**NUMPY FUNDAMENTALS**

**What is NumPy?**

**NumPy is the fundamental package for scientific computing in Python.**

**It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and it provide fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.**

**At the core of the NumPy package, is the ndarray object. This encapsulates n-dimensional arrays of homogeneous data types.**

**NumPy Arrays Vs Python Sequences:**

**NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically).**

**Changing the size of a ndarray will create a new array and delete the original.**

**The elements in a NumPy array are all required to be of the same data type (homogeneous), and thus will be the same size in memory.**

**NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data.**



**Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.**

**A growing plethora of scientific and mathematical Python-based packages / libraries like TensorFlow, Pandas, Scikit-learn etc. are using NumPy arrays. though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays.**

**Creating NumPy Arrays:**

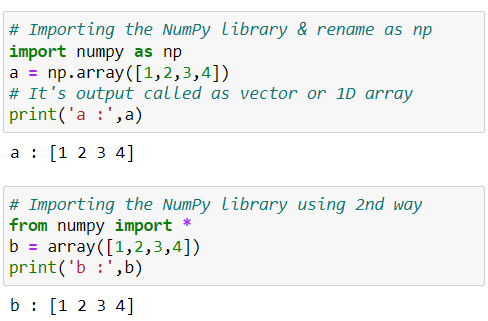
**NumPy is used to work with arrays. The array object in NumPy is called ndarray.**

**We can create a NumPy ndarray object by using the array() function.**

**There are two ways to importing the NumPy module:**

**~ import numpy as np**

**~ from numpy import \***

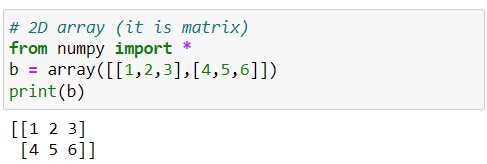
****

**Creating 2D NumPy Arrays:**

**2D array are represented as collection of rows and columns.**

**In machine learning and data science NumPy 2D array known as a matrix.**

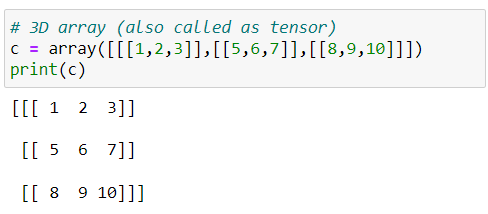
**Specially use to store and perform an operation on input values.**

****

**Creating 3D NumPy Arrays:**

**In machine learning and data science NumPy 3D array known as a tensor.**

**Specially used to store and perform an operation on three-dimensional data like colour image.**

****

**Note: There are so many ways to create numpy arrays depending on situations for that we use other function that are provided by numpy library.**

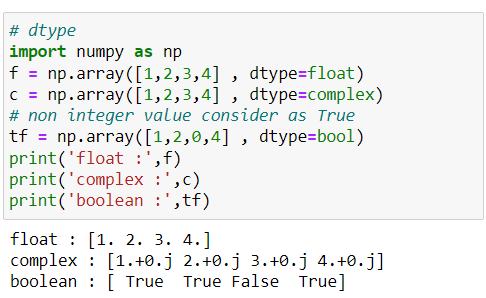
**Creating numpy with different datatype using dtype:**

**It refers to the data type of elements stored in a NumPy array**.

**Allows you to create arrays with different data types, such as integers, floating-point numbers, and more.**

**When creating NumPy arrays, you can indeed specify the data type of the elements using the dtype parameter.**

**Syntax: arr = np.array( [1,2,3] , dtype=float )**

****

### np.arange() function:

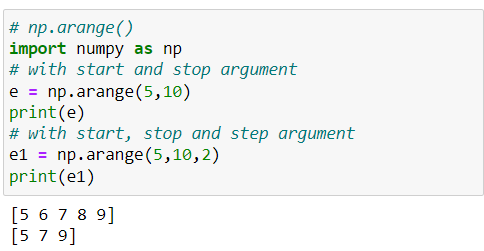
**Used to create arrays containing regularly spaced values within a specified range.**

