# Data Handling in NumPy

**Data can be defined as a systematic record of a particular quantity. It is the different values of that quantity represented together in a set. Data is a collection of facts and figures to be used for a specific purpose such as a survey or analysis. When data is arranged in an organized form, can be called information.**

**Purpose of Data**

* **Improve Business Performance**
* **Improve Decision Making**
* **To find solution of problems**
* **To get desired results**
* **To perform various calculation**
* **For analysing various activity**

**Structured and Unstructured Data**

**We can easily find structured data in our database system in the form of fields such as names, dates, addresses etc., as the time goes by, people think how to handle unstructured data like text, image, video, audio, etc. that might give you something useful to make decision in your business.**

**Data Processing Cycle**

**Data processing cycle as the term suggests a sequence of steps or operations for processing data, i.e., processing raw data to the usable form. The processing of data can be done by number of data processing methods. Stages of data processing:**

**1. Input – The raw data after collection needs to be fed in the cycle for processing. This is considered the first step and called input.**

**2. Processing – Once the input is provided the raw data is processed by a suitable or selected processing method. This is the most important step as it provides the processed data in the form of output which will be used further.**

**3. Output – This is the outcome and the raw data provided in the first stage is now “processed” and the data is useful and provides information and no longer called data.**

**Basic Statistic Method for Understanding the Data**

**Mean/Average: Mean or Average is a central tendency of the data i.e. a number around which a whole data is spread out. In a way, it is a single number which can estimate the value of whole data set. Let’s calculate mean of the data set having 8 integers.**

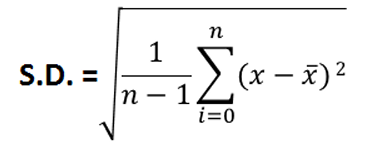


**Medium: Median is the value which divides the data in 2 equal parts i.e. number of terms on right side of it is same as number of terms on left side of it when data is arranged in either ascending or descending order.**

**Median will be a middle term, if number of terms is odd Median will be average of middle 2 terms, if number of terms is even.**

**Mode: Mode is the term appearing maximum time in data set i.e. term that has highest frequency.**

**Standard deviation: Standard deviation is the measurement of average distance between each quantity and mean. That is, how data is spread out from mean. A low standard deviation indicates that the data points tend to be close to the mean of the data set, while a high standard deviation indicates that the data points are spread out over a wider range of values. When we are asked to find SD of some part of a population, a segment of population; then we use sample Standard Deviation.**



**where x is mean of a sample.**

**Variance: Variance is a square of average distance between each quantity and mean. That is it is square of standard deviation.**

Variance = (*S*.

# **NUMPY**

**NUMPY-NumPy stands for “Numeric Python” or “Numerical python”. NumPy is a package that contains several classes, functions, variables etc. to deal with scientific calculations in Python. NumPy is useful to create and process single and multi-dimensional arrays. In addition, NumPy contains a large library of mathematics like linear algebra functions and Fourier transformations. The arrays which are created using NumPy are called n dimensional arrays where n can be any integer. If n = 1 it represents a one-dimensional array. If n= 2, it is a two-dimensional array etc. NumPy array can accept only one type of elements. We cannot store different data types into same arrays.**

NumPy arrays are used to store lists of numerical data, vectors and matrices. The NumPy library has a large set of routines (built-in functions) for creating, manipulating, and transforming NumPy arrays. Python language also has an array data structure, but it is not as versatile, efficient and useful as the NumPy array. The NumPy Contiguous memory allocation: The memory space must be divided into the fined sized position and each position is allocated to a single data only. Now Contiguous Memory Allocation: Divide the data into several blocks and place in different parts of the memory according to the availability of memory space. The NumPy array is officially called ndarray but commonly known as array. In rest of the chapter, we will be referring to NumPy array whenever we use “array”. following are few differences between list and Array.

## **Difference Between List and Array**

|  |  |
| --- | --- |
| List | NumPy Array |
| List can have elements of different data types for example, [1,3.4, ‘hello’, ‘a@’] | All elements of an array are of same data type for example, an array of floats may be: [1.2, 5.4, 2.7] |
| Elements of a list are not stored contiguously in memory | Array elements are stored in contiguous memory locations. This makes operations on arrays faster than lists. |
| Lists do not support element wise operations, for example, addition, multiplication, etc. because elements may not be of same type. | Arrays support element wise operations. For example, if A1 is an array, it is possible to say A1/3 to divide each element of the array by 3. |
| Lists can contain objects of different datatype that Python must store the type information for every element along with its element value. Thus lists take more space in memory and are less efficient. | NumPy array takes up less space in memory as compared to a list because arrays do not require to store datatype of each element separately. |
| List is a part of core Python. | Array (ndarray) is a part of NumPy library. |

## **Creation of NumPy Arrays from List:**

There are several ways to create arrays. To create an array and to use its methods, first we need to import the NumPy library.

