Numpy

# Table of Contents

|  |  |
| --- | --- |
| **Chapter - 1 : Introduction**   1. Why use Numpy? 2. Numpy Features 3. Advantages of Numpy over Normal Array |  |
| **Chapter - 2 : Ndarray and Its Attributes**   1. Ndarray 2. Attributes 3. Creating numpy ndarray object 4. Dimensions in Arrays 5. Numpy Array Indexing and Slicing |  |
| **Chapter - 3 : Basic Functions**   1. np.where() 2. Functions of Rounding 3. Basic Statistical Functions |  |
|  |  |

# 

# 

# Introduction

Here is the [**link**](https://drive.google.com/file/d/1ifNS2e9AbHwiOZOBeJZJ-pQt9JKkdeYq/view?usp=sharing) to the notebook. Make a copy of it and explore!

Numpy is a library for the python programming language adding support to large, multi-dimensional arrays and matrices along with a large collection of high-level mathematical functions to operate on these arrays.

Numpy can deal with N-dimensional arrays.

To use Numpy in Python, we can import the numpy package as follows:



## Why use Numpy?

In Python we have lists that serve the purpose of arrays, but they are slow to process.

Numpy aims to provide an array object that is up to 50x faster than traditional Python lists.

The array object in Numpy is called **ndarray**, it provides a lot of supporting functions that make working with ndarray very easy.

Arrays are very frequently used in data science, where speed and resources are very important.

## Numpy Features

Numpy is useful when it comes to array manipulation. Following table lists some features of numpy used for array creation and manipulation.

#### 

|  |  |
| --- | --- |
| **Feature** | **Description** |
| **Numpy 1-D Array** | Making 1D array |
| **Numpy 2-D Array** | Making 2D array |
| **Array Multiplication** | Multiplying 2 or more array |
| **numpy.ones** | Matrix filled with ones |
| **numpy.zeros** | Matrix filled with zeros |
| **numpy.random** | Matrix filled with random numbers |
| **numpy.arange** | Create array with increments of a fixed step size |
| **numpy.linspace** | Create array of fixed length |
| **numpy.full** | Create a constant array of any number ‘n’ |
| **numpy.tile** | Create a new array by repeating an existing array for a particular number of times |
| **numpy.eye** | Create an identity matrix of any dimension |
| **numpy.random.randint** | Random integer |
| **Numpy 3-D Array** | Making 3-D array |

# Advantages of Numpy over Normal Array

* Numpy uses much less memory to store data
* It allows creation of N-dimensional arrays
* Mathematical operations on Numpy n-dimensional arrays
* More powerful slicing and Broadcasting functionality
* Efficient Data Representation

# Numpy provides a help function, providing the documentation for its methods, functions, classes and modules, by using the .info() function.

# 

# Ndarray and Its Attributes

## Ndarray:

An ndarray is a (usually fixed-size) multidimensional container of items of the same type and size.

**Syntax:**

**numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)**

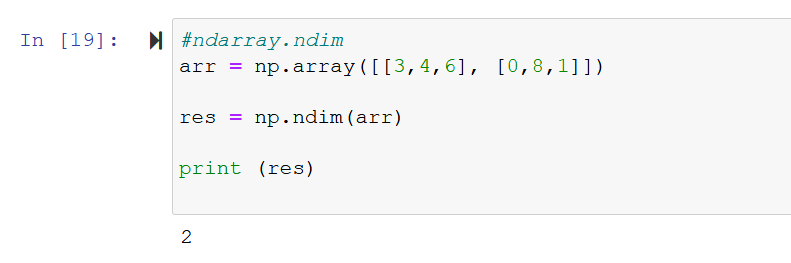
#### 

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **object** | It represents the collection object. It can be a list, tuple, dictionary, set, etc. |
| **dtype** | We can change the data type of the array elements by changing this option to the specified type. The default is none. |
| **copy** | It is optional. By default, it is true which means the object is copied. |
| **order** | There can be 3 possible values assigned to this option. It can be C (column order), R (row order), or A (any) |
| **subok** | The returned array will be a base class array by default. We can change this to make the subclasses pass through by setting this option to true. |
| **ndmin** | It represents the minimum dimensions of the resultant array. |