**Takes three arguments: start, stop, and step.**

**It generates values starting from start, up to (but not including) stop, with increments of step.**

**If step is not provided, it defaults to 1.**

**Syntax: np.arange( start , stop, step, dtype=None )**

****

**np.reshape() function:**

**Used to change the shape (dimensions) of an array without changing its data.**

**Returns a new array with the same data but with a different shape.**

**Useful when we want to convert a 1D array into a two-dimensional array or vice versa.**

**It can also be used to create arrays with a specific shape, such as matrices and tensors.**

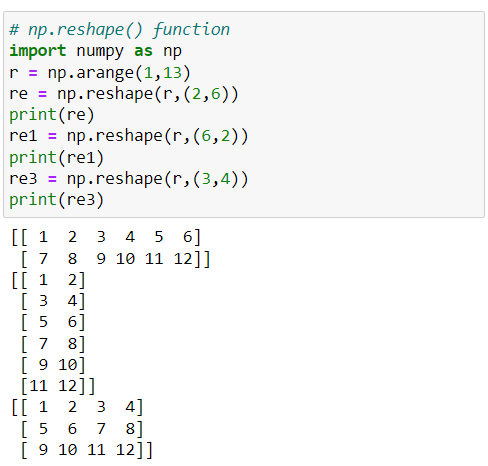
**Syntax: np. reshape (a, new\_shape, order='C')**

**a:** **input array.**

**new\_shape:** **shape of new array**

**order:** **{'C', 'F', 'A'}, optional**

**Note:** **New array dimension number product is equal to number of items that are present in inside the original array.**

****

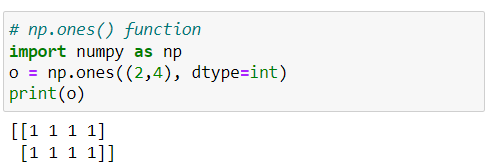
**np.ones() function:**

**Returns a new array of given shape and dtype, where the element’s value is set to 1.**

**Default dtype is float .**

**It is useful in deep learning to initialize the weights values**

**Syntax: np.ones( shape, dtype=None, order='C' )**

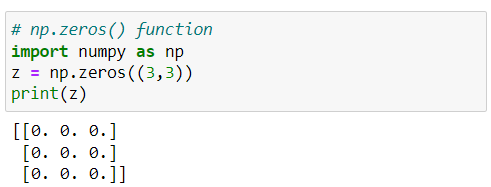


#### **np.zeros() function:**

**Returns a new array of given shape and type, where the element’s value as 0.**

**Default dtype is float .**

**Syntax: np.zeros( shape, dtype=float, order='C' )**

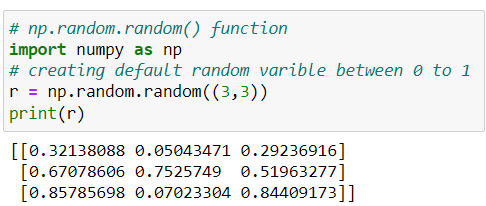
****

**np.random.random() function:**

**Used for generating random numbers.**

**Here 1st random is class name and other one is method name follows OOP concept.**

**Syntax: np. random. random( shape, dtype=None )**

****

#### **np.linspace() function : (Linear/ linearly space)**

**Returns evenly spaced numbers over a specified interval.**

**Use for plotting the ML algorithm result.**

**Syntax: np.linspace( start, stop, num=50, dtype=float,**

**axis=0 )**

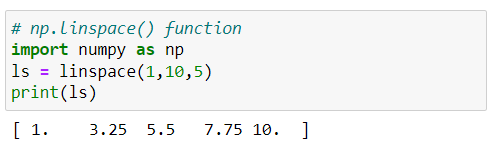
**start:** **starting value of the sequence**

**stop:** **end value of the sequence**

**num:** **number for spacing & default is 50**

**axis:** **axis for evenly spaced numbers &**

**default is 0.**

****

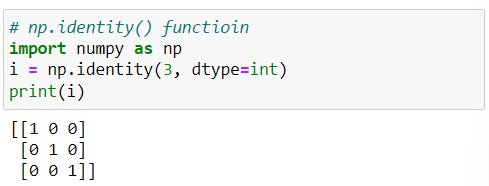
#### **np.identity() function:**

**Returns a square identity matrix of size n x n means diagonally items are 1 and remain all numbers becomes 0's**

**Syntax: np.identity( n, dtype=float )**

**n: size of the identity matrix**

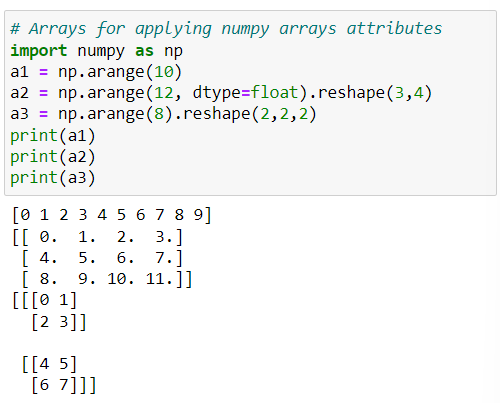
**dtype: we can use another datatype**

****

#### **Attributes of NumPy Arrays:**

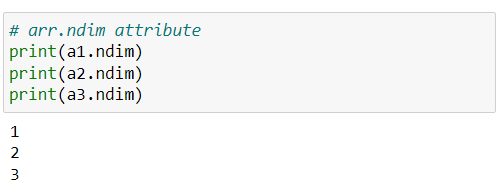
**NumPy array is the most used construct of numpy in machine learning and deep learning.**

