1D Numpy array

In [137…

```
import numpy as np
def numpy_1D():
    #1D array ==> Rank 1 array
    array = np.array([1,2,3])
    print(array)
    print(array.shape)

    return array

def print_array(array):
    print("\t\t Printing using Loop\n")

    for i in range(0,len(array)):
        print("\t", array[i], end = "\t")

array = numpy_1D()
print_array(array)
```

```
[1 2 3]
(3,)
                 Printing using Loop

         1                2                3
```

2D numpy Array

In [138…

```
def numpy_2D():
    #2D array ==> Rank2 array
    array = np.array([[1,2,3],[4,5,6],[7,8,9]])
    print(array, "\n")
    print(array.shape)

    return array

def print_array(array):
    print("\t\t Printing using Loop\n")

    for i in range(0,len(array)):
        for j in range(0,len(array)):
            print("\t", array[i][j], end = "\t")

        print("\n")

array = numpy_2D()
print_array(array)
```

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

(3, 3)
                 Printing using Loop

         1                2                3
```

```
4                5                6

7                8                9
```

Print selective by Slicing, Rank2 array

In [92]:

```python
def print_selective(array):
    print("==> All Elements of 1st Column Are\n",array[:,1], "\n") # 1st Column will print, This is a Rank 1 result

    print("==> All Elements of 2nd row Are\n",array[2,:], "\n") # 2nd row will print, This is a Rank 1 result

    print("==> All Elements of 0th and 1st row\n",array[0:2,:], "\n") # 0th and 1st row, This is a Rank 2 result

    print("==> All Elements of 1st and 2nd col\n",array[:,1:3], "\n") # 1st and 2nd col, This is a Rank 2 result

    print("==> Elements [0][1], [0][2], [1][1], [1][2]\n", array[0:2,1:3], "\n") #Selective Elemenst, This is a Rank 2 result

print_selective(array)
```

```
==> All Elements of 1st Column Are
 [2 5 8]

==> All Elements of 2nd row Are
 [7 8 9]

==> All Elements of 0th and 1st row
 [[1 2 3]
 [4 5 6]]

==> All Elements of 1st and 2nd col
 [[2 3]
 [5 6]
 [8 9]]

==> Elements [0][1], [0][2], [1][1], [1][2]
 [[2 3]
 [5 6]]
```

Print all values of 1D array that are > 3

In [100…

```python
def approch_1(a):
    #By Creating a mask;
    b = a > 3 #This will create an array of bool where each index of b represent
                                                   #either condition is true / false for respective index
    print("Array Of Bool : ",b)

    print("By Creating array of bool : ",a[b], "\n") # All indexes that are true will be displayed

def approch_2(a):
    #By Directly running
    print("Direct Approch : ",a[a > 3]) # All indexes that are are true on condition will be displayed


a = np.array([1,5,3,4,0,2,9])
approch_1(a)
approch_2(a)
```

```
Array Of Bool :  [False  True False  True False False  True]
By Creating array of bool :  [5 4 9]

Direct Approch :  [5 4 9]
```

Add Element by Element a Numpy Matrix

In [111…

```
def add(a,b):
    print("Approch 1\n",a + b)
    print("\nApproch 2\n",np.add(a,b))

a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])

add(a,b) # -, *, / operations could be performed
```

```
Approch 1
 [[ 6  8]
 [10 12]]

Approch 2
 [[ 6  8]
 [10 12]]
```

Compute sum of each row and col of Numpy Matrix

In [118…

```
def sum_each_row(a):
    print("Sum of each row is    : ",np.sum(a, axis = 1)) # axis = 1 represent row

def sum_each_col(a):
    print("Sum of each column is : ",np.sum(a, axis = 0)) # axis = 0 represent col

a = np.array([[1,2,3],[3,4,6],[7,8,9]])

sum_each_row(a)
sum_each_col(a)
```

```
Sum of each row is    :  [ 6 13 24]
Sum of each column is :  [11 14 18]
```

Transpose of numpy matrxi

In [123…

```
a = np.array([[1,2,3],[3,4,6],[7,8,9]])

print(a, "\n")
print(a.T)

#Transpose of rank1 matrix does nothing
```

```
[[1 2 3]
 [3 4 6]
 [7 8 9]]

[[1 3 7]
 [2 4 8]
 [3 6 9]]
```

BroadCasting

In [125…

```
# we have to add [1,0,1] in each row of a 2D numpy array
# Fisrt approch is to make a same dimension matrix of [1,0,1] as a and add both
# Second approch is to use a loop
# Then comes a better approch BROADCASTING

a = np.array([[1,2,3],[3,4,6],[7,8,9]])
v = np.array([1,0,1])

print(a + v) #This will automatically map 1 0 1 as a 3D(since a is 3D) matrix and in each row of a
```

```
[[ 2  2  4]
 [ 4  4  7]
 [ 8  8 10]]
```

Create Numpy by default functions

In [134…

```
a = np.zeros((2,2)) #Creates a 2 x 2 matrix with all 0 values
print("With all zeros \n",a)

a = np.ones((2,2)) #Creates a 2 x 2 matrix with all 1 values
print("\nWith all ones \n",a)

a = np.full((3,2), 9) #Creates a 3 x 2 matrix with all 9 values
print("\nWith all default \n",a)

a = np.eye(3) #Creates a 3 x 3 identical matrix
print("\n n x n Identical Matrix \n",a)

a = np.random.random((2,3)) #Creates a 2x3 random matrix
print("\n n x n Random values Matrix \n",a)

b = np.empty_like(a) #Creates an empty matrix with same shape as a
print("\nEmpty Matrix \n",b)
```

```
With all zeros 
 [[0. 0.]
 [0. 0.]]

With all ones 
 [[1. 1.]
 [1. 1.]]

With all default 
 [[9 9]
 [9 9]
 [9 9]]

 n x n Identical Matrix 
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

 n x n Random values Matrix 
 [[0.24222002 0.84395713 0.52989856]
 [0.33942715 0.55702864 0.67557841]]

Empty Matrix 
```

```
[[0.24222002 0.84395713 0.52989856]
[0.33942715 0.55702864 0.67557841]]
```