GO_STP_5856 - Ashwin S

Assignment-4

```
import numpy as np
```

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

```
print("The numpy version is: ", np.__version__)
print("\t")
print("The numpy configuration is: ")
print(np.__config__.show())
```

```
The numpy version is:  1.19.5

The numpy configuration is:
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)
print(x)
```

```
[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

```
n = np.array([100, 200, 300, 400])
print(n)
print(n.dtype)
```

```
[100 200 300 400]
int64
```

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

```
n = np.array([100, 200, 300, 400])
n.ndim
```

```
1
```

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

```
x = np.zeros(10)
print(x)
x[4]=1
print(x)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[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)
```

```
[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)

```
a = np.asarray([1,2,3,4,5,6,7,8,9,10])
a[::-1]
```

```
array([10,  9,  8,  7,  6,  5,  4,  3,  2,  1])
```

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

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

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

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

```
print("Indices of non zero elements :",np.nonzero([1,2,0,0,4,0]))
```

```
Indices of non zero elements : (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.rand(3,3,3)
```

```
array([[[0.42960378, 0.16099102, 0.50592921],
        [0.87084065, 0.23620197, 0.94073536],
        [0.77965716, 0.77517454, 0.39810145]],

       [[0.08261043, 0.81271615, 0.5303792 ],
        [0.43211013, 0.61352487, 0.25478224],
        [0.46970828, 0.25019416, 0.73972747]],

       [[0.63392141, 0.47181306, 0.62033515],
        [0.02667348, 0.55229987, 0.19173192],
        [0.67069659, 0.01102647, 0.34208456]]])
```

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

```
n=np.random.rand(10,10)
print("The Maximum values is",np.max(n))
print("The Minimum values is",np.min(n))
```

```
The Maximum values is 0.9900691691445314
The Minimum values is 0.01254677289810291
```

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

```
Z = np.random.random(10)
m = Z.mean()
print (m)
```

```
0.46590050793687876
```

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

```
Z = np.ones((10,10))
Z[1:-1,1:-1]=0
```

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

```
x = np.ones((3,3))
print("Original array:")
print(x)
print("0 on the border and 1 inside in the array")
x = np.pad(x, pad_width=1, mode='constant', constant_values=0)
print(x)
```

```
Original array:
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
0 on the border and 1 inside in the array
[[0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0.]]
```

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

```
a = np.array([[ 1 ,2, 3, 4, 5, 6, 7], [ 8, 9, 10, 11, 12, 13, 14]])
print(a)
print("\t")
print(a[1,5])
```

```
print("\t")
print(a[0])
print("\t")
print(a[:,2])
print("\t")
print(a[0,:][1:6:2])
print("\t")
a[1,5] = 20
print(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

```
arr = np. array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
arr_2d = np. reshape(arr, (2, 5))
print(arr_2d)
```

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

```
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?

```
# importing numpy package
import numpy as np

# importing system module
import sys

# declaring a list of 1000 elements
S= range(1000)
```

```
S= range(1000)

# printing size of each element of the list
print("Size of each element of list in bytes: ",sys.getsizeof(S))

# printing size of the whole list
print("Size of the whole list in bytes: ",sys.getsizeof(S)*len(S))

# declaring a Numpy array of 1000 elements
D= np.arange(1000)

# printing size of each element of the Numpy array
print("Size of each element of the Numpy array in bytes: ",D.itemsize)

# printing size of the whole Numpy array
print("Size of the whole Numpy array in bytes: ",D.size*D.itemsize)
```

```
Size of each element of list in bytes:  48
Size of the whole list in bytes:  48000
Size of each element of the Numpy array in bytes:  8
Size of the whole Numpy array in bytes:  8000
```

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

```
# importing required packages
import numpy
import time

# size of arrays and lists
size = 1000000

# declaring lists
list1 = range(size)
list2 = range(size)

# declaring arrays
array1 = numpy.arange(size)
array2 = numpy.arange(size)

# list
initialTime = time.time()
resultantList = [(a * b) for a, b in zip(list1, list2)]

# calculating execution time
print("Time taken by Lists :",
  (time.time() - initialTime),
  "seconds")

# NumPy array
initialTime = time.time()
resultantArray = array1 * array2
```

```
# calculating execution time
print("Time taken by NumPy Arrays :",
  (time.time() - initialTime),
  "seconds")
```

```
Time taken by Lists : 0.131819486618042 seconds
Time taken by NumPy Arrays : 0.003958225250244141 seconds
```

✓ 0s completed at 1:46 PM