**Let us look into some important attributes of this numpy array.**

****

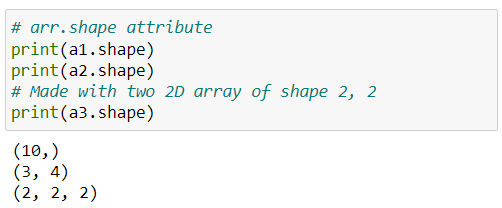
**arr.ndim attribute:**

**Returns the number of dimensions of a given numpy array.**

****

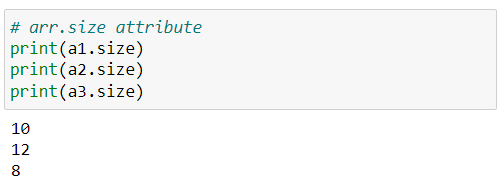
**arr.shape attribute:**

**Determine the dimensions of the array and returns a tuple of integers that represent the size of the array in each dimension.**

****

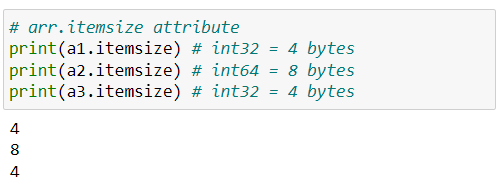
#### **arr.size attribute:**

**Returns the total number of elements in the array.**

****

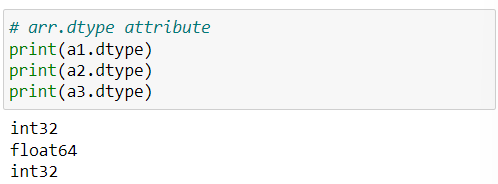
#### **arr.itemsize attribute:**

**Returns the size (in bytes) of each element in the array.**

****

#### **arr.dtype attribute:**

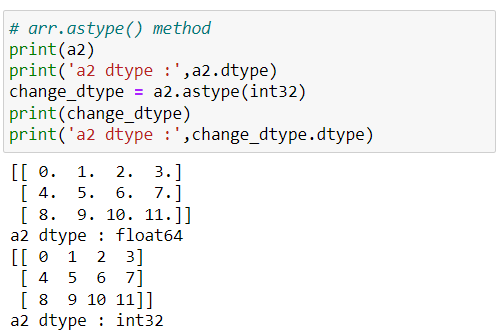
**Returns the datatype of the elements in the array.**

****

#### **Changing datatype using .astype() method:**

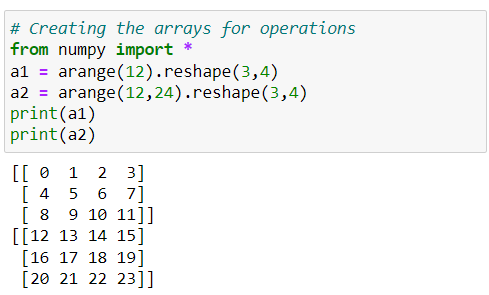
**Change the data type of the elements in the array.**

**More useful in converting the float datatype reduction in integer value.**

****

#### **NumPy Array Operations:**

**Use for performing mathematical operations**

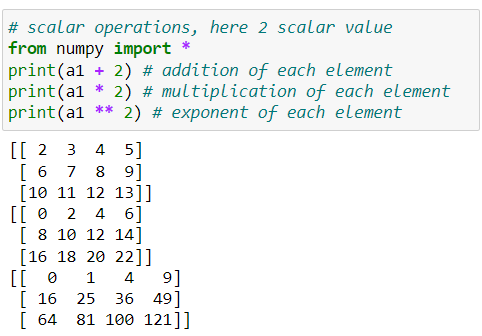
****

**Scalar operations:**

**Scalar operation is an** **operation between a scalar value (a single number) and an array.**

**It can be performed using arithmetic operators such as +, -, \*, and /.**

**Scalar value perform operation with each individual element in the array.**

****

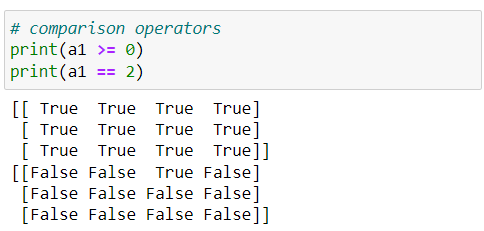
#### **Comparison / relational operations:**

