# Assignment - 4:

## By: Mohit Mahi

## Reg No.:- GO_STP_8131

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

In [ ]:
```
import numpy as np

print ("The numpy version is", np.__version__)
np.show_config()
```

```
The numpy version is 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)]
```

## 2. Create a null vector of size 10

In [ ]:
```
import numpy as np

Z = np.zeros(10)
print(Z)
```

```
[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

In [ ]:
```
import numpy as np

arr = np.array([10,20,30,40,50,60])
print("The type of array is", type(arr))
print("The type of array is", arr.dtype)
```

```
The type of array is <class 'numpy.ndarray'>
The type of array is int64
```

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

In [ ]:
```
import numpy as np

dimn = arr.ndim
print("Number of dimensions : ",dimn)
bytesperele = arr.itemsize
print("Bytes per element : ",bytesperele)
bytesused = arr.nbytes
print("Bytes of memory used : ",bytesused)
```

```
Number of dimensions :  1
Bytes per element :  8
Bytes of memory used :  48
```

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

In [ ]:
```
import numpy as np

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

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

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

In [ ]:
```
import numpy as np

vector = np.arange(10, 50)
print(f"Create a vector with values ranging from 10 to 49 are: \n{vector}")
```

```
Create a vector with values ranging from 10 to 49 are:
[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).

In [ ]:
```
import numpy as np

vector = np.arange(39, 50)
print(f"Original vector : {vector}")

rev_vector = vector[::-1]
print(f"Reversed vector : {rev_vector}")
```

```
Original vector : [39 40 41 42 43 44 45 46 47 48 49]
Reversed vector : [49 48 47 46 45 44 43 42 41 40 39]
```

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

In [ ]:
```
import numpy as np

Z = np.arange(9)
Z = Z.reshape(3,3)

print(f"3x3 matrix with values ranging from 0 to 8 :\n{Z}")
```

```
3x3 matrix with values ranging from 0 to 8 :
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

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

In [ ]:
```
import numpy as np

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

```
Indices of non-zero elements from [1,2,0,0,4,0] are : (array([0, 1, 4]),)
```

## 10. Create a 3x3 identity matrix.

In [ ]:
```
import numpy as np

Z = np.eye(3)
print(Z)
```

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

## 11. Create a 3x3x3 array with random values.

In [ ]:
```
import numpy as np

Z = np.random.random((3,3,3))
print(Z)
```

```
[[[0.00991462 0.94450975 0.21079387]
  [0.71328463 0.23408263 0.49761156]
  [0.8284198  0.55118692 0.76160298]]

 [[0.56287773 0.69340547 0.8831653 ]
  [0.72955363 0.71776147 0.1264022 ]
  [0.48038996 0.63337728 0.15621236]]

 [[0.16298819 0.1573379  0.96370428]
  [0.87342146 0.0806868  0.10118197]
  [0.28118563 0.42930238 0.42000566]]]
```

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

In [ ]:
```
import numpy as np

Z = np.random.random((10,10))
print(f"Array Created :\n{Z}")
Zmin, Zmax = Z.min(), Z.max()
print(f"In 10x10 array, the minimum random value is {Zmin} and Maximum random value i
s {Zmax}")
```

```
Array Created :
[[0.9647133  0.97781285 0.29344826 0.71686878 0.54586206 0.8734829
  0.91304336 0.11771377 0.00291778 0.11649864]
 [0.27085893 0.43417418 0.40811347 0.81432251 0.82255282 0.70425718
  0.55668267 0.25405075 0.45043628 0.83946999]
 [0.59814078 0.76258911 0.42310026 0.87473225 0.69122917 0.10585823
  0.83259954 0.78115217 0.53783463 0.81242998]
 [0.8618518  0.56232085 0.6679914  0.7546396  0.961027   0.26246513
  0.95490684 0.82875802 0.01494669 0.73202635]
 [0.46551596 0.88672546 0.50849305 0.4499002  0.8577835  0.99189016
  0.97199629 0.81586452 0.89603849 0.17315736]
 [0.96607438 0.958996   0.52146457 0.25365219 0.87500277 0.16219142
  0.0637128  0.38803907 0.92426449 0.2320587 ]
 [0.79957777 0.84224509 0.27389388 0.54969131 0.33198822 0.82303662
  0.02913309 0.73577699 0.92213305 0.42744401]
 [0.4322859  0.57222756 0.40373306 0.98244462 0.03334661 0.35049282
  0.72766723 0.39030974 0.54570062 0.2300691 ]
 [0.21489733 0.57073544 0.03939993 0.06354429 0.63063829 0.30102064
  0.86307545 0.18978314 0.30796173 0.86968884]
 [0.82804338 0.31260522 0.9522299  0.80750358 0.29878814 0.97033975
  0.15698155 0.23131996 0.79218203 0.90786351]]
In 10x10 array, the minimum random value is 0.002917784472824425 and Maximum random
value is 0.9918901608543239
```

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

In [ ]:
```
import numpy as np

