 = range(size)
l2 = range(size)
start = time.time()
result = [(x+y) for x,y in zip(l1,l2)]
n1 = np.arange(size)
n2 = np.arange(size)
#list itemwise sum
print((time.time()-start)*1000)
#numpy array itemwise sum
start = time.time()
result1 = n1+n2
import time
print((time.time()-start)*1000)
```

```
101.1359691619873
1.4362335205078125
```

✓ 0s completed at 2:23 PM