**NumPy provides comparison operators such as ==, <, >, <=, >= etc. for comparing elements in two arrays.**

**This operations return a boolean array with the same shape as the input arrays.**

**~ True indicates condition is satisfied**

**~ False indicates** condition **not satisfied**

****

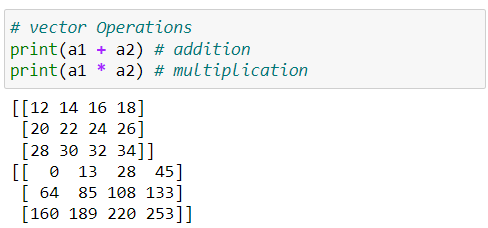
#### **Vector operations:**

**A vector operation is an operation between two arrays of the same size.**

**Vector operations can also be performed using arithmetic operators.**

**When two arrays are added, the corresponding elements in each array are added together & also similarly for another operations.**

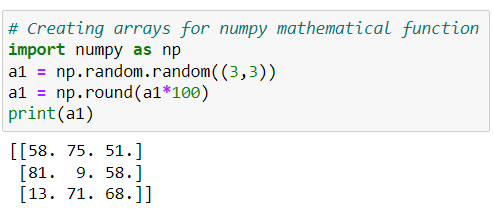
**Note: It is important to note that vector operations can only be performed on arrays of the same shape. If the arrays have different shapes, NumPy will raise a ValueError.**

****

#### **NumPy Array Functions:**

**Some common NumPy array function that is use in machine learning and deep learning etc.**

#### **Numpy array mathematical functions:**

**Use for performing mathematical function**

#### **min():**

**Return the minimum of element in array.**

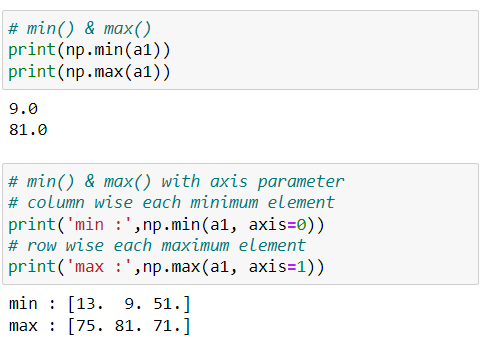
**max():**

**Return the maximum of element in array.**

**But we can also use NumPy min and max to compute the minima and maxima of each column and rows.**

**~ column-wise represents axis=0**

**~ row-wise represents axis=1**



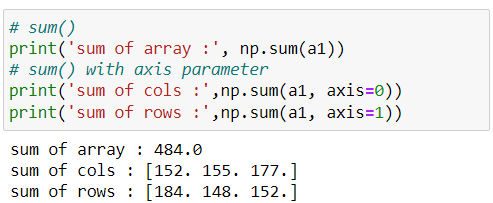
#### **sum():**

**Used to calculate the sum of elements in a NumPy array.**

**Also used to find the sum of all elements in the array or along a specific axis of a multi-dimensional array.**

**~ column-wise represents axis=0**

**~ row-wise represents axis=1**

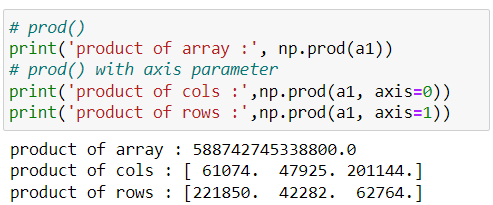
****

#### **prod():**

**Return the product of all element in an array.**

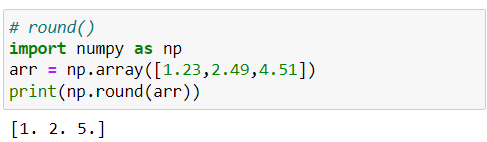
**Along with the axis parameter to calculate the product along a specific axis of a multi-dimensional array.**

