|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

|  |  |
| --- | --- |
| **Title:** NumPy library of Python | |
| **Name: Dhruv Pankhania**  **Div: C4-1** | **Roll No.: 16010124216** |

**AIM:** To explore the Numpy library of Python

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Expected OUTCOME of Experiment:**

**CO5:** Use Numpy Library functions

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resource Needed: Python IDE**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Theory:**

**NumPy: A** Python library used for working with arrays.

**Batch:**

**Experiment / Grade: AA /**

• It also has functions for working in the domain of linear algebra, Fourier transform,

and matrices.

• NumPy stands for Numerical Python.

**Signature of**

• 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 the 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:

PP Sem I/ 2024-2025

|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

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 apply the operation element-wise** arr = np.array([[1., 2., 3.], [4., 5., 6.]])

print(arr)

[[1. 2. 3.]

[4. 5. 6.]]

PP Sem I/ 2024-2025

|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

print(arr \* arr)

[[ 1. 4. 9.]

[16. 25. 36.]]

print(arr - arr)

[[0. 0. 0.]

[0. 0. 0.]]

● **Shape of an Array**

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

● **Reshaping arrays**

o Reshaping means changing the shape of an array.

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

o By reshaping we can add or remove dimensions or change the number of elements

o in each dimension.

● **Iterating Arrays**

o Iterating means going through elements one by one.

o As we deal with multi-dimensional arrays in numpy, we can do this using the basic

o for loop of Python. If we iterate on a 1-D array it will go through each element

o one by one.

● **Joining NumPy Arrays**

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

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

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

0.

● **Splitting NumPy Arrays**

o Splitting is the reverse operation of Joining.

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

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

● **NumPy Searching Arrays**

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

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

● **Sorting Arrays**

o Sorting means putting elements in an ordered sequence.

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

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

PP Sem I/ 2024-2025

|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

● **NumPy Filter Array**

o 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** |
| 1 | import numpy as np  ?  arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])  print(arr.shape) | (2, 4) |
| 2 | 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) | [[ 1 2 3]  [ 4 5 6]  [ 7 8 9]  [10 11 12]] |
| 3 | import numpy as np  ?  arr = np.array([1, 2, 3])  for x in arr:  print(x) | 1  2  3 |
| 4 | import numpy as np    arr1 = np.array([1, 2, 3])  arr2 = np.array([4, 5, 6])  arr = np.concatenate((arr1, arr2))  print(arr) | [1 2 3 4 5 6] |
| 5 | import numpy as np  ?  arr = np.array([1, 2, 3, 4, 5, 6])  newarr = np.array\_split(arr, 3)  print(newarr) |  |
| 6 | import numpy as np  ?  arr = np.array([1, 2, 3, 4, 5, 4, 4])  x = np.where(arr == 4)  print(x) | (array([3, 5, 6]),) |
| 7 | import numpy as np  ?  arr = np.array([3, 2, 0, 1])  print(np.sort(arr)) | [0 1 2 3] |
| 8 | import numpy as np  ?  arr = np.array([41, 42, 43, 44])  x = [True, False, True, False]  newarr = arr[x]  print(newarr) | [41 43] |

PP Sem I/ 2024-2025

|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

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, and 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:  
Q2:**import numpy as np  
a = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])  
b = np.zeros(a.shape[1], *dtype*="int32")  
for i in range(a.shape[1]):  
  b[i] = a[0, i] + a[1, i]  
print(b)

**Q3:**import numpy as np  
monthly\_high = np.array([29, 32, 35, 40, 38, 34, 33 ,33, 34, 31, 29])  
montly\_low = np.array([15, 17, 21, 25, 28, 27, 26, 25, 24, 22, 18, 15])  
monthly\_avg = np.zeros(11)  
for i in range(11):  
  monthly\_avg[i] = (monthly\_high[i] + montly\_low[i])/2  
print(monthly\_avg)

PP Sem I/ 2024-2025

|  |  |  |
| --- | --- | --- |
|  | **K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University) Department of Science and Humanities** |  |

**Output(s):**

**Q2:**

**A screen shot of a computer

Description automatically generated**

**Q3:**

**A screenshot of a computer

Description automatically generated**

**Conclusion:**

Through our exploration of NumPy, we have learned that it is an incredibly powerful library for numerical computations in Python, especially when working with arrays. We discovered how efficiently it handles data processing tasks, such as slicing, indexing, and performing mathematical operations on multi-dimensional arrays. This makes it ideal for managing large datasets and preparing them for further analysis. The ability to extract specific rows, columns, or subarrays and apply transformations has been particularly useful for tasks like organizing, filtering, and manipulating data. Overall, learning NumPy has provided us with a solid foundation for handling data efficiently.

**Post Lab Descriptive Questions**

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

print(np.random.randint(1, 100))

A screenshot of a computer

Description automatically generated

1. **Explain the slicing of 2-D Array**

In a 2-D NumPy array, slicing allows you to extract specific rows, columns, or subarrays using the syntax array[row\_start:row\_end, column\_start:column\_end]. The row\_start and row\_end define the range of rows, while column\_start and column\_end define the range of columns; slicing is non-inclusive of the end index. Omitting these indices defaults to including all rows or columns. For example, array[1, :] extracts the second row, array[:, 2] extracts the third column, and array[1:3, 1:4] extracts a subarray from rows 1 to 2 and columns 1 to 3. Negative indices allow slicing from the end, and steps can be used, such as array[::2, ::2], to select every other row or column. To reverse rows or columns, use array[::-1, :] or array[:, ::-1]. This makes slicing a powerful tool for manipulating arrays efficiently.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**

PP Sem I/ 2024-2025