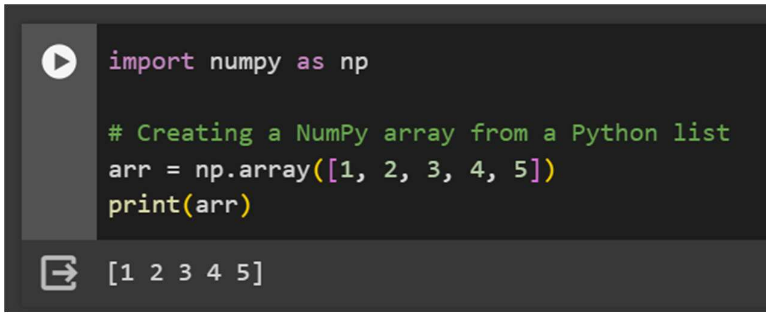
Fundamentals of Machine Learning
Lab Assignment – 7

Name: Niranjana J
USN: 22BTRAD027
Branch: CSE- AI & DE

Implementing Numpy.

- Numpy stands for Numerical Python. It is a package for general-purpose array processing. It offers tools for manipulating these arrays as well as a high-performance multidimensional array object. This is the core Python module for scientific computing. The program is open-source. Used for scientific computing and numerical operations
- provides support for large, multi-dimensional arrays and matrices and operations to be done on them
- used in areas such as data analysis, machine learning, and scientific research
- It provides features like indexing, slicing, array operations etc.

Importing the numpy library. An array is created using numpy and stored as arr.

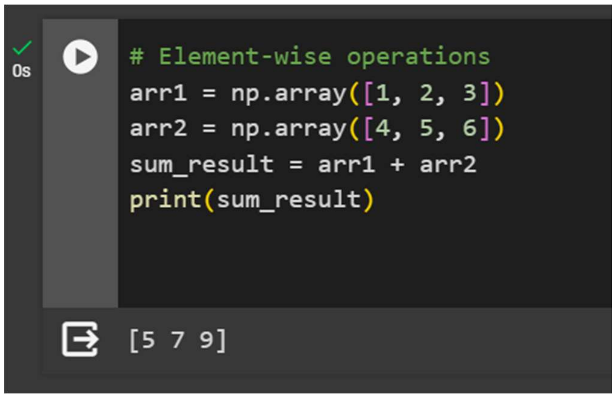


```
import numpy as np

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

```
[1 2 3 4 5]
```

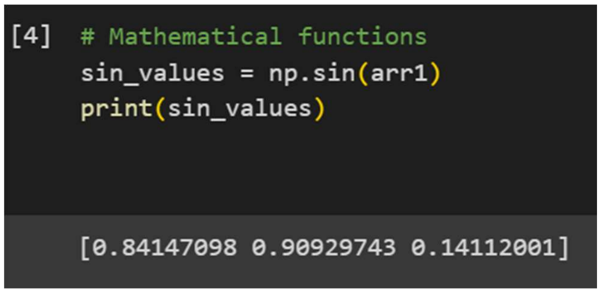
To find the sum of element sin 2 arrays, the below program can be used. Here, 2 arrays of same configuration is stored as two different arrays and added using the ‘+’ operator and stored in the variable sum_variable and printed out.



```
# Element-wise operations
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])
sum_result = arr1 + arr2
print(sum_result)
```

```
[5 7 9]
```

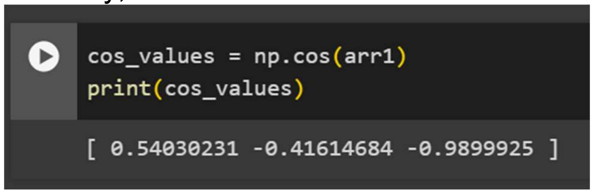
As for the mathematical operations, sin value of the numbers stored in the array can be found using the function np.sin(array_name). In this example, the sin values of the numbers saved in arr1 , [1,2,3] is displayed.



```
[4] # Mathematical functions
    sin_values = np.sin(arr1)
    print(sin_values)

    [0.84147098 0.90929743 0.14112001]
```

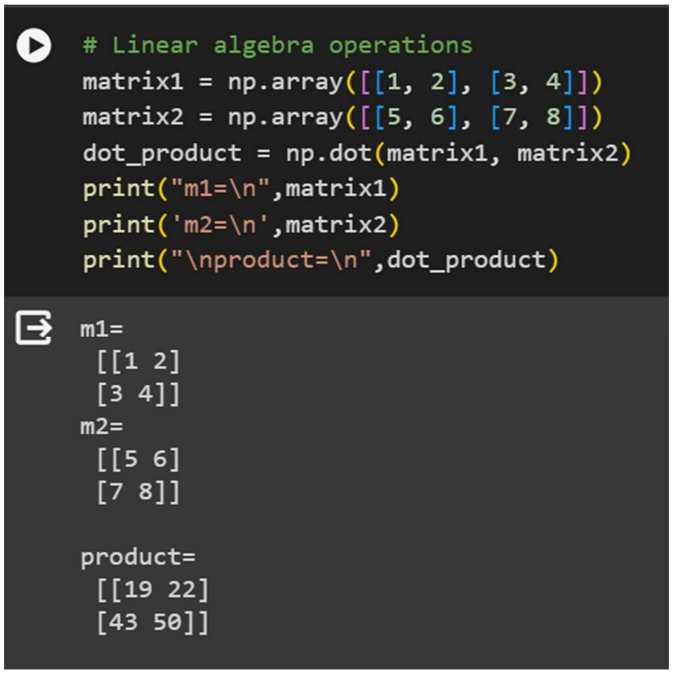
Similarly, cosine values can also be found.