**It is a common operation in various mathematical and statistical calculations.**

****

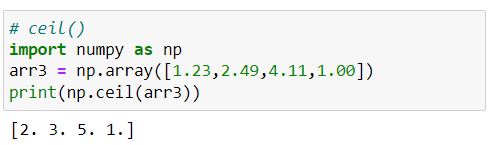
#### **round():**

**Used to rounds the elements of an array to the nearest integer or to a specified number of decimals.**

****

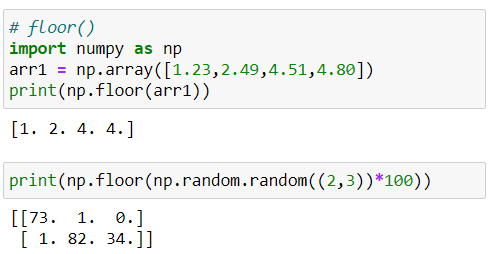
#### **ceil():**

**Used to rounds the elements of an array up to the nearest integer.**



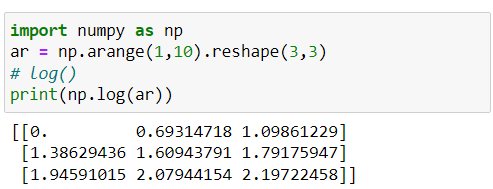
#### **floor():**

**Used to rounds the elements of an array down to the nearest integer.**



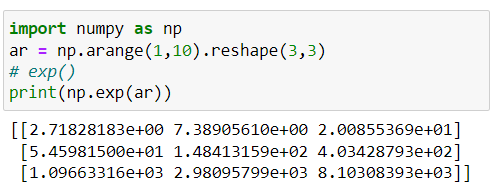
**log():**

**To calculate the natural logarithm of an array or a scalar.**



#### **exp():**

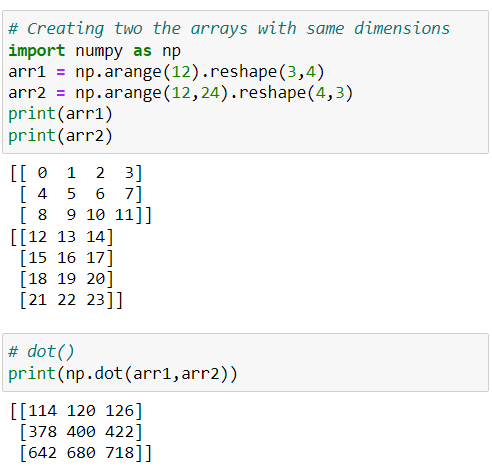
**To calculate the exponential of an array or a scalar.**



**dot():**

**Function takes two array arguments and returns their dot product.**

**The dot product of two vectors and specifying the condition that they must have the same dimensionality.**

****

#### **Numpy array statistical functions:**

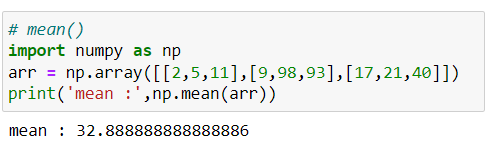
**Here, only a few functions related to statistics have been introduced. We will cover the remaining functions in the statistics session.**

#### **mean():**

**Used to calculate the arithmetic mean or average of the elements in each array.**

**The mean is the sum of all the values in the array divided by the total number of values.**

**It is a common measure of central tendency.**

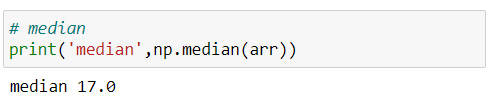
****

**median():**

**Used to calculate the median of the elements in an array.**

**The median is the middle value of a dataset when it is ordered.**

**It is a measure of central tendency that is less affected by outliers than the mean.**

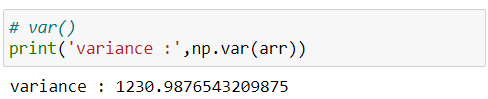


**var():**

**Used to calculate the variance of the elements in an array.**

**Variance is a measure of how much the values in a dataset vary from the mean.**

**It gives you an idea of the spread or dispersion of the data points.**

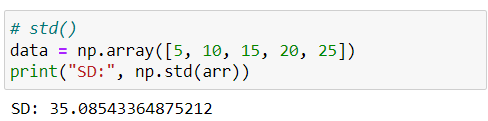
****

**std():**

**Used to calculate the standard deviation of the elements in an array.**

**The standard deviation is a measure of how much the values in a dataset deviate from the mean.**

**It is another measure of the spread or dispersion of the data points, like variance.**

****

**Indexing in NumPy:**

**In NumPy, each element in an array is associated with a number. The number is known as an array index**.

**NumPy array indexing refers to the process of accessing elements or subarrays within a NumPy array.**

**In short, fetching the element from an array.**

**Note: Array start form 0 index.**

**1D Indexing in numpy array:**

**NumPy array indexing is used to access values in the 1D & multi-dimensional arrays.**

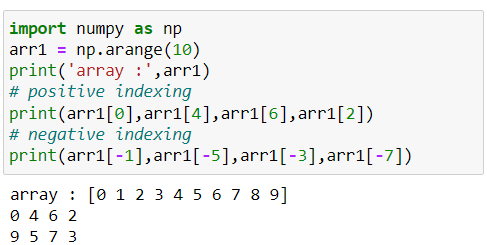
**Indexing is an operation, use this feature to get a selected set of values from a NumPy array.**

