Numpy 1

Ex.No. Date:

Aim:

To install Numpy package and do the basic functions

Description:

1. Declare the Numpy array
2. Create the array with full of zero values
3. Create the array with a scalar value filled
4. Create an array
5. with random values
6. Create a range values with step input
7. Do the reshape the array
8. Do the flattening the array
9. Get the shape, dimension, type and size of an array
10. Convert an array from one type to another

Program:

```
import numpy as np

a = np.array([[1, 2, 4j, [S, 8, 7]])

print ("Array created using passed list:\n", a)


# Creating a 3X4 array with all zeros

b = np.zeros((3, 4))

print ("\nAn array initialized with all zeros:\n", b)


#Create a constant value array of complex type

c - np.full((3, 3), 6)

print ("\nAn array initialized with all 6s.\n", c)


#Create an array with random values

d = np.random.random((2, 2))

print ("\nA random array:\n", d)
```

```
# Create a sequence of integers from 0 to 30 with steps of S e -
np.arange(0, 30, 5)
pi int ("\nA sequential array with steps of 5:\n", e)
```

```
# Printing shape of array
print("\nShape of array: ", nrr.shape)
```

```
t/ Pi inting type of  elenie!zts in ari a\
print( "\nArrny stores *I ments of type: ", ari .‹Ttyp*)
```

```
#converting datatypes from integer to float
newtype=arr.astype('f')
```

```
print("\nConverted array elemnets:\n",newtype)
print("Converted array type:",newtype.dtype)
```

OUTPUT:

```
Array created using passed list:
[(1  24]
 [S 8 7]]

An array initialized with all zeros:
\i* o. o. o.
{0. 0. 0. 0.]
[0. 0. 0. 0.]]

An array initialized with all 6s.
[[6 6 6]
 [6 6 6]
  6 6 6]]

A random array:
[(0.65863873 0.07137801]
 {0.8254548 0.91084018]]

A sequential array with steps of 5: [
0 S 10 1S 20 2S]
Originai ari ay:
[[1 2 3 4]
    242)
 [1201)
hesh3pecl  array[4,3] :
[[1 2 3]
 [4  5  2{
```

| 4 2 1

2 0 1]]

1 2 3 4]

[5 2 4 2]

[1 2 0 1]]

Fattened array:

[1 2 3 4 5 2 4 2 1 2 0 1

No. of dimensions: 2

Shape of array: (3, 4)

Array stores elements of type: int64

Converted array eiemnets:

[(1.2. 3. 4.]

[S. 2. 4. 2.]

[1.2. 0. 1.]]

Converted array type: float32

Numpy 2

Ex.No.

Date:

Aim:

To perform Numpy functions Description:

1. Perform different form of slicing
2. Assign index
3. Do the different join functions — join, horizontal join, vertical join and depth join
4. Do the splitting of an array

Program:

```
import numpy as np


# An exemplar array
arr = np.array([[-1, 2, 0, 4],
                4,-0.S, 6,0],
                [2.6, 0, 7, 8],
                [3, —7, 4, 2.0]])
```

0 Integer ai ray indexing exam ple

te'm p - arr[|0, 1, 2, 3], |3 2, 1, 0]j

```
print ("\nElements at indices (0, 3), (1, 2), (2, 1),"
                                "(3, 0):\n", temp)

4 boolean array indexing example Cond

= arr > 2

# cond is a boolean array

temp = arr[cond]

print ("\nEtement s greater than 2:\n", temp)

#Return every other element from the entire array:

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

print("\nOriginal array:",arr)

print("\nReturns every other elements in the array:arr[::2]:",arr[::2j)

#joining two array

arr1 = np.array([1, 2, 2])

arr2 = np.array([4, S, 6])

arr = np.concatenate((arr1, arr2))

print("\nOriginal arrays:\n",arr1,arr2)

print("\nJoined array:\n",arr)


arr = np.hstack((arr1, arr2))

print("\nHorizondal joining:\n",arr)

4Vertical join

arr = np.vstack((arr1, arr2))

print("\nVertical joining:\n",arr)

#Depth join

arr = np.dstack((arr1, arr2))
```

```
print("\nDepth joini ng:\n",arr)

#Splitting array
arr = np.array([1, 2, 2, 4, 5, 6]) newarr
= np.array split(arr, 3)
print("\nOriginal Array:\n",arr)
print("\nSptitted array:\n",newarr)
#Oisplaying splitted array in another form
print("\nSplitted array in another form:\n™)
print(newarr[0])
print(newarr[1])
print(nevvarr[2])
```