#NumPy is loaded as np (we can assign any #name), numpy must be written in lowercase

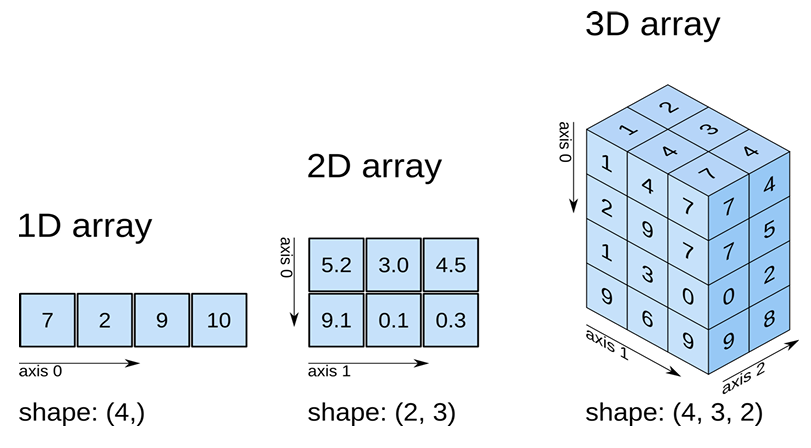
import numpy as np

The NumPy’s array() function converts a given list into an array. For example,

#Create an array called array1 from the #given list.

array1 = np.array([10,20,30])

## **Array Structure in NumPy**



## **Type of Array in NumPy**

An array in NumPy is of the following types-

1. 1D Array

2. 2D Array

3. N-Dimension Array

## **One-Dimension Array**

One dimensional array contains elements only in one dimension, we will explore the concept of one-dimensional arrays in NumPy, a powerful library in Python for numerical computing. One-dimensional arrays, also known as vectors, are fundamental data structures in NumPy that allow for efficient storage and manipulation of numerical data. We will cover how to create, access, and perform operations on one-dimensional arrays, providing examples to illustrate each concept.

## **Creating a One-Dimensional Array**

To create a one-dimensional array in NumPy, you can use the numpy.array() function. Here’s a simple example:

import numpy

a = numpy.array([10,20,30,40,50])

print(a)

Output: [10,20,30,40,50]

import numpy as np

a = np.array([10,20,30,40,50])

print(a)

Output: [10, 20, 30, 40, 50]

Note: if we use the following statement then there is no need to add anything in front of array function.

from numpy import \*

a = array([10, 20,30,40,50])

print(a)

Output : [10, 20,30,40,50]

## **Implementation of One-Dimension Array in NumPy**

Creating array in numpy can be done in several ways. Some of the important ways are

* Using array() function
* Using linspace() function
* Using arange() function
* Using zeros() and ones() functions

**numpy.array() function**

Using this function we can create array of any data type, but if not data types is mentioned the default data type will be the "int" For e.g :

from numpy import \*

Arr=array([10,20,30,40,50],int) is similar to

arr = array([10,20,30,40,50])

While creating array if one of the values in the specified list belongs to float then all the values will be converted to float by default.

from numpy import \*

a = array([10,30,40.5, 50,100])

print(a)

Output : = [10.0,30.0,40.5,50.0,100.0]

**numpy.linspace() Function**

The linspace() function is used to create an array with evenly spaced points between a starting and ending point. The following examples demonstrate the use of linspace() function.

Syntax- linspace(start, stop, N)

* Start represents the starting number
* Stop represents the ending element
* N represents the number of parts the elements should be divided

import numpy as np

a = np.linspace(1,10,10)

print(a)

Output :[ 1. 2. 3. 4. 5. 6. 7. 8. 9. 10.]