**It just like normal indexing like list and, we can you positive or negative indexing.**

**~ positive indexing : array start from 0 index position**

**~ negative indexing :  array start from end -1 index position**

**Syntax: array[ index\_position ]**

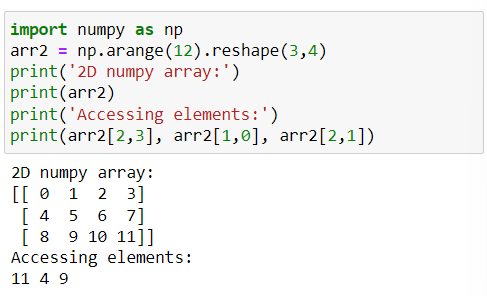
****

**2D Indexing in numpy array:**

**2D numpy arrays are like a table with rows and columns.**

**For accessing elements, we need to specify the row index and column index of the element.**

**Syntax: array[ row\_index , column\_index\_that\_row ]**

****

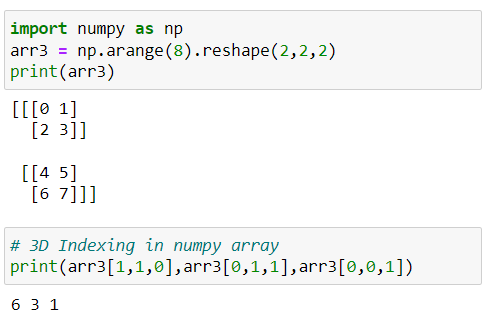
**Note: Array rows & cols start from 0 index.**

**3D Indexing in numpy array:**

**3D numpy arrays are like a table with rows and columns.**

**For accessing elements, we need to specify the row index and column index of the element.**

**Syntax: array[ arr\_index, row\_index , column\_index\_of\_row ]**

****

**Note: Array rows & cols start from 0 index.**

**Slicing in NumPy:**

**NumPy array slicing is used to extract some portion of data from the actual array.**

**NumPy slicing is slightly different.**

**Slicing can be done with the help of (:).**

**Syntax: array[ start : stop : step ]**

**start:  index by default considers as ‘0’**

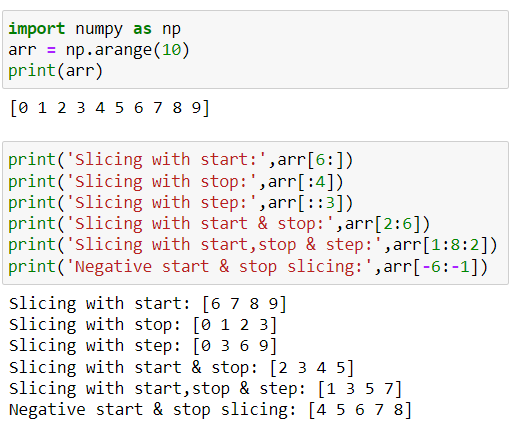
**stop: index considers as a length of the array.**

**step: default is ‘1’.**

**1D Slicing in numpy array:**

**For 1D numpy arrays we use basic slicing, step slicing,**

**and omitting the indices.**

****

**2D Slicing in numpy array:**

**A 2D NumPy array can be thought of as a matrix, where each element has two indices, row index and column index.**

**To slice a 2D NumPy array, we can use the same syntax as for slicing a 1D NumPy array.**

**The only difference is that we need to specify a slice for each dimension of the array and use comma ‘,’ for separating the rows and columns.**

