
 SOMAIYA VIDYAVIHAR UNIVERSITY K J Somaiya College of Engineering	K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities	
<p> Name: Shreyans Tatiya Batch: C5_3 Roll No.: 16010123325 Experiment / assignment / tutorial No. Grade: AA / AB / BB / BC / CC / CD /DD Signature of the Staff In-charge with date: </p>		

Title : NumPy library of Python

AIM: To explore the Numpy library of Python

Expected OUTCOME of Experiment:

CO2: Use Numpy Library functions

Resource Needed: Python IDE

Theory:

NumPy : A Python library used for working with arrays.

- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy stands for Numerical Python.
- The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy.

For Installation of NumPy:

- pip install numpy

Example

```
import numpy
```

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

```
print(arr)
```

output:?

For Creation of NumPy ndarray Object:

- NumPy is used to work with arrays. The array object in NumPy is called ndarray.
- We can create a NumPy ndarray object by using the array() function.
Example:

```
import numpy as np
arr=np.array([1, 2, 3, 4, 5])
print(arr)
print(type(arr))
```

Creating ndarrays:

```
array = np.array([[0,1,2],[2,3,4]])
```

output:

```
[[0 1 2]
 [2 3 4]]
```

```
array = np.zeros((2,3))
```

```
[[0. 0. 0.]
 [0. 0. 0.]]
```

```
array = np.ones((2,3))
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
```

```
array = np.eye(3)
```

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

```
array = np.arange(0, 10, 2)
```

```
[0, 2, 4, 6, 8]
```

```
array = np.random.randint(0, 10, (3,3))
```

```
[[6 4 3]
 [1 5 6]
 [9 8 5]]
```

Slicing arrays

Slicing in python means taking elements from one given index to another given index.

We pass slice instead of index like this: [start:end].

We can also define the step, like this: [start:end:step].

Arithmetic with NumPy Arrays:

Any arithmetic operations between equal-size arrays applies the operation element-wise

```
arr = np.array([[1., 2., 3.], [4., 5., 6.]])
```

```
print(arr)
```

```
[[1. 2. 3.]
 [4. 5. 6.]]
```

```
print(arr * arr)
```

```
[[ 1.  4.  9.]
 [16. 25. 36.]]
```

```
print(arr - arr)
[[0.  0.  0.]
 [0.  0.  0.]]
```

- **Shape of an Array**

The shape of an array is the number of elements in each dimension.

- **Reshaping arrays**

Reshaping means changing the shape of an array.

The shape of an array is the number of elements in each dimension.

By reshaping we can add or remove dimensions or change number of elements in each dimension.

- **Iterating Arrays**

Iterating means going through elements one by one.

As we deal with multi-dimensional arrays in numpy, we can do this using basic for loop of python. If we iterate on a 1-D array it will go through each element one by one.

- **Joining NumPy Arrays**

Joining means putting contents of two or more arrays in a single array.

In SQL we join tables based on a key, whereas in NumPy we join arrays by axes.

We pass a sequence of arrays that we want to join to the concatenate() function, long with the axis. If axis is not explicitly passed, it is taken as 0.

- **Splitting NumPy Arrays**

Splitting is reverse operation of Joining.

Joining merges multiple arrays into one and Splitting breaks one array into multiple.

We use array_split() for splitting arrays, we pass it the array we want to split and the number of splits.

- **NumPy Searching Arrays**

You can search an array for a certain value, and return the indexes that get a match.

To search an array, use the where() method.

- **Sorting Arrays**

Sorting means putting elements in an ordered sequence.

Ordered sequence is any sequence that has an order corresponding to elements, like numeric or alphabetical, ascending or descending.

The NumPy ndarray object has a function called sort(), that will sort a specified array.

- **NumPy Filter Array**

Getting some elements out of an existing array and creating a new array out of them is called filtering. In NumPy, you filter an array using a boolean index list.

1.Problem statement:



Python Code	Output
<pre>import numpy as np arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]]) print(arr.shape)</pre>	(2, 4)
<pre>import numpy as np arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]) newarr = arr.reshape(4, 3) print(newarr)</pre>	<pre>[[1 2 3] [4 5 6] [7 8 9] [10 11 12]]</pre>
<pre>import numpy as np arr = np.array([1, 2, 3]) for x in arr: print(x)</pre>	<pre>1 2 3</pre>
<pre>import numpy as np arr1 = np.array([1, 2, 3]) arr2 = np.array([4, 5, 6]) arr = np.concatenate((arr1, arr2)) print(arr)</pre>	[1 2 3 4 5 6]
<pre>import numpy as np arr = np.array([1, 2, 3, 4, 5, 6]) newarr = np.array_split(arr, 3) print(newarr)</pre>	<pre>[array([1,2]), array([3,4]), array([5, 6])]</pre>
<pre>import numpy as np arr = np.array([1, 2, 3, 4, 5, 4, 4]) x = np.where(arr == 4) print(x)</pre>	(array([3, 5, 6]),)
<pre>import numpy as np arr = np.array([3, 2, 0, 1]) print(np.sort(arr))</pre>	[0 1 2 3]
<pre>import numpy as np arr = np.array([41, 42, 43, 44]) x = [True, False, True, False] newarr = arr[x] print(newarr)</pre>	[41 43]

2. Write a python program to calculate the sum of all columns in a 2D NumPy array.
3. Create two NumPy arrays representing monthly high and low temperatures for a year. Calculate the monthly average temperatures, the overall average high and low temperatures, and identify the months with the highest and lowest average temperatures.