OUTPUT:

```
Original array:
[[—1. 2. 0. d. }
  4. —0.5 6. U. ]
 [ 2.6 0. 7.  8.
  3.  -7.  4.   2. ]]

Every other rows:nri [0:3:2]:
 j-1.  2.  0.  4.  ]
  2.6 0.  7.   8. ]j

Array with first 2 rows and 3 columns:arr[: 2,:3]:
[[—1. 2. 0. ]
  4. —0.5 6. ])

Elements at indices (0, 3), (1, 2), (2, 1),(2, 0):
{4. 6. 0. 3.]
```

Elements greater than 2:

|4. 4. 6. 2.6 7. 8. 2. 4.]

original array: [1 2 3 4 5 6 7]

Returns every other elements in the array:arr{: :2]: (1 3 S 7]

Original arrays:

[1 2 3] [4 5 6]

Joined array:

[1 2 3 4 5 6}

[illegible]

[1 2 3 4 5 6]

Vertical joining:

{[1 2 3]

[4 5 6]]

Depth joining:

[([1 4]

(2 5]

[3 6]]j

Original Array:

|1 2 3 4 5 6]

Splitted array:

[array([1, 2]), array([3, 4)), array([5, 6])}

Splitted array in another form:

[1 2]

|3 4]

|5 6]

Numpy 3

Ex.No.

Date:

Aim:

To perform Numpy functions Description:

1. Do the index retrieval
2. Do the basic operation with respect to index
3. Do the sorting operation of an array
4. Do the filtering the array value

```
import numpy as np

 inclexes where  the vnlrie  is 4: nri

- up.array{ [1, 2, 3. ^., 5, 4, 4])

print("Origi nal arrny:", ari )

x = np.where(arr = - ^.)

prints "\n Indexes where the value is 4:", x)
```

```
arr = np.array([[3, 2, 4], [S, 0, 1]])
print(”\nOriginal array:",arr)
print("\nSorted array:",np.sort(arr))

#Filter
arr = np.array([41, 42, 43, 44]) x
= [True, False, True, False]
newarr = arr[x]
print("\nOriginat array:",arr)
print("\nFilter index:",x)
print("\nFilter array:",newarr)

arr = np.array([41, 42, 43, 44])
filter arr = arr > 42
newarr = arr[filter arr]
print("\nOriginal array:",arr)
print("\nFilter array:condition- >42:",filter arr)
print("\nNew array:",newarr)
```

Output:

```
Original array: [1 2 3 4 S 4 4j
Indexes where the value is 4:(array((2, S, 6]),}

Origins i ari ay:(1 2 3 4 S 6 7 8j
Indexes where the vnlues are seven: {array([1, 3, 5, 7j),)

Original array: [3 2 0 1]
Sorted array: [0 1 2 3j
```

Original array: {[3 2 4]

{S 0 1]]

Sorted array: [[2 3 4]

{015)

original array:[41 42 43 44}

Filter index: {True, False, True, False]

Filter array: [41 43a

original array: {41 42 43 44]

Filter array:condition- >42: [False False True True]

New array:{43 44]

Numpy 4

Ex.No.

Date:

Aim:

To perform Numpy functions

Description:

1. Do the vector operations —addition, subtraction, multiplication and division
2. Do the scalar operation
3. Do the vectorize operation

Program:

```
import numpy as np
arr1=[10,20,30,40,50]
arr2=[2,4,5,8,10]
a=np.array(arr1)
b=np.array(arr2)
print("Original arrays")
print(a)
print(b)
print("\nVector addition")

print("\nVector subtraction”)
print(a-b)division") print(a/b)

print("\nvector
multiplication")
print{a”b) print(“\nVector
print("\nVectoi Dot prods ct" )
print{a.dot(b))
print("\nScalar multiplication")
sclr=5
```

```
print(" scalar value=",sclr)
print(" array=",a)
print(" result=",a*sclr)


def my_func(x, y):
                #"Retur r› x y If x •y t›therw‹se   etut    x+Y"


            return x - y

            ...

            return x + y
print( "\n\nNum(› y. v'eczorize method"I
orint(”IRetur n x-v 'f  x>y,  otherwise •e*uir  x+y)   •
arr 1='1ß.d.2fl]
ar‹ t -I 3, 30j
ver  fhnr = np.vertorizei my  funr)
orint("a r rayl : ".arr1)
print(' a ravJ:  ’.arr2\
prir t("result "’,ver fcint:iai r1,arr7) t
```

```
Original arrays
 10 20 30 4050]
  2 4 S 8 10]

Vector addition
[12 24 35 48 60]

Vector subtraction
  8 16 25 32 40]

Vector multiplicatlon
```

20 80 150 320 500]

Vector division

[5. 5. 6. 5. S.]

', 'c *. „ ,.,. n, ,D *, '-,

Scalar multiplication

scnlnr vnlrie- 5

oi ray- [10 20 30 40 50]

result- [50 100 150 200 250]