NumPy

Learn in 3 hours

Amit Thinks


Join
Statistical Operations
Arithmetic Operations
Sort
Slice
Reshape

Numpy Tutorial for Beginners

1. NumPy Introduction
2. Installing NumPy
3. Create NumPy arrays
4. Dimensions in NumPy Arrays
5. Initialize NumPy Arrays
6. Datatypes in NumPy
7. NumPy Array Indexing
8. NumPy Array Slicing
9. NumPy Array Shape
10. Reshape a NumPy array
11. Iterate NumPy Arrays
12. Joining NumPy Arrays
13. Split NumPy Array
14. Search an Array for a value

15. Sorting NumPy Arrays
16. Axes in NumPy arrays
17. Intersection of NumPy Arrays
18. Difference between NumPy arrays
19. Arithmetic Operations on NumPy arrays
20. Scalar operations on NumPy arrays
21. Statistical Operations on NumPy arrays
22. Random module in NumPy
23. NumPy Logs
24. NumPy LCM and HCF

WHAT IS NumPy?

NumPy is a Python library for numeric and scientific computing. It stands for numerical python and consists of multi-dimensional array objects. It is faster with less LOC.

NumPy allows you to perform various operations on Arrays, including

- Array Slicing
- Array Join
- Array Split
- Array Reshape
- Array Filter

CREATE NUMPY ARRAYS

To create NumPy arrays, use **numpy.array()** method in Python. We can create arrays using any of the following ways:

- Create a NumPy array by passing Tuple
- Create a NumPy array by passing List

Let's see them one by one:

DIMENSIONS IN NUMPY ARRAYS

Dimensions of an array in NumPy are also called the Rank of an Array. Here, we will see how to check how many dimensions an array has used with the **numpy.ndarray.ndim** attribute .

With that, we will also see some examples to create 0D, 1D, and 2D, arrays:

- Create a Zero-dimensional NumPy array
- Create a One-dimensional NumPy array
- Create a Two-dimensional NumPy array

INITIALIZE NUMPY ARRAYS

To Initialize NumPy arrays, we can use the **numpy.zeros()** method. Using this method, initialize NumPy arrays with zeros.

Let us see two examples to initialize NumPy arrays:

NUMPY - DATATYPES

Python NumPy supports the following data types:

- b – boolean
- u – unsigned integer
- f – float
- c – complex float
- m – timedelta
- M – datetime
- O – object
- S – string
- U – unicode string

GET THE DATATYPE OF A NUMPY ARRAY WITH INTEGERS

To get the datatype of a NumPy array with integers, use the **dtype** property. Let us see an example:

CONVERT ONE DATATYPE TO ANOTHER

Use the **astype()** to create a copy of an array and then set the new data type. This is how you can convert one type to another.

In the below example, we have an array of strings. We will convert it to an integer, with i as a parameter as shown below:

NUMPY - ARRAY INDEXING

Array indexing is accessing array elements. In NumPy, access an array element using the index number. The 0th index is element 1 and the flow goes on as shown below:

- index 0 – element 1
- index 1 – element 2
- index 2 – element 3
- index 3 – element 4

NUMPY - ARRAY INDEXING

In this lesson, we will cover the following topics to understand Array Indexing in NumPy:

- Access elements from a 1D Array
- Access elements from a 2D Array
- Access elements from a 3D Array
- Access elements from the last with Negative Indexing

Let us begin with accessing elements from a One-Dimensional i.e. 1D array:

NUMPY – ACCESS ELEMENTS FROM A TWO-DIMENSIONAL ARRAY

Accessing elements work as a matrix in a 2D Array i.e.

```
a[0,0] = dimension 1 element 1st
a[0,1] = dimension 1 element 2nd
a[0,2] = dimension 1 element 3rd

a[1,0] = dimension 2 element 1st
a[1,1] = dimension 2 element 2nd
a[1,2] = dimension 2 element 3rd

a[2,0] = dimension 3 element 1st
a[2,1] = dimension 3 element 2nd
a[2,2] = dimension 3 element 3rd
```

The following are some examples to access specific elements from a 2D array:

- Example 3: Access the 1st dimension elements from a Two-Dimensional array
- Example 4: Access the 2nd dimension elements from a Two-Dimensional array

NUMPY – ARRAY SLICING

Slicing the array in a range from start to end is what we call Array Slicing. Slice (range) is passed in place of index i.e.

```
n[1:3]
```

The above will slice two elements from index 1 to index 3 because it includes the start index, but excludes the end index.

The following is the syntax to slice NumPy arrays:

```
arr[start:end]
```

Here, **start** is the beginning index (include), and the **end** is the last index (excluded)

NUMPY – SLICE ONE-DIMENSIONAL ARRAYS WITH STEP

We can also include steps while slicing. Step means a jump of elements. If the step is 2, that means 2 jumps for the next element.

