

# Lab Assignment 7

## Fundamentals of Machine Learning

Name: M Jeyapathy

USN: 22BTRAD016

### Q.Implementing Numpy

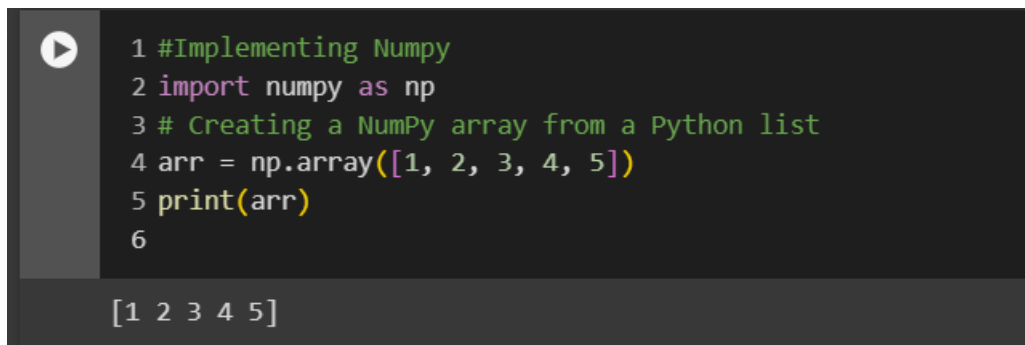
1, #Implementing Numpy

```
import numpy as np
```

```
# Creating a NumPy array from a Python list
```

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

```
print(arr)
```

A screenshot of a code editor with a dark background. On the left, there is a play button icon. The code is written in a syntax-highlighted font. The output of the code is displayed at the bottom of the editor.

```
1 #Implementing Numpy
2 import numpy as np
3 # Creating a NumPy array from a Python list
4 arr = np.array([1, 2, 3, 4, 5])
5 print(arr)
6
```

[1 2 3 4 5]

2, # Element-wise operations

```
arr1 = np.array([1, 2, 3])
```

```
print(arr1)
```

```
arr2 = np.array([4, 5, 6])
```

```
sum_result = arr1 + arr2
```

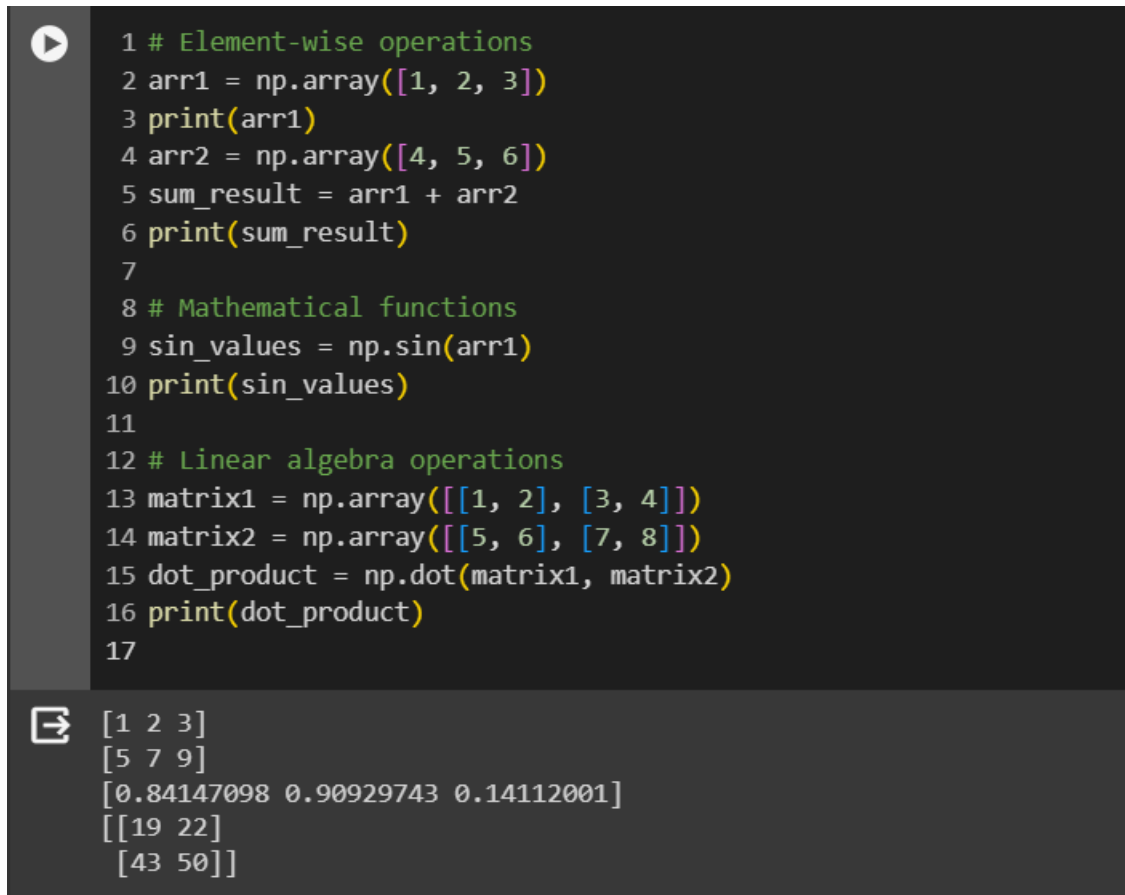
```
print(sum_result)
```

```
# Mathematical functions
```

```
sin_values = np.sin(arr1)
```

```
print(sin_values)
```

```
# Linear algebra operations
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
dot_product = np.dot(matrix1, matrix2)
print(dot_product)
```