## Attributes:

**ndarray.ndim**

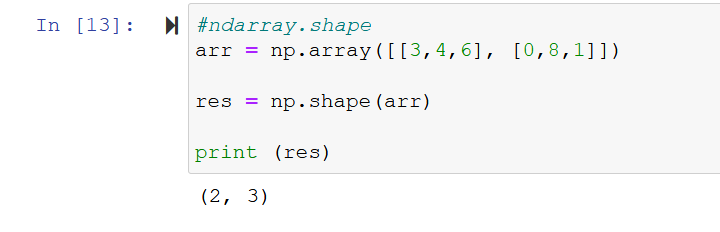
ndim represents the number of dimensions (axes) of the ndarray.



E.g. for this 2-dimensional array [ [3,4,6], [0,8,1]], the value of ndim will be 2. This ndarray has two dimensions (axes) - rows (axis=0) and columns (axis=1)

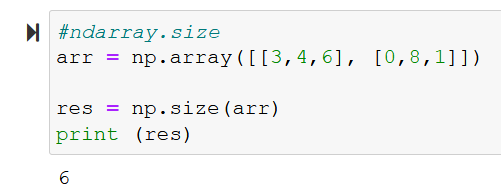
**ndarray.shape**

Shape is a tuple of integers representing the size of the ndarray in each dimension.



E.g. for this 2-dimensional array [[3,4,6],[0,8,1]], value of shape will be (2,3) because this ndarray has two dimensions - rows and columns - and the number of rows is 2 and the number of columns is 3.

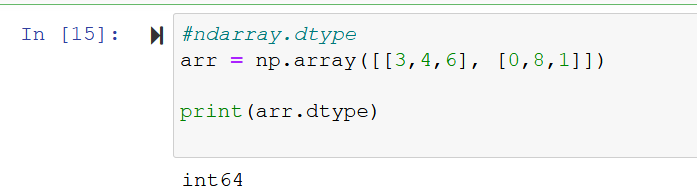
**ndarray.size**

****

Size is the total number of elements in the ndarray. It is equal to the product of elements of the shape. e.g. for this 2-dimensional array [[3,4,6],[0,8,1]], shape is (2,3), size will be product (multiplication) of 2 and 3 i.e. (2\*3) = 6. Hence, the size is 6.

**ndarray.dtype**

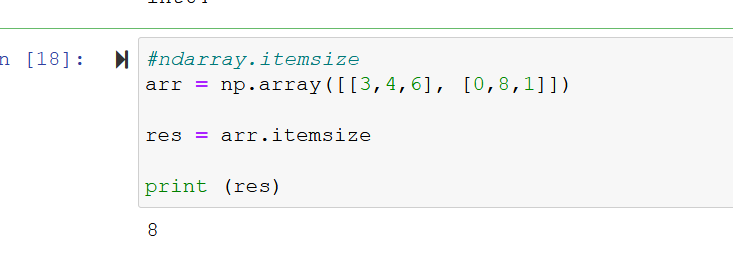
Dtype tells the data type of the elements of a Numpy array. In the Numpy array, all the elements have the same data type.