**Syntax: array[ row\_start : row\_stop : row\_step , col\_start :**

**col\_stop : col\_step ]**

**~ row\_start : specifies starting index**

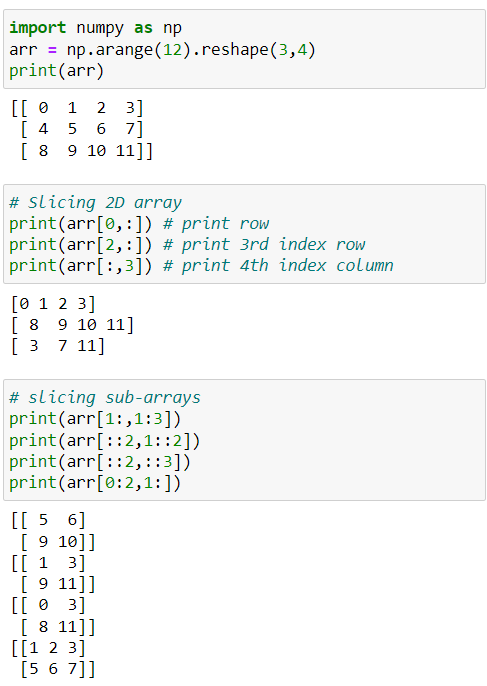
**~ row\_stop : stopping index**

**~ row\_step : step size for the rows respectively**

**~ col\_start : specifies starting index**

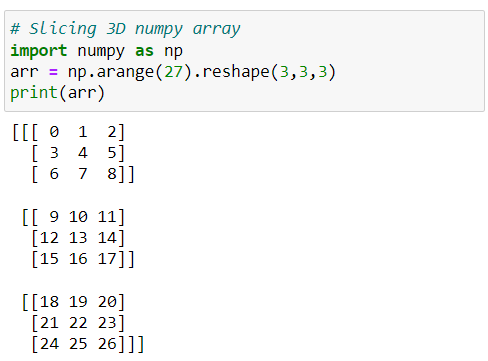
**~ col\_stop : stopping index**

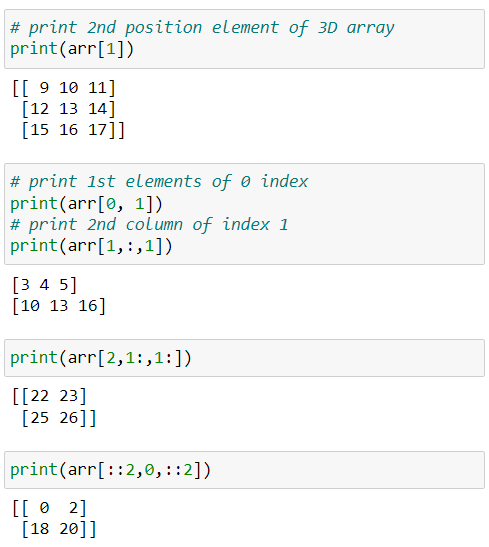
**~ col\_step : step size for the columns respectively**

****

**3D Slicing in numpy array:**

**A 2D NumPy array can be thought of as a matrix, where each element has two indices, row index and column index.**

****

****

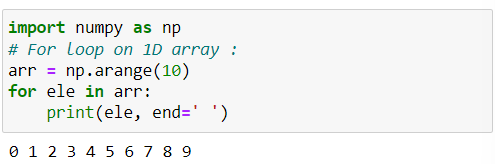
**Iteration On NumPy Array:**

**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.**

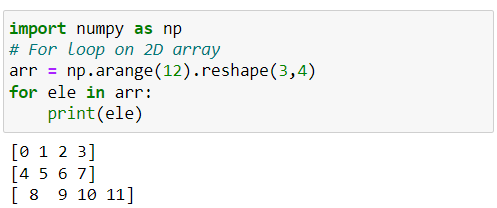
**Iterating on 1D numpy array:**

**it will go through each element one by one.**

****

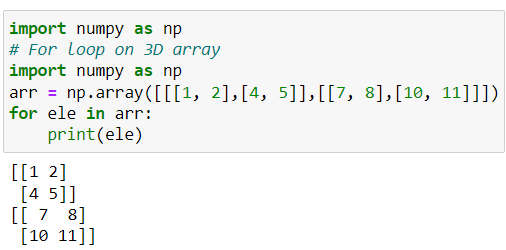
**Iterating on 2D numpy array:**

**It will go through all the rows.**

****

**Iterating on 3D numpy array:**

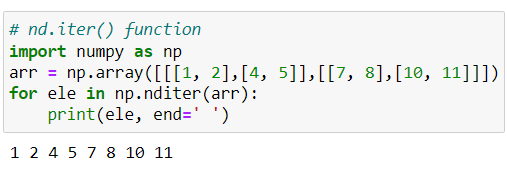
**It will go through all the 2-D arrays.**

****

**np.nditer() function:**

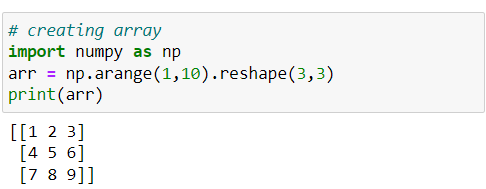
**It is a NumPy function that provides an efficient way to iterate over elements of a NumPy array.**

**It allows iterating over multiple arrays simultaneously and provides a number of optional arguments that can be used to customize the iteration process.**

****

**Reshaping in NumPy:**

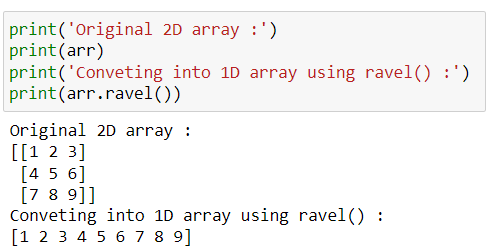
**In reshaping we commonly use reshape() and transpose() but sometimes we need to use revel() function**.