Z = np.random.random(30)
print("The Vector is:\n", Z)

Zmean = Z.mean()
print(f"Mean of Z vector is : {Zmean}")
```

```
The Vector is:
 [0.25599126 0.16898488 0.11658057 0.11642584 0.81181251 0.72560024
 0.68719007 0.82431707 0.03648122 0.78381984 0.257645   0.28955688
 0.69945985 0.84426357 0.94587125 0.93802582 0.89069582 0.1927745
 0.45092155 0.56726267 0.95055234 0.86017034 0.20611045 0.44789453
 0.80142752 0.88495367 0.86170551 0.27899063 0.9787008  0.78911424]
Mean of Z vector is : 0.5887766818800738
```

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

In [ ]:
```
import numpy as np

Z = np.ones((10,10))

Z[1:-1, 1:-1] = 0
print(f"The Required 2-D array is :\n\n{Z}")
```

```
The Required 2-D array is :

[[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?

In [ ]:
```
import numpy as np

Z = np.ones((5,5))
Z = np.pad(Z, pad_width=1, mode='constant', constant_values=0)
print(Z)
```

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

## 16. How to Access/Change 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

In [ ]:
```python
import numpy as np

arr = np.array([np.arange(1,8),np.arange(8,15)])
print(f"Original Array :\n{arr}")

print("Getting Element 13 : ",arr[1,5])

print("Getting First Row only : ",arr[0,:])

print("Getting Third Column only : ",arr[:,2])

print("Printing 2,4,6 from array : ",arr[0,1:6:2])

arr[1,5] = 20
print("New array after replacing 13 by 20 :\n",arr)
```

```
Original Array :
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
Getting Element 13 :  13
Getting First Row only :  [1 2 3 4 5 6 7]
Getting Third Column only :  [ 3 10]
Printing 2,4,6 from array :  [2 4 6]
New array after replacing 13 by 20 :
 [[ 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?

In [ ]:
```python
import numpy as np

# 1D array
arr1d = np.arange(10)
print("1D array : ",arr1d)

#converting 1D array to 2D array with rows
arr2d = arr1d.reshape(2,5)
print("Modified 2D array :\n",arr2d)
```

```
1D array :  [0 1 2 3 4 5 6 7 8 9]
Modified 2D array :
 [[0 1 2 3 4]
 [5 6 7 8 9]]
```

## 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])

In [ ]:

```
import numpy as np

a = np.array([1,2,3])
print("Initial Array :", a)

print("Desired Output:")
new = np.append(np.repeat(a,3),np.tile(a,3))
new
```

```
Initial Array : [1 2 3]
Desired Output:
```

Out[ ]:

```
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?

In [ ]:

```
#memory allocation

l = range(1000)

import sys
print("LIST")
a=10
print("Memory allocated to one element : ",sys.getsizeof(a))   #memory allocated to a
print("Total Memory taken by list : ",sys.getsizeof(a)*len(l))

print("NUMPY ARRAY")
a1 = np.arange(1000)
print("Memory allocated to one element : ",a1.size)
print("Total Memory taken by Numpy array : ",a1.size*a1.itemsize)
#conclusion: numpy takes less memory allocation than list
```

```
LIST
Memory allocated to one element :  28
Total Memory taken by list :  28000
NUMPY ARRAY
Memory allocated to one element :  1000
Total Memory taken by Numpy array :  8000
```

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

In [ ]:

```python
#Numpy vs List -- speed
import numpy as np
import time
import sys

size = 1000000
l1 = range(size)
l2 = range(size)

n1 = np.arange(size)
n2 = np.arange(size)

#list itemwise sum
start = time.time()
result = [(x+y) for x,y in zip(l1,l2)]
print("Time taken by Python List : ",(time.time()-start)*1000)

#numpy array itemwise sum
start = time.time()
result1 = n1+n2
print("Time taken by NumPy Array : ",(time.time()-start)*1000)
```

```
Time taken by Python List :  126.57308578491211
Time taken by NumPy Array :  2.5625228881835938
```