E.g. for this Numpy array [[3,4,6],[0,8,1]], dtype will be int64

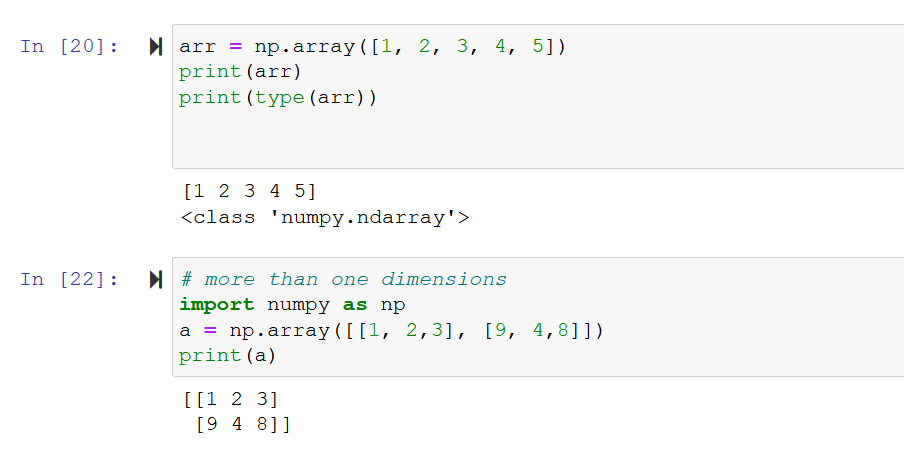
**ndarray.itemsize**

Itemsize returns the size (in bytes) of each element of a Numpy array.



E.g. for this Numpy array [[3,4,6],[0,8,1]], itemsize will be 8, because this array consists of integers and size of integer (in bytes) is 8 bytes.

## Creating numpy ndarray object



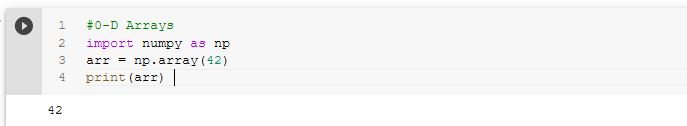
**Dimensions in Arrays**

### 

### 0-D Arrays

0-D arrays, or scalars are the elements in an array

For example:

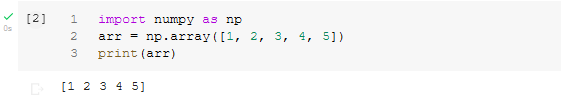


### 

### 1-D Arrays

An array that has 0-D arrays as its elements is called a uni-dimensional or 1-D array.

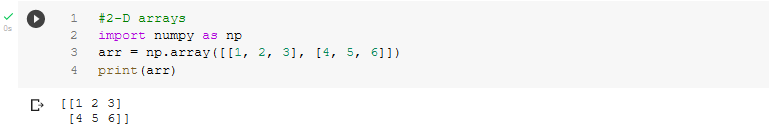
For example:



### 2-D Arrays

### An array that has 1-D arrays as its elements is called a 2-D array.

For example:



**Numpy Array Indexing and Slicing**

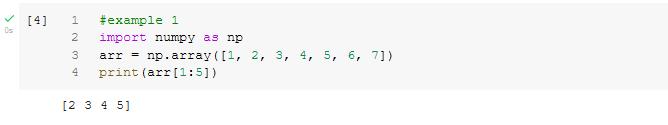
**Slicing**

Slicing in Python means taking elements from one given index to another given index. We pass slice instead of index like this - **[start:end]**

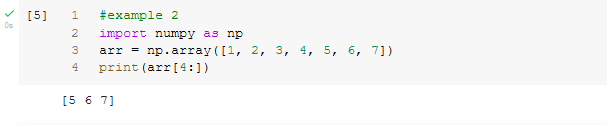
We can also define the step, like this: **[start:end:step]**

* If we don’t mention start, it is set at 0 by default
* If we don’t pass end , it is set at the length of array in that dimension
* If we don’t mention pass, it is set at 1 by default.

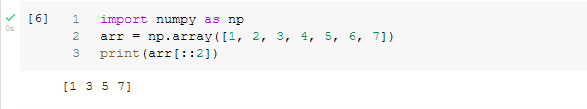
Example 1:



Example 2:



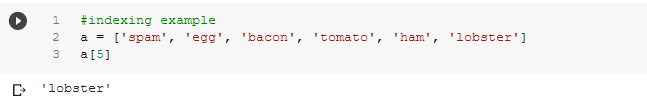
Example 3:



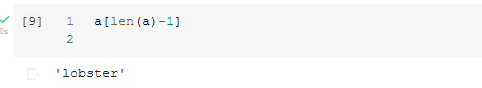
**Indexing**

“Indexing” means referring to an element of an iterable(for example, an array) by its position within the iterable.

