DS JULY 2022 Batch
Module 7 : Python Essential Packages

Topics

- NumPy
- Pandas
- Matplotlib

NumPy

NumPy

NumPy, which stands for Numerical Python, is a library consisting of multidimensional array objects and a collection of routines for processing those arrays.
Using NumPy, mathematical and logical operations on arrays can be performed.

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.

Installing NumPy

!pip install numpy

Importing NumPy

import numpy as np

Checking NumPy Version

np.__version__

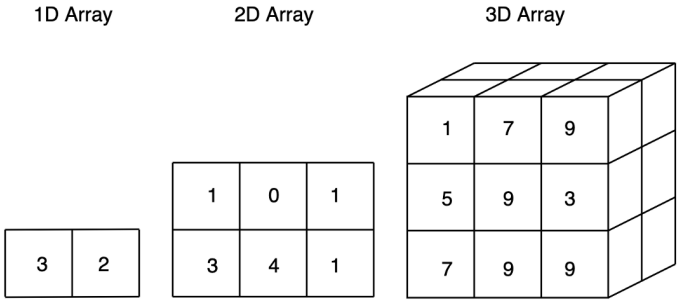
Arrays in NumPy

The most important object defined in NumPy is an N-dimensional array type called ndarray.
It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.
Every item in an ndarray takes the same size of block in the memory.

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

Syntax:

np.array(data, dtype,ndmin)



data : Elements of array
dtype : Data type of elements
ndmin : minimum dimension of array

0 dimension array : np.array(element)
1 dimension array : np.array([elements]) 1 dimension array
2 dimension array : np.array([[elements] , [elements]])
3 dimension array : np.array([[[elements] , [elements], [elements]]])

Higher dimension array : np.array([elements],ndmin=n)

Indexing and Slicing

NumPy array indexing

In NumPy indexing is similar to as we do in case of lists except for the fact that NumPy arrays can be multidimensional. You can access an array element by referring to its index number.

Syntax:

array_name[index 1,index 2,…index n] (In case of n dimensional array)

NumPy array Slicing

Slicing in python means taking elements from one given index to another given index.

If we want to access some elements from a particular dimension in an array then we have to pass the range in place of index.

Syntax :

array_name[index 1, start_index : last_index ,…index n]

Note: If you want to access all the elements of a particular dimension then we don't have to pass start and last index of the range.

Data Types of array

Data types

Data type of elements in an array can be checked using dtype method.

Syntax:

array_name.dtype

Changing data type of an array

We can change the data types of elements in an array using astype() function.

Syntax :

array_name.astype (new data type)

copy vs view

copy

- copy function is used to create a copy of an array.
- When we create a copy of an array it is altogether a new array.
- Whatever changes we make in original array will not reflect in copy of the array.

Syntax:

```
array_name.copy()
```

view

- view function is used to create a view of an array.
- When we create a view of an array it is just a reflection of old array.
- Whatever changes we make in original array will be reflect in view of the array.

Syntax:

```
array_name.view()
```

Reshape

Reshaping arrays

- Reshaping means changing the shape of an array.
- The shape of an array is the number of elements in each dimension.
- By reshaping we can add or remove dimensions or change number of elements in each dimension.

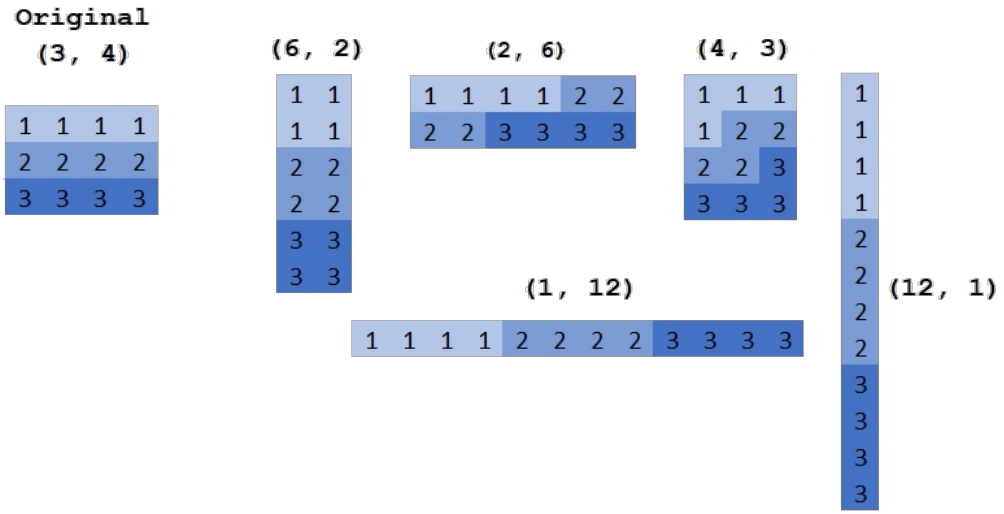
.

Syntax:

array_name.reshape(new dimensions)

Note : The new dimensions should be such that the Number of elements in new dimensions should be same as number of elements in old dimension.

Example : If we have a array of shape (4,3) then we can reshape it to an array (6,2).



Joining arrays

Joining numpy arrays

- Joining means putting contents of two or more arrays in a single array.
- We pass a sequence of arrays that we want to join to the concatenate() function, along with the axis.