The following is the syntax:

```
arr[start:end:step}
```

Let us see an example:

NUMPY – ARRAY SHAPE

In Python NumPy, the number of elements in each dimension of an array is called the shape. To get the shape of a NumPy array, use the **numpy.shape** attribute in Python.

In this lesson, we have covered the following examples:

- Check the shape of a Zero-Dimensional NumPy array
- Check the shape of a One-Dimensional NumPy array
- Check the shape of a Two-Dimensional NumPy array
- Check the shape of a Three-Dimensional NumPy array

NUMPY – ARRAY RESHAPE

In Python NumPy, the number of elements in each dimension of an array is called the shape. To reshape the number of dimensions, use the **numpy.reshape()** method in Python.

Why reshape?

Reshape allows you to edit the number of elements in a dimension. Additionally, with reshape(), you can add or remove dimensions.

Let us see two examples to reshape arrays in NumPy:

- Reshape dimensions from 1D to 2D
- Convert 3D Array to 1D array (Flattening the array)

ITERATE NUMPY ARRAYS

Iteration is displaying each element of an array. Here, we will see how to iterate through NumPy arrays using loops and other methods. We will cover the following examples:

- Iterate a One-Dimensional NumPy array
- Iterate a Two-Dimensional NumPy array
- Iterate a Three-Dimensional NumPy array

JOIN NUMPY ARRAYS

We can join two or more arrays in a single new array using the following:

- **concatenate() method:** Joining along with axis, unlike keys in SQL.
- **stack methods**: Joining along a new axis, unlike keys in SQL.

JOIN NUMPY ARRAYS USING STACK METHODS

Join arrays in NumPy using the stack methods, such as

- stack()
- hstack()
- vstack()
- dstack()
- column_stack()

Let us see the examples one by one:

JOIN NUMPY ARRAYS - HSTACK()

Stack along rows using the **hstack()** method in NumPy. Let us see how to join NumPy arrays using the hstack() method:

Let us see the example:

JOIN NUMPY ARRAYS - VSTACK()

Stack along columns using the **vstack()** method in NumPy. Let us see how to join NumPy arrays using the vstack() method:

Let us see the example:

JOIN NUMPY ARRAYS - DSTACK()

Stack along depth i.e. height using the **dstack()** method in Numpy. Let us see how to join NumPy arrays using the dstack() method:

Let us see the example:

JOIN NUMPY ARRAYS - COLUMN_STACK()

Stack according to columns using the **column_stack()** method in NumPy. Let us see how to join NumPy arrays using the column_stack() method:

Let us see the example:

SPLIT – NUMPY ARRAYS

Split means to break/ slash an array into multiple arrays. To split an array, use the **array_split()** method. The following is the syntax:

```
array_split(arr_name, split_num)
```

Here, **arr_name** is the name of the array, **split_num** is the count of splits.

In this lesson, we will learn how to,

- Split a 1D array
- Access a split 1D array
- Split a 2D array and how to access

SEARCH A NUMPY ARRAY FOR A VALUE

In a NumPy array, we can search for a specific value using the **numpy.where()** method. This method returns the index where the specific element is found.

Let us see an example to search an array for a value in NumPy:

SORTING NUMPY ARRAYS

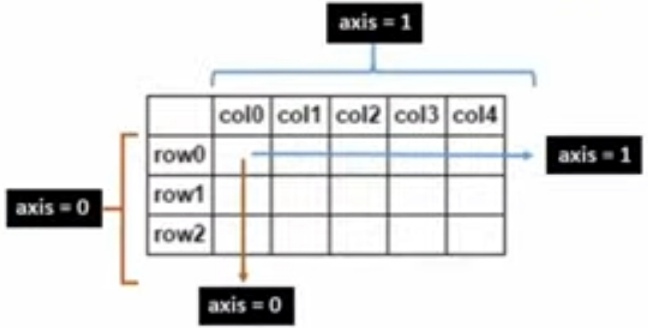
The **numpy.sort()** function is used in NumPy to sort arrays in a sequence. This sequence can be ascending or descending. Here, we will see how to,

- Sort a One-Dimensional Integer Array
- Sort a One-Dimensional String Array
- Sort a Two-Dimensional Array

AXES IN NUMPY ARRAYS

In NumPy arrays, the axis is the direction along the rows and columns. A two-dimensional NumPy array has the following two corresponding axes:

```
vertical axis (axis 0)
horizontal axis (axis 1)
```



NUMPY - GET MINIMUM VALUE WITH AXES

Specifying axes will calculate values along that specific axes only.

Let us see an example to understand the concept of axis and get the minimum value. We have used the **min()** method and the **axis** attribute:

NUMPY ARRAYS - INTERSECTION

If you want to get the intersection of two arrays, use the **intersect1d()** method in NumPy. Intersection means finding common elements between two arrays. In this lesson, we will see some examples:

- Find the intersection between two arrays
- Finding intersection between unsorted arrays sorts the resultant array
- Find the intersection between two arrays with different elements

NUMPY – FINDING INTERSECTION BETWEEN TWO UNSORTED ARRAYS SORTS THE ARRAY

The **intersect1d()** method finds the intersection and also sorts the result. Let us see an example. We have created two unsorted integer arrays. We will find the intersection between these arrays that will display a sorted result:

NUMPY – DIFFERENCE BETWEEN TWO ARRAYS

To find the difference between two arrays, use the **numpy.setdiff1d()** method. This means if we are subtracting two arrays, then **setdiff1d()** will subtract the common elements from the 1st array and display the rest of the elements of the 1st array.