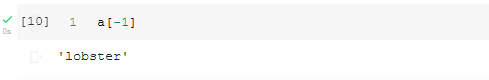
For example:



Or



Or



# Basic Functions

**numpy.where()**

This function either replaces or processes the elements of the numpy array that satisfy the condition.

**Syntax -**

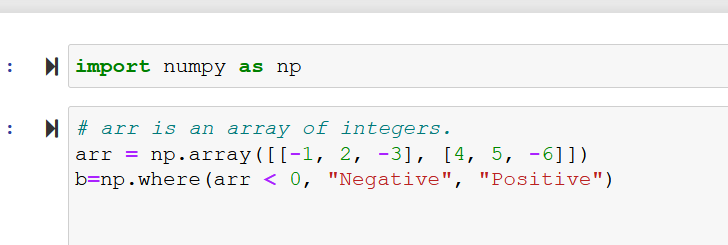
**numpy.where(condition[,x,y])**

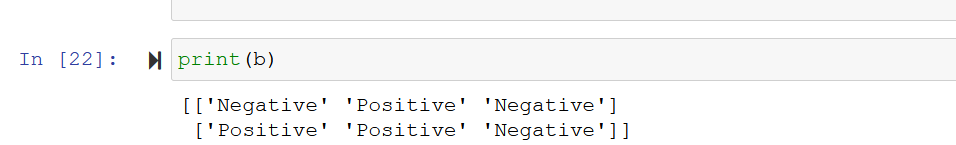
|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **condition** | When True, yield x, otherwise yield y. |
| **x,y** | Return elements, either from x or y, depending on condition. |

Note: If x,y are omitted, it simply returns the indices of the elements satisfying the condition

np.where() is a function that returns ndarray which is **x if condition is True and y if False**.