A screenshot of a Jupyter Notebook interface. The top part shows a code cell with 17 lines of Python code. The code includes comments for 'Element-wise operations', 'Mathematical functions', and 'Linear algebra operations'. It defines two 1D arrays, arr1 and arr2, and calculates their sum. It also calculates the sine of arr1. Finally, it defines two 2x2 matrices, matrix1 and matrix2, and calculates their dot product. The bottom part shows the output of the code cell, which displays the sum of the arrays, the sine values, and the dot product matrix.

```
1 # Element-wise operations
2 arr1 = np.array([1, 2, 3])
3 print(arr1)
4 arr2 = np.array([4, 5, 6])
5 sum_result = arr1 + arr2
6 print(sum_result)
7
8 # Mathematical functions
9 sin_values = np.sin(arr1)
10 print(sin_values)
11
12 # Linear algebra operations
13 matrix1 = np.array([[1, 2], [3, 4]])
14 matrix2 = np.array([[5, 6], [7, 8]])
15 dot_product = np.dot(matrix1, matrix2)
16 print(dot_product)
17
```

```
[1 2 3]
[5 7 9]
[0.84147098 0.90929743 0.14112001]
[[19 22]
 [43 50]]
```

### 3, # Indexing

```
print(arr[0]) # Accessing the first element
```

# Slicing

```
print(arr[1:4]) # Accessing elements from index 1 to 3
```

```
1 # Indexing
2 print(arr[0]) # Accessing the first element
3
4 # Slicing
5 print(arr[1:4]) # Accessing elements from index 1 to 3
6
```

```
1
[2 3 4]
```

4, # Shape of an array

```
print(arr.shape)
```

# Reshaping

```
reshaped_arr = arr.reshape(5, 1)
```

```
print(reshaped_arr)
```

```
1 # Shape of an array
2 print(arr.shape)
3
4 # Reshaping
5 reshaped_arr = arr.reshape(5, 1)
6 print(reshaped_arr)
7
```

```
(5,)
[[1]
 [2]
 [3]
 [4]
 [5]]
```

**Github Link: <https://github.com/Jeyapathy/Machine-Learning>**