Syntax:

np.concatenate((arr1, arr2),axis)

Stacking

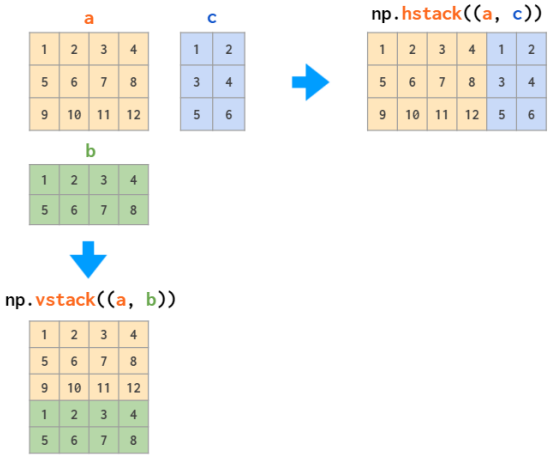
- Stacking is same as concatenation, the only difference is that stacking is done along a new axis.
- We can concatenate two 1-D arrays along the second axis which would result in putting them one over the other, ie. Stacking using stack() method.

Syntax:

np.stack((arr1, arr2), axis=1)

np.hstack((arr1, arr2), axis=1) (For horizontal stacking)

np.vstack((arr1, arr2), axis=1) (For vertical stacking)



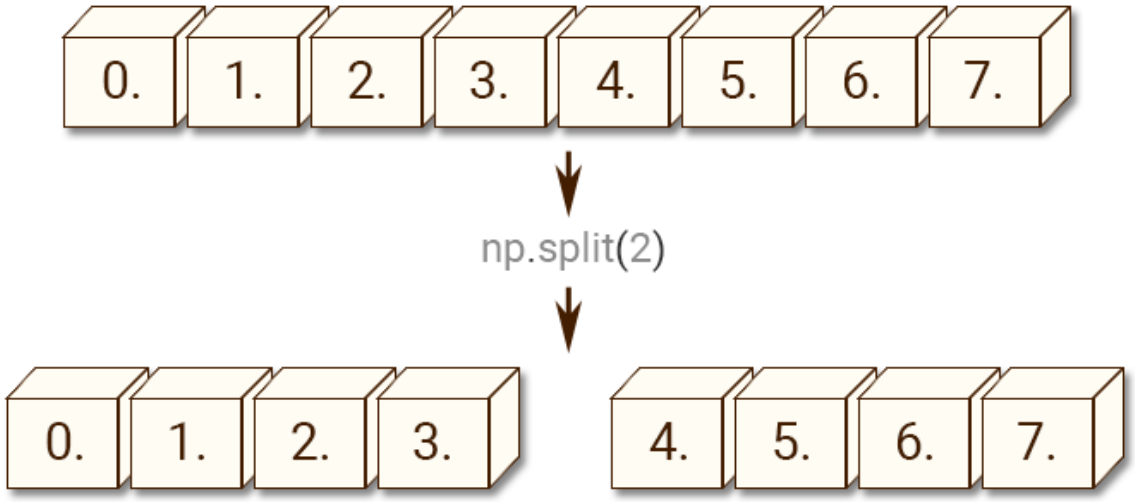
Splitting arrays

Splitting numpy arrays

- Splitting is reverse operation of Joining.
- Joining merges multiple arrays into one and Splitting breaks one array into multiple.
- We use array_split() for splitting arrays, we pass it the array we want to split and the number of splits.

Syntax:

np.array_split(array_name, num_split)



Array Search

Searching arrays

- You can search an array for a certain value, and return the indexes that get a match.
- To search an array, use the where() method.

Syntax:

np.where(condition)

Search Sorted

- There is a method called searchsorted() which performs a binary search in the array, and returns the index where the specified value would be inserted to maintain the search order.

Syntax:

np.searchsorted(array_name, element)

Array Sorting

Sorting arrays

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

Syntax:

np.sort(array)

Filtering arrays

- The NumPy ndarray can be filtered by passing a filter

Syntax:

array_name[filter]

Other Useful Functions

Functions	Description	Syntax
sum()	Sum of all the elements in the array	np.sum(array)
mean()	average of all the elements in the array	np.mean(array)
random.rand()	Generate random numbers between 0 and 1	np.random.rand(shape)
random.randn()	Generate random numbers from standard normal distribution	np.random.randn(shape)
random.randint()	Generate integers between two numbers	np.random.randint(low,high,size)
zeros()	Generate arrays of zero	np.zeros(shape)
ones()	Generate arrays of one	np.ones(shape)
log()	Returns natural logarithm of elements	np.log(array)
ceil()	Returns ceil values of elements	np.ceil(array)
floor()	Returns floor values of elements	np.floor(array)
unique()	Returns unique elements of an array	np.unique(array)
ravel()	Converts multidimensional array to one dimensional array	np.ravel(array)
dot()	returns dot product of two arrays	np.dot(array1,array2)
add()	Element wise addition operation on two arrays	np.add(array1,array2)
multiply()	Element wise multiplication of two arrays	np.multiply(array1,array2)
subtract()	Element wise subtraction operation on two arrays	np.subtract(array1,array2)