Example -

****

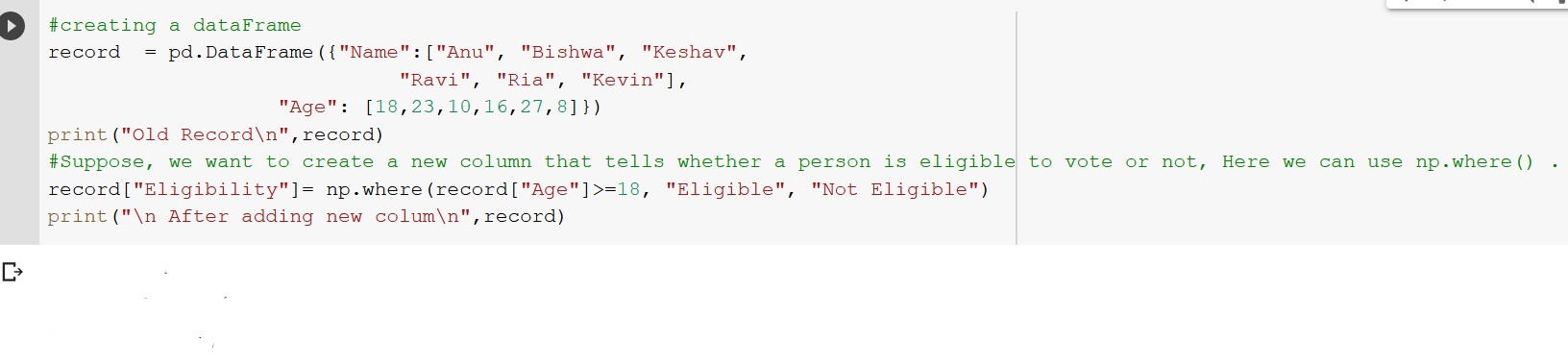
****

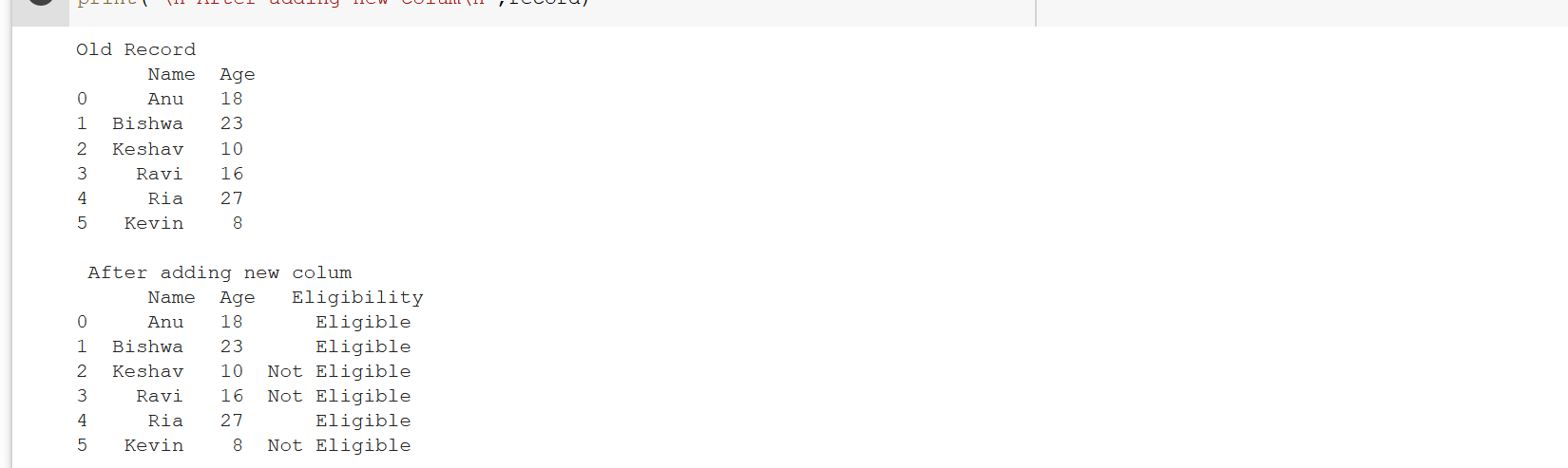
In the above example a new Numpy array is created based on the condition given. If the elements of the array are less than zero it is replaced with “Negative” otherwise with “Positive”.

## Using np.where() in a dataframe

## Where function is not exclusive to Numpy arrays. It can also be used in Pandas DataFrame to extract rows or to create new columns based on the given conditions.

Example -





A new column “Eligibility” is created using np.where() , its values are based on the condition. If a person’s age is greater than 18, the value assigned is “Eligible” otherwise “Not Eligible”.

## Important Points about np.where()

* While passing all 3 arguments to numpy.where(). Then all the 3 numpy arrays must be of the same length otherwise it will raise the following error,
  + ValueError: operands could not be broadcast together with shapes

**Functions of Rounding**

**numpy.around()**

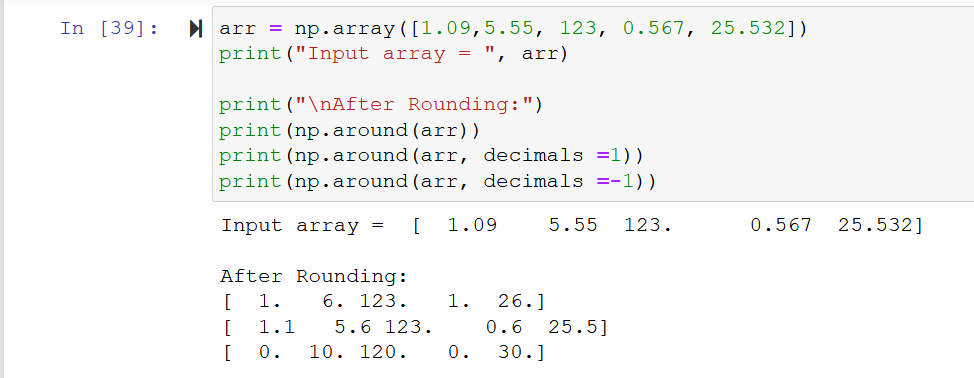
Numpy around() is a mathematical function that helps the user to evenly round array elements to the given number of decimals. The around() method takes up to three parameters and returns an array with all the elements being rounded off to the specified value.