```
cos_values = np.cos(arr1)
print(cos_values)

[ 0.54030231 -0.41614684 -0.9899925 ]
```

Numpy can also be used for matrix multiplication. Using the np.dot(matrix1, matrix2) function, the dot product of the required matrices can be calculated.

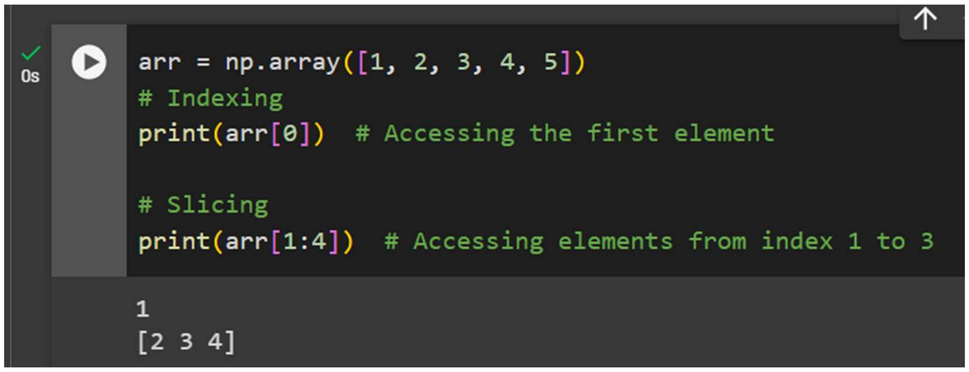


```
# Linear algebra operations
matrix1 = np.array([[1, 2], [3, 4]])
matrix2 = np.array([[5, 6], [7, 8]])
dot_product = np.dot(matrix1, matrix2)
print("m1=\n",matrix1)
print('m2=\n',matrix2)
print("\nproduct=\n",dot_product)

m1=
 [[1 2]
 [3 4]]
m2=
 [[5 6]
 [7 8]]

product=
 [[19 22]
 [43 50]]
```

Since indexing is an important feature of numpy, slicing can be done as required using the indices. In the first part of the snippet of code given below, the 0th element is fetched using its index. In the second part, the array is sliced which gives the elements present in 1,2 and 3 indices of the array.

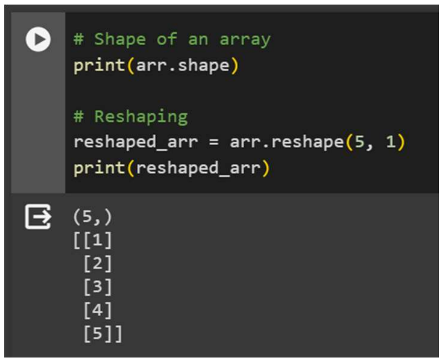


```
arr = np.array([1, 2, 3, 4, 5])
# Indexing
print(arr[0])  # Accessing the first element

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

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

The shape of the array can be displayed using the .shape function. To reshape the same array, .reshape(rows, columns) function can be used.

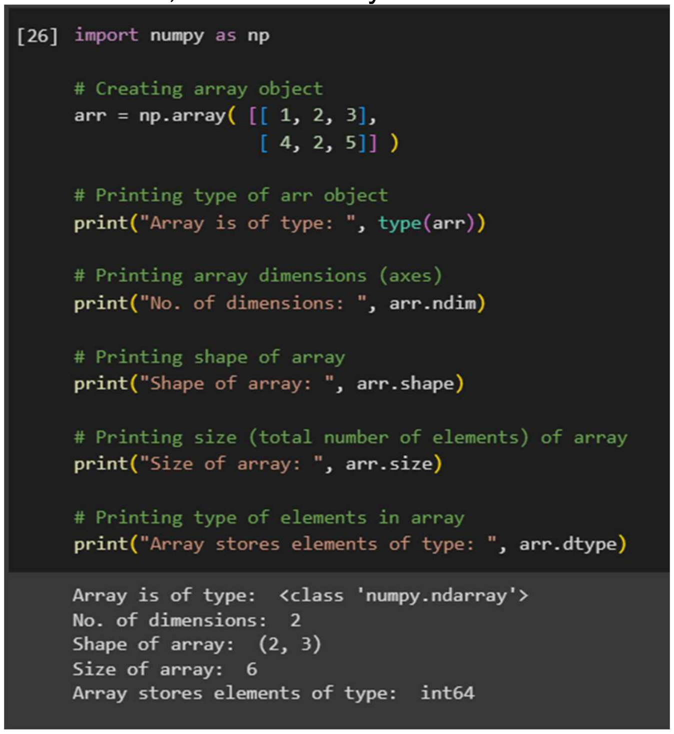


```
# Shape of an array
print(arr.shape)

# Reshaping
reshaped_arr = arr.reshape(5, 1)
print(reshaped_arr)
```

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

To conclude, the basic array details can be found using:



```
[26] import numpy as np

     # Creating array object
     arr = np.array( [[ 1, 2, 3],
                      [ 4, 2, 5]] )

     # Printing type of arr object
     print("Array is of type: ", type(arr))

     # Printing array dimensions (axes)
     print("No. of dimensions: ", arr.ndim)

     # Printing shape of array
     print("Shape of array: ", arr.shape)

     # Printing size (total number of elements) of array
     print("Size of array: ", arr.size)

     # Printing type of elements in array
     print("Array stores elements of type: ", arr.dtype)
```

```
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of array:  (2, 3)
Size of array:  6
Array stores elements of type:  int64
```

Github link:
https://github.com/niranjana628/ML