**numpy.arange() Function**

The arange() function in NumPy is same as range() function in Python. The following format is used to create an array using the arange() function**.**

Syntax- arange(start,stop,stepsize)

arange(10) – will create an array with values [0,1,2,3,4,5,6,7,8,9]

arange(5,10) – will create an array with values [5, 6,7,8,9]

arange(10,1,-1) will create an array with values[10,9,8,7,6,5,4,3,2]

import numpy as np

a = np.arange(10)

b = np.arange(5,10)

c = np.arange(10,1,-1)

print(a)

print(b)

print(c)

Output-

[0,1,2,3,4,5,6,7,8,9]

[5,6,7,8,9]

[10, 9, 8, 7, 6, 5, 4, 3, 2]

**Creating array using ones() and zeros() functions**

We can use zeros() function to create an array with all zeros. The ones() function will is useful to create an array with all 1s. They are written in the following format-

Syntax:

zeros(n,datatype)

ones(n,datatype)

Note : if datatype is missing then the default value will be float.

zeros(5) will create an array with five zero values.

ones(5) will create an array with five 1 values.

Example:

import numpy as np

K = np.zeros(5)

R = np.ones(5)

print(K)

print(R)

Output :

[0.,0.,0.,0.,0.]

[1.,1.,1.,1.,1.]

**Mathematical Operations on Arrays**

It is possible to perform various Mathematical operations like addition, subtraction, division etc. on the elements of any arrays. The functions of math module can be applied to the elements of any array.

Before performing mathematical operations, you need to create a NumPy array. You can create an array using the numpy.array() function or by using built-in functions like numpy.arange() or numpy.zeros().

Import numpy as np

array1 = np.array([1, 2, 3, 4, 5])

array2 = np.arange(10)

array3 = np.zeros((2, 3)) # 2x3 array

**Basic Mathematical Operations**

NumPy supports a variety of mathematical operations that can be performed element-wise on arrays.

**Addition:** You can add two arrays together using the + operator.

result\_add = array1 + array1 # Element-wise addition

**Subtraction:** Subtraction can be performed similarly using the - operator.

result\_sub = array1 - array1 # Element-wise subtraction

**Multiplication:** For multiplication, use the \* operator.

result\_mul = array1 \* 2 # Multiply each element by 2

**Division:** Division is performed using the / operator.

result\_div = array1 / 2 # Divide each element by 2

**Exponentiation:** You can raise each element to a power using the \*\* operator.

result\_exp = array1 \*\* 2 # Square each element

**Aggregate Functions:** NumPy provides several functions to perform aggregate operations on arrays.

**Sum:** You can calculate the sum of all elements in an array using numpy.sum().

total\_sum = np.sum(array1)

**Mean:** To find the mean of the elements, use numpy.mean().

mean\_value = np.mean(array1)

**Maximum and Minimum:** To find the maximum and minimum values, use numpy.max() and numpy.min().

max\_value = np.max(array1)

min\_value = np.min(array1)

**Universal Functions (ufuncs):** NumPy also provides universal functions (ufuncs) that operate element-wise on arrays. These include trigonometric functions, logarithmic functions, and more.

**Example:**

**Sine Function**

angles = np.array([0, np.pi/2, np.pi])

sine\_values = np.sin(angles) # Calculate sine of each angle

**Example: Logarithm**

log\_values = np.log(array1) # Natural logarithm of each element.

## **N-Dimension Array in NumPy**

NumPy is a powerful library in Python that provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. This document explores the concept of N-dimensional arrays in Numpy, detailing how to create, manipulate, and utilize them effectively for various computational tasks.

**Introduction to N-Dimensional Arrays**

An N-dimensional array is a generalization of 1D (vector), 2D (matrix), and higher-dimensional arrays. In NumPy, these arrays are represented by the ndarray object, which allows for efficient storage and manipulation of data. The dimensionality of an array is defined by the number of axes it has, where each axis corresponds to a dimension.

## **Creating N-Dimensional Arrays**

Numpy provides several methods to create N-dimensional arrays:

**Using `numpy.array():** You can create an N-dimensional array from a nested list or tuple:

import numpy as np

array\_3d = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])

print(array\_3d)

**Using `numpy.zeros()`, `numpy.ones()`, and `numpy.empty()`:**

These functions allow you to create arrays filled with zeros, ones, or uninitialized values, respectively:

zeros\_array = np.zeros((2, 2, 2))

print(zeros\_array)

ones\_array = np.ones((2, 2, 2))

print(ones\_array)

empty\_array = np.empty((2, 2, 2))

print(empty\_array)

**Using `numpy.arange()` and `numpy.reshape()`:**

You can create an array with a range of values and then reshape it into an N-dimensional array:

array\_1d = np.arange(8)

array\_3d\_reshaped = array\_1d.reshape((2, 2, 2))

print(array\_3d\_reshaped)