**Syntax** -

**numpy.around(array, decimal (int), out (output array))**

|  |  |
| --- | --- |
| **Parameter** | **Description** |
| **array** | Input array |
| **decimal** | The number of decimals to round to. Default is 0. If negative, the integer is rounded to position to the left of the decimal point |
| **out** | [optional] Output resulted array |

Example -

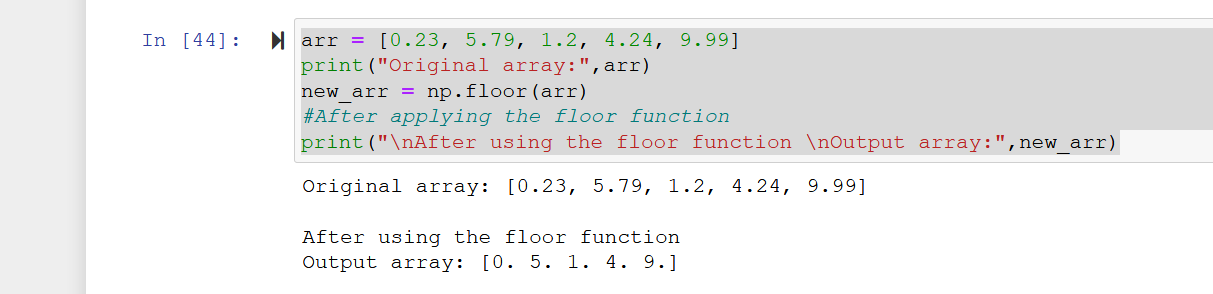


**numpy.floor()**

This function returns the floor value of the input array elements. The floor of a number is the largest integer less than or equal to a given number.

Note: In Python, flooring always is rounded away from 0.

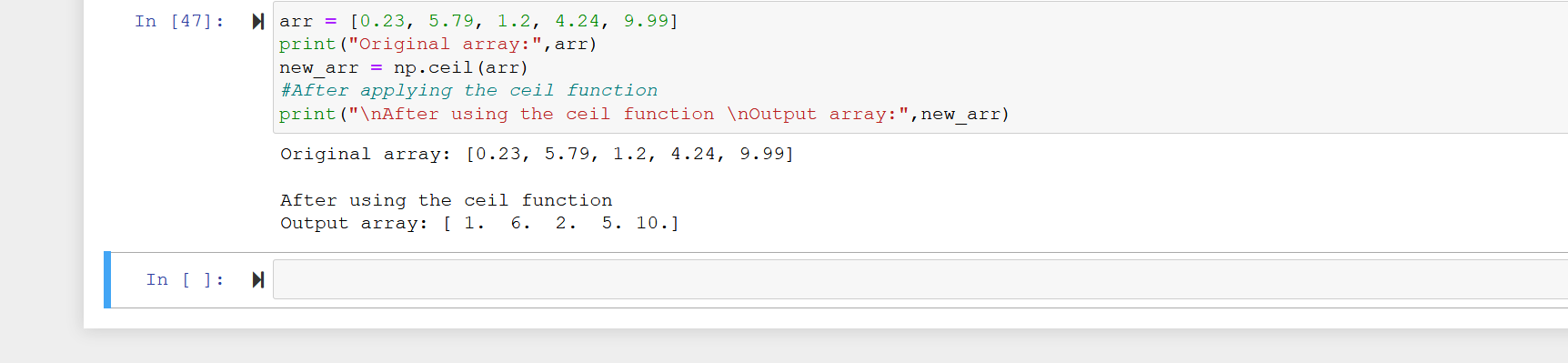
Example -

****

**numpy.ceil()**

The np. ceil() function in Python is used to find the ceil of the elements of the array.the ceil element of a number is the smallest integer that is greater than or equal to that number.