Books/ Journals/ Websites referred:

1. Reema Thareja, *Python Programming: Using Problem Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A modular Approach*, Pearson India, Second Edition 2018, India

Implementation details:

2.

```
import numpy

def colsum(arr, n, m):
    for i in range(n):
        sum = 0
        for j in range(m):
            sum += arr[j][i]
        print(sum, end = " ")

# creating the 2D Array
TwoDList = [[1, 2, 3], [4, 5, 6],
             [7, 8, 9], [10, 11, 12]]
TwoDArray = numpy.array(TwoDList)

print("2D Array:")
print(TwoDArray)

print("\nColumn-wise Sum:")
colsum(TwoDArray, len(TwoDArray[0]), len(TwoDArray))
```

3.

```
import numpy as np

# Monthly high and low temperatures for a year
high_temps = np.array([56, 53, 50, 53, 55, 58, 61, 63, 64, 63, 60])
low_temps = np.array([44, 43, 42, 42, 43, 44, 44, 43, 41, 41, 41])

# Monthly average temperatures
monthly_average_temps = (high_temps + low_temps) / 2

# Overall average high and low temperatures
overall_average_high = np.mean(high_temps)
overall_average_low = np.mean(low_temps)

# Identify the months with the highest and lowest average temperatures
highest_month_average = np.argmax(monthly_average_temps)
lowest_month_average = np.argmin(monthly_average_temps)

# Month names for easier reading
month_names = np.array(["January", "February", "March", "April", "May", "June",
                        "July", "August", "September", "October", "November", "December"])

print(f"Monthly average temperatures: {monthly_average_temps}")
print(f"Overall average high temperature: {overall_average_high}")
print(f"Overall average low temperature: {overall_average_low}")
print(f"Month with the highest average temperature: {month_names[highest_month_average]}")
print(f"Month with the lowest average temperature: {month_names[lowest_month_average]}")
```

Output(s):

2.

```
2D Array:
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

```
Column-wise Sum:
22 26 30
```

3.

```
Monthly average temperatures: [50.  48.  46.  47.5 49.  51.  52.5 53.  52.5 52.  50.5]
Overall average high temperature: 57.81818181818182
Overall average low temperature: 42.54545454545455
Month with the highest average temperature: August
Month with the lowest average temperature: March
```

Conclusion:

In conclusion, my NumPy experiment showcased the powerful and efficient nature of the library for scientific computing and data manipulation. With the ability to handle large data sets, manipulate arrays, and perform vectorized operations, NumPy truly is the fundamental library for scientific computing in Python.

Post Lab Descriptive Questions

1. Generate a random integer from 0 to 100 using NumPy random function?

Ans:

You can use the numpy random function randint() to generate a random integer

```
1 import numpy as np
2
3 random_integer = np.random.randint(0, 101)
4 print(random_integer)
```

In this code, randint() takes two arguments, the lower limit (0) and the upper limit (101), where the upper limit is exclusive. The generated random integer will be in the range of 0 to 100, inclusive.

2. Explain the slicing of 2-D Array

Ans:

Slicing is a Python technique that enables a developer to retrieve or modify subsets of an existing data structure.

In the case of 2-D arrays (also known as matrices or lists of lists), slicing can be performed in both the row and column directions.

For example, consider the following 2-D array:

```
array_2D = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

If we want to get the subarray from the first row and second column to the third row and third column, we can do it like this:

```
subarray = array_2D[0:3, 1:3]
```

The slicing syntax in Python is [start:stop:step], where start is the starting index, stop is the ending index, and step is the number of indices to skip between each successive element.

In the example above, we are slicing from index 0 to 3 (exclusive) in the row direction and from index 1 to 3 (exclusive) in the column direction.

This operation would result in the following subarray:

```
[[2, 3], [5, 6], [8, 9]]
```

Remember that in Python, the indexing starts from 0. If we omit the start index, it defaults to 0. Similarly, if we omit the stop index, it defaults to the maximum possible index. If we omit the step, it defaults to 1.

Here's a step-by-step breakdown of how slicing works:

1. Start at the start index (inclusive).
2. Keep adding the step to the current index until the next index exceeds the stop index (exclusive).
3. Include all the indices that meet these conditions in the slice.

This means that in our example, the elements at indices (0, 1), (0, 2), (1, 1), (1, 2), (2, 1), and (2, 2) are included in the slice.

Date: _____

Signature of faculty in-charge