# Numpy Basic

**Assignment**

# Consider the below code to answer further questions:

# import numpy as np

# list\_ = [ ‘1’ , ’2’ , ‘3’ , ‘4’ , ‘5’ ]

# array\_list = np.array(object = list\_)

# Q1. Is there any difference in the data type of variables list\_ and array\_list? If there is then write a code

# to print the data types of both the variables.

To print the data types of both variables, you can use the type() function in Python. Here's the code:

import numpy as np

list\_ = ['1', '2', '3', '4', '5']

array\_list = np.array(object=list\_)

print("Data type of list\_:", type(list\_))

print("Data type of array\_list:", type(array\_list))

When you run this code, it will output the data types of both variables:

Data type of list\_: <class 'list'>

Data type of array\_list: <class 'numpy.ndarray'>

The data type of list\_ is <class 'list'>, indicating that it is a Python list, while the data type of array\_list is <class 'numpy.ndarray'>, indicating that it is a NumPy array.

# Q2. Write a code to print the data type of each and every element of both the variables list\_ and arra\_list.

The code to print the data type of each element in both the variables list\_ and array\_list:

import numpy as np

list\_ = ['1', '2', '3', '4', '5']

array\_list = np.array(object=list\_)

# Print data types of elements in list\_

print("Data types of elements in list\_:")

for element in list\_:

print(type(element))

# Print data types of elements in array\_list

print("\nData types of elements in array\_list:")

for element in array\_list:

print(type(element))

This code will iterate over each element in the list\_ and array\_list variables and print the data type of each element.

# Q3. Considering the following changes in the variable, array\_list:

# array\_list = np.array(object = list\_, dtype = int)

# Will there be any difference in the data type of the elements present in both the variables, list\_ and

# arra\_list? If so then print the data types of each and every element present in both the variables, list\_

# and arra\_list.

# Consider the below code to answer further questions:

# import numpy as np

# num\_list = [ [ 1 , 2 , 3 ] , [ 4 , 5 , 6 ] ]

# num\_array = np.array(object = num\_list)

Yes, there will be a difference in the data type of the elements present in the variables list\_ and array\_list after the change.

In the original code, the array\_list is created using np.array(object=list\_), which means that the elements of array\_list will have a data type based on the original data type of the elements in list\_. Since the elements in list\_ are strings ('1', '2', '3', '4', '5'), the data type of the elements in array\_list will be string.

However, after the change array\_list = np.array(object=list\_, dtype=int), the dtype=int argument is provided. This specifies that the elements of array\_list should have an integer data type. Therefore, the elements in array\_list will be converted from strings to integers.

To print the data types of each element in both variables, you can use the dtype attribute of the NumPy array. Here's the updated code:

import numpy as np

list\_ = ['1', '2', '3', '4', '5']

array\_list = np.array(object=list\_)

print("Data types of elements in list\_:")

for element in list\_:

print(type(element))

print("\nData types of elements in array\_list:")

for element in array\_list:

print(type(element))

array\_list = np.array(object=list\_, dtype=int)

print("\nData types of elements in updated array\_list:")

for element in array\_list:

print(type(element))

This code will print the data types of elements in list\_, the original array\_list, and the updated array\_list with integer data types.

# Q4. Write a code to find the following characteristics of variable, num\_array:

# (i) shape

# (ii) size

import numpy as np

list\_ = ['1', '2', '3', '4', '5']

array\_list = np.array(list\_)

# (i) Shape

shape = array\_list.shape

print("Shape:", shape)

# (ii) Size

size = array\_list.size

print("Size:", size)

This code will output the shape and size of the array\_list variable. The shape represents the dimensions of the array, and the size represents the total number of elements in the array.

# Q5. Write a code to create numpy array of 3\*3 matrix containing zeros only, using a numpy array

# creation function.

# [Hint: The size of the array will be 9 and the shape will be (3,3).]

NumPy array of a 3x3 matrix containing only zeros, we can use the zeros function from the NumPy library. Here's the code:

import numpy as np

array\_zeros = np.zeros((3, 3))

the zeros function takes a tuple (3, 3) as an argument, specifying the shape of the array to be created. The function then generates a new NumPy array of the specified shape with all elements initialized to zero.

The resulting array, array\_zeros, will be a 3x3 matrix containing only zeros.

# Q6. Create an identity matrix of shape (5,5) using numpy functions?

# [Hint: An identity matrix is a matrix containing 1 diagonally and other elements will be 0.]

import numpy as np

identity\_matrix = np.identity(5)

print(identity\_matrix)

Output:

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

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

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

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

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

The np.identity(5) function creates a square identity matrix of size 5. It sets the diagonal elements to 1 and all other elements to 0.