Example -

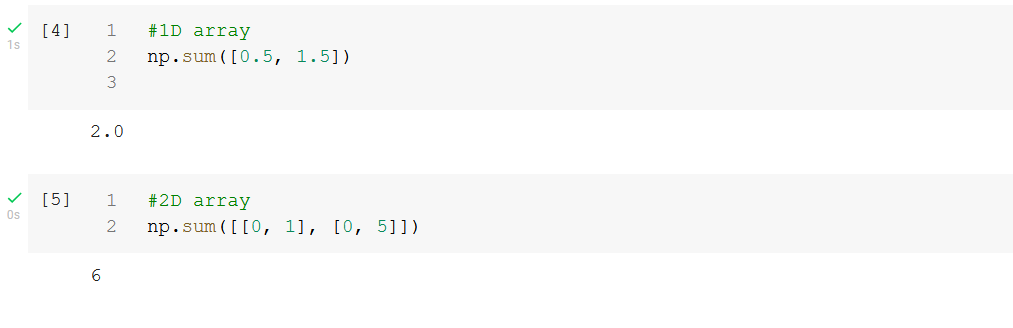


# Basic Statistical Function

**numpy.sum()**

The np.sum() function is used to find the sum of elements over a given axis. The default, axis=None, will sum all of the elements of the input array.If axis is negative it counts from the last to the first axis.If axis is a tuple of ints, a sum is performed on all of the axes specified in the tuple instead of a single axis or all the axes as before.

Example -

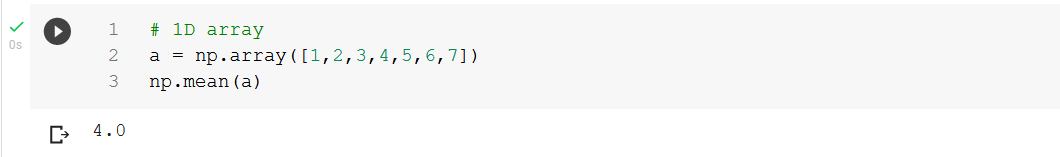


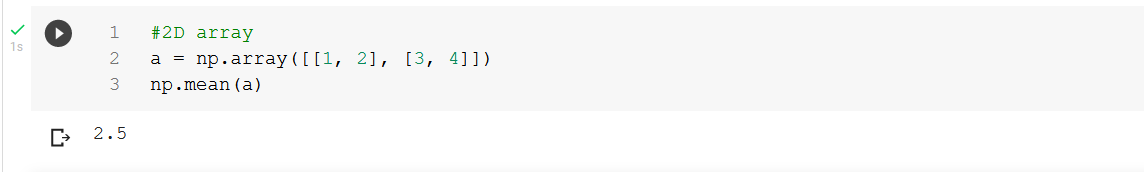
**numpy.mean()**

The np.mean() function is used to compute the arithmetic mean along the specified axis.

Returns the average of the array elements. The average is taken over the flattened array by default, otherwise over the specified axis.float64 intermediate and return values are used for integer inputs.

