Experiment No.: 2

Title: Implementation of array processing using NumPy.

Objectives: 1. To learn NumPy arrays.

Theory:

Python is a great general-purpose programming language on its own, but with the help of a few popular libraries (NumPy, SciPy, Matplotlib) it becomes a powerful environment for scientfic computing.

NumPy:

Numpy (Numerical Python) is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and routines for processing these arrays. NumPy is often used along with packages like SciPy (Scientific Python) and Matplotlib (plotting library for python).

Operations using NumPy:

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

NumPy Arrays:

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

```
import numpy as np
np.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)
```

The above constructor takes the following parameters −

Sr.No.	Parameter & Description
1	**object** Any object exposing the array interface method returns an array, or any (nested) sequence.
2	**dtype** Desired data type of array, optional

3	**copy** Optional. By default (true), the object is copied
4	**order** C (row major) or F (column major) or A (any) (default)
5	**subok** By default, returned array forced to be a base class array. If true, sub-classes passed through
6	**Ndmin** Specifies minimum dimensions of resultant array

We can initialize numpy arrays from nested Python lists, and access elements using square brackets:

Examples:

import numpy as np	# import numpy and name it np
a = np.array([1, 2, 3])	# Create a rank 1 array
print(type(a))	# Prints "<class 'numpy.ndarray'>"
print(a.shape)	# Prints "(3,)"
print(a[0], a[1], a[2])	# Prints "1 2 3"
a[0] = 5	# Change an element of the array
print(a)	# Prints "[5, 2, 3]"
b = np.array([[1,2,3],[4,5,6]])	# Create a rank 2 array
print(b.shape)	# Prints "(2, 3)"
print(b[0, 0], b[0, 1], b[1, 0])	# Prints "1 2 4"
Numpy also provides many functions to create arrays:	
a = np.zeros((2,2))	# Create an array of all zeros
print(a)	# Prints "[[0. 0.] # [0. 0.]]"
b = np.ones((1,2))	# Create an array of all ones
print(b)	# Prints "[[1. 1.]]"
c = np.full((2,2), 7)	# Create a constant array

print(c)	# Prints "[[7. 7.] # [7. 7.]]"
d = np.eye(2)	# Create a 2x2 identity matrix
print(d)	# Prints "[[1. 0.] # [0. 1.]]"
e = np.random.random((2,2))	# Create an array with random values
print(e)	# Might print "[[0.91940167 0.08143941] # [0.68744134 0.87236687]]"
Creating an array from sub-classes:	
np.array(np.mat('1 2; 3 4'))	# Creates array([[1, 2], [3, 4]])
np.array(np.mat('1 2; 3 4'), subok=**True**)	# Creates matrix([[1, 2], [3, 4]])

Array indexing:

Numpy offers several ways to index into arrays: fields access, Slicing and advanced indexing.

Slicing: Slicing is the way to choose a range of values in the array. We use a colon (:) in square brackets.

Syntax: **[Start : Stop : Step]**

# slice items between indexes
a = np.arange(10) print a[2:5] # prints [2 3 4]
# slice items starting from index
a = np.array([[1,2,3],[3,4,5],[4,5,6]]) print a[1:] # prints [[3 4 5] # [4 5 6]]
Slicing can also include ellipsis (…) to make a selection tuple of the same length as the dimension of an array. If ellipsis is used at the row position, it will return an ndarray comprising of items in rows.
a = np.array([[1,2,3],[3,4,5],[4,5,6]]) print a[...,1] # this returns array of items in the second column
print a[1,...] # this returns array of all items from the second row
print a[...,1:] # this returns array of all items from column 1 onwards
Integer Indexing: selecting any arbitrary item in an array based on its Ndimensional index
x = np.array([[1, 2], [3, 4], [5, 6]]) y = x[[0,1,2], [0,1,0]] # includes elements at (0,0), (1,1) and (2,0) from the first array print y # [1 4 5]
Boolean Indexing: x = np.array([[0, 1, 2],[3, 4, 5],[6, 7, 8],[9, 10, 11]])
print x[x > 5] # prints [6 7 8 9 10 11]