The 1st and 2nd arrays are placed as two-parameter values as shown in the below syntax:

```
setdiff1d(array1, array2)
```

Let us now see two examples and understand the concept.

EXAMPLE 2 – FIND THE DIFFERENCE BETWEEN TWO ARRAYS

Let us see the above example, with parameters being swapped. That means now, we will perform **n2 – n1**.

Let us see an example:

ARITHMETIC OPERATIONS ON NUMPY ARRAYS

Perform arithmetic operations on NumPy arrays using the sum(), subtract(), multiply(), and divide() methods. Let us see the operations one by one:

- Add NumPy Arrays
- Subtract NumPy Arrays
- Multiply NumPy Arrays
- Divide NumPy Arrays

ADD NUMPY ARRAYS

To add the elements of NumPy arrays, the **sum()** method is used. It adds each and every element of both arrays and displays the result in a new array.

Let us see the following examples to add arrays:

- Add all the elements of a NumPy arrays
- Add column values using the axis parameter
- Add individual array values using the axis parameter

SUBTRACT NUMPY ARRAYS

Use the **subtract()** method to subtract one array from another. It subtracts and displays the result in a new array.

Let us see an example to subtract NumPy arrays:

MULTIPLY NUMPY ARRAYS

Use the **multiply()** method to multiply elements of one array to another. It multiplies and displays the result in a new array.

Let us see an example to multiply NumPy arrays:

DIVIDE NUMPY ARRAYS

The **divide()** method divides elements of one array with elements of another. It divides and displays the result in a new array.

Let us see an example to divide NumPy arrays:

NUMPY ARRAYS - SCALAR OPERATIONS

Scalar operations on NumPy arrays include performing addition or subtraction, or multiplication on each element of a NumPy array. Let us see the following examples:

- Addition Operation on NumPy Arrays
- Subtraction Operation on NumPy Arrays
- Multiplication Operation on NumPy Arrays
- Division Operation on NumPy Arrays

NUMPY ARRAYS – ADDITION OPERATION

The Addition Operation is adding (+) a value to each element of a NumPy array. Let us see an example:

NUMPY ARRAYS – SUBTRACTION OPERATION

The Subtraction operation is subtracting (-) a value from each element of a NumPy array. Let us see an example:

NUMPY ARRAYS – MULTIPLICATION OPERATION

The Multiplication operation is multiplying (*) a value to each element of a NumPy array. Let us see an example:

NUMPY ARRAYS – DIVISION OPERATION

The Division operation is to divide (/) a value from each element of a NumPy array. Let us see an example:

NUMPY ARRAYS – STATISTICAL OPERATIONS

We can perform statistical operations, like mean, median, and standard deviation on NumPy arrays. Let's see them one by one:

- Mean
- Median
- Standard Deviation

STATISTICAL OPERATIONS - MEAN

Mean is the average of the given values. Here, we will find the mean of the array elements using the **mean()** method.

Let us see an example:

STATISTICAL OPERATIONS - MEDIAN

Median is the middle value of the given values. Here, we will find the median of the array elements using the **median()** method.

Let us see an example:

STATISTICAL OPERATIONS – STANDARD DEVIATION

Standard Deviation is a measure of the amount of variation or dispersion of the given values. Here, we will find the standard deviation of the array elements using the **std()** method.

Let us see an example:

RANDOM MODULE IN NUMPY

To work with Random numbers, NumPy has a module called **random**. To use the module, import it at the beginning of a Python program as shown below:

```
from numpy import random
```

Let us see some examples:

- Generate a random number
- Generate a random array with a fixed size
- Generate one of the random values based on an array of values

GENERATE A RANDOM NUMBER

To generate a random number, we have used the **randint()** method of the random module.

Let us see an example:

GENERATE A RANDOM NUMBER WITH A FIXED SIZE

We will generate random elements of an array using the **random.randint()** method. The size of the array is set using the **size** parameter.

Let us see an example:

GENERATE ONE OF THE RANDOM VALUES BASED ON AN ARRAY OF VALUES

One of the random values based on an array of values can be generated using the **choice()** method of the **random** module.

Let us see an example:

NUMPY - LOGS

Python NumPy has some predefined functions to work with Logs, such as numpy.log2(), numpy.log10(), etc. Let us see some examples:

- Log Base 2
- Log Base 10

NUMPY – LCM AND HCF

Let us see how to find the LCM and HCF of two numbers in NumPy.

The **numpy.lcm()** is used to find the LCM, whereas **numpy.gcd()** method returns the GCD in Python.