Example -





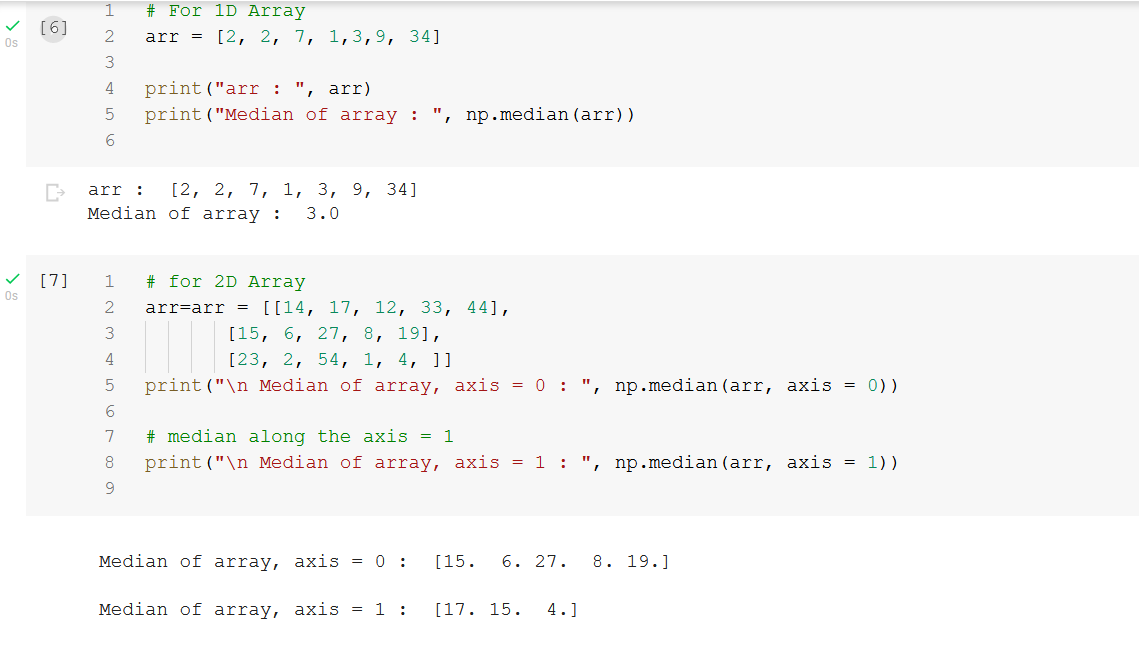
**numpy.median()**

numpy.median(arr, axis = None) : Compute the median of the given data (array elements) along the specified axis.

The default, axis=None, will return the median of the given data.

If axis=0, it will return the median along the column, and if axis =1, it will median along the rows.

Example -



**numpy.std()**

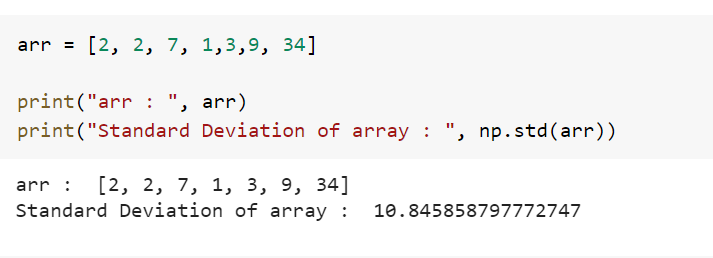
numpy.median(arr, axis = None) : Compute the standard deviation of the given data (array elements) along the specified axis.

The default, axis=None, will return the standard deviation of the given data.

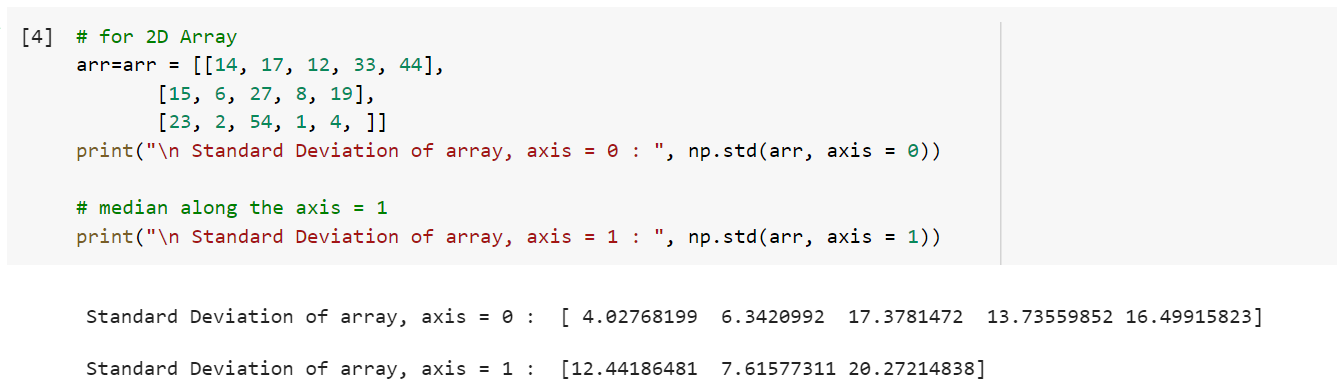
If axis=0, it will return the standard deviation along the column, and if axis =1, it will standard deviation along the rows.

Example -

1. For 1D array

****

1. For 2D Array

****