Experiment - 1

Objective: Understanding data formats of NumPy: ndarrays (1D, 2D and 3D arrays), Array Creation Routines

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed.

Ndarray Object:

The most important object defined in NumPy is an N-dimensional array type called ndarray. It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.

Every item in an ndarray takes the same size of block in the memory. Each element in ndarray is an object of data-type object (called dtype).

The basic ndarray is created using an array function in NumPy as follows:

```
numpy.array
```

It creates an ndarray from any object exposing array interface, or from any method that returns an array.

```
numpy.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

Program 1: Write a Program to create and display a 1D NumPy array

```
#import the numpy library and aliasing as np
import numpy as np
arr = np.array ([1, 5, 6, 9])
print (arr)
```

Output:

```
[1 5 6 9]
```

Program 2: Write a Program to create and display a 2D NumPy array

```
import numpy as np
arr = np.array ([[1, 5], [6, 9]])
print (arr)
```

Output:

```
[[1 5]
 [6 9]]
```

Program 3: Write a Program to create and display a 3D NumPy array

```
import numpy as np
arr = np.array ([[[1, 5], [6, 9], [5,5]], [[0, 5], [6, 0], [7,5]]])
print (arr)
```

Output:

```
[[[1 5]
  [6 9]
  [5 5]]

 [[0 5]
  [6 0]
  [7 5]]]
```

Program 4: Write a program to create an array with 5 dimensions and verify that it has 5 dimensions

```
import numpy as np
arr = np.array ([1, 5, 6, 7, 9], ndmin = 5)
print (arr)
print ('Number of Dimensions =', arr.ndim)
```

Output:

```
[[[[[1 5 6 7 9]]]]]
Number of Dimensions = 5
```

ndarray.shape: This array attribute returns a tuple consisting of array dimensions, and number of array elements. It can also be used to resize the array.

Program 5: Write a program to create an array and print its dimensions using `shape`

```
import numpy as np
arr = np.array ([[1, 5, 6], [6, 7, 9]])
print (arr.shape)
```

Output:

```
(2, 3)
```

Program 6: Write a program to create an array. Resize the array using `shape`

```
import numpy as np
arr = np.array ([[1, 5, 6], [6, 7, 9]])
arr.shape = (3, 2)
print (arr)
```

Output:

```
[[1 5]
 [6 6]
 [7 9]]
```

Program 6: Write a program to create an array. Resize the array using `reshape`

```
import numpy as np
arr = np.array ([[1, 5, 6], [6, 7, 9]])
arr_new = arr.reshape (3, 2)
print (arr_new)
```

Output:

```
[[1 5]
 [6 6]
 [7 9]]
```

Program 6: Write a program to create an 1D array. Convert the array into a 3D array using `reshape`

```
import numpy as np
a = np.arange(24)
print (a)
print (a.ndim)
b = a.reshape (2, 3, 4)
print (b)
```

Output:

```
[0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
21 22 23]
1
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

ARRAY CREATION ROUTINES

A. **np.zeros** and **np.ones:** Creates arrays filled with zeros and ones respectively.

Program:

```
import numpy as np

# Creating arrays filled with zeros and ones
zeros_arr = np.zeros((2, 3))  # 2x3 array of zeros
ones_arr = np.ones((3, 2))    # 3x2 array of ones

# Printing the arrays
print("Zeros Array:")
print(zeros_arr)

print("\nOnes Array:")
print(ones_arr)
```

Output:

```
Zeros Array:
[[0. 0. 0.]
 [0. 0. 0.]]

Ones Array:
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

B. **np.arange** and **np.linspace:** Generates arrays with regularly spaced values.

Program:

```
import numpy as np

# Generating arrays with regularly spaced values
range_arr = np.arange(0, 10, 2)    # Array from 0 to 10 with
step 2
linspace_arr = np.linspace(0, 1, 5)  # 5 equally spaced values
between 0 and 1

# Printing the arrays
print("Range Array:")
print(range_arr)

print("\nLinspace Array:")
print(linspace_arr)
```

Output:

```
Range Array:
```

```
[0 2 4 6 8]

Linspace Array:
[0.   0.25 0.5  0.75 1.  ]
```

C. **np.eye** and **np.diag:** Creates identity matrices and extracting diagonals.

Program:

```
import numpy as np

# Creating identity matrices and extracting diagonals
identity_mat = np.eye(3)           # 3x3 identity matrix
diag_arr = np.diag([5, 2, 9])      # 3x3 diagonal matrix with
specified values

# Printing the arrays
print("Identity Matrix:")
print(identity_mat)

print("\nDiagonal Array:")
print(diag_arr)
```

Output:

```
Identity Matrix:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]

Diagonal Array:
[[5 0 0]
 [0 2 0]
 [0 0 9]]
```

D. **np.random** module: Generates arrays with random values.

Program:

```
import numpy as np

# Generating arrays with random values
rand_arr = np.random.rand(2, 2)    # 2x2 array with random
values between 0 and 1
randn_arr = np.random.randn(2, 2)  # 2x2 array with random
values from a normal distribution

# Printing the arrays
print("Random Array:")
print(rand_arr)

print("\nRandom Normal Array:")
```

```
print(randn_arr)
```

Output:

```
Random Array:
[[0.5453134  0.50643161]
 [0.55442449 0.3398527 ]]

Random Normal Array:
[[ 1.43451196 -1.20343859]
 [-1.10799431  1.18800686]]
```