****

**.ravel():**

**Converting the n-dimensional array into flatten (1D) array.**

**Syntax:  arr.ravel() or np.ravel(arr)**

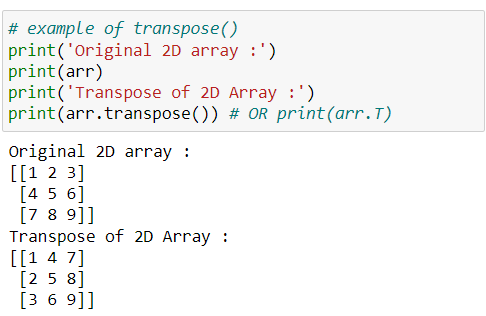
****

**.transpose() or .T:**

**Applied on 2D arrays to swipe the rows and columns of an array.**

**Using transpose() function or we can also use the short name .T to transpose a 2D array.**

**Syntax:  arr.transpose() or arr.T**

****

**Stacking in NumPy:**

**Stacking is the concept of joining arrays in NumPy.**

**Arrays having the same dimensions can be stacked.**

**We can stack arrays along different axes using the functions.**

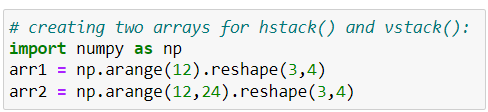
**~ column-wise represents axis=0**

**~ row-wise represents axis=1**

**~ np.hstack() : horizontal stacking**

**~ np.vstack() : vertical stacking  
Sometimes we have multiple data source means data come from databases, API and another data comes from web scrapping etc. so that data is similar data for multiple source then we can stack the data for data analysis.**

**Note: Shape/dimension of the array should be same**

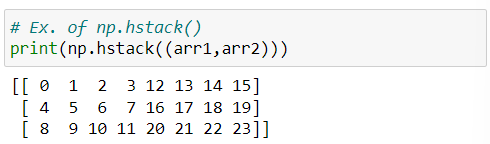
****

**np.hstack():**

**Horizontal stacking concatenates the arrays in sequence horizontally (column-wise).**

**This function stacks arrays horizontally (along axis 1)**

**Syntax:  np.hstack( (arr1, arr2) )**

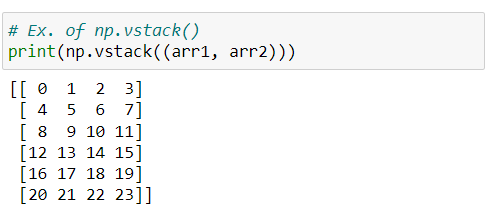
****

**np.vstack():**

**Vertical stacking means concatenates the arrays in sequence vertically (row-wise).**

**This function stacks arrays vertically (along axis 0).**

**Syntax:  np.vstack( (arr1, arr2) )**

****

**Splitting:**

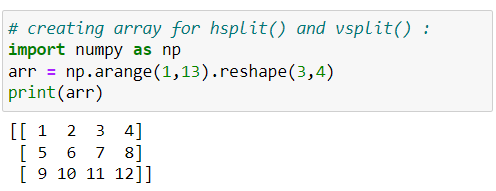
**Splitting is reverse operation of stacking.**

**We can split the arrays into sub-arrays of the same shape**

**~ np.hsplit(): horizontal splitting**

**~ np.vsplit(): vertical splitting.**

**Note: Only be used to split an array into sub-arrays of equal size**

****

**np.hsplit():**

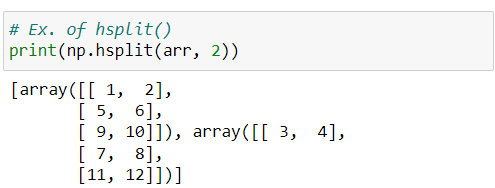
**hsplit() function is used to split a numpy array into multiple sub-arrays horizontally (column-wise).**

**Pass the input array and the number of sub-arrays as arguments.**

**Syntax:  np.split( arr, sub\_arrays\_size )**

**arr: input array**

**sub\_arrays\_size: number for splitting the array**

****

**np.vsplit():**

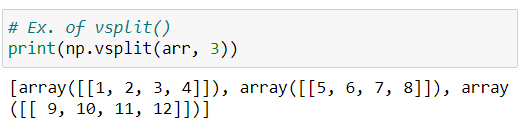
**vsplit() function is used to split a numpy array into multiple sub-arrays vertically (row-wise).**

**Pass the input array and the number of sub-arrays as arguments.**

**Syntax:  np.vsplit( arr, sub\_arrays\_size )**

**arr: input array**

**sub\_arrays\_size: number for splitting the array**

****