**BASICS OF NUMPY**

****

**What is NUMPY?**

* NumPy stands for Numerical Python.
* NumPy is a python library used for working with arrays.
* NumPy is a free Python library equipped with a collection of complex mathematical operations suitable for processing statistical data.

NumPy package has:

* A powerful N-Dimensional array
* Sophisticated Functions
* Tools for Integrating C/C++
* NumPy package is used to perform different operations.

**Basic functions of NumPy**

The main data structure in NumPy is the NumPy array. All the data is stored in arrays. NumPy provides a vast collection of mathematical and statistical operations which can be performed on these arrays.

**NumPy**

**Filtering Statistical Array Arithmetic Aggregation Reshaping Array**

**Models**

**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.

**You can import NumPy as-**

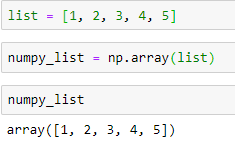
* import numpy as np

list = [1, 2, 3, 4, 5]

**We can produce a NumPy array called numpy\_list and display the result:**

numpy\_list = np.array(list)

numpy\_list #This line gives the result of the array created



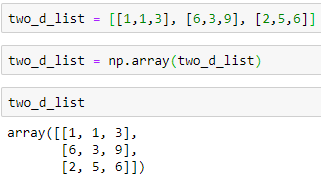
**What we have just done is casting a python list into a one-dimensional array. To get a 2-dimensional array, we have to cast a list of list as shown below.**

two\_d\_list= [[1,1,3], [6,3,9], [2,5,6]]

2d\_arr = np.array(two\_d\_list)

2d\_arr #This line gives the result of the array created

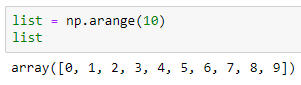
We have effectively created a 2-d array that has 3 rows and 3 columns.



**Forming NumPy array using arange() built-in function.**

Similar to the python built-in range() function, we will build a NumPy array using arange().

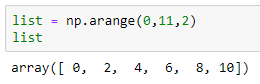
* list = np.arange(10) # (OR) list = np.arange(0,10)



This generates 10 digits of values from index 0 to 10.

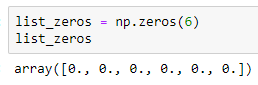
* It is essential to note that the arange() function can also take 3 arguments. The third argument signifies the step size of the operation. For example, to obtain all even numbers from 0 to 10, simply add a step size of 2 as below.

list = np.arange(0,11,2)



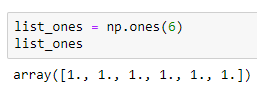
* We can also generate a one-dimensional array of six zeros.

list\_zeros = np.zeros(6)



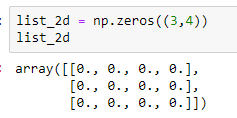
* We can also generate a one-dimensional array of six ones.

list\_ones = np.ones(6)



* Similarly, we could generate a two-dimensional array of zeros having 3 rows and 4 columns.

list\_2d = np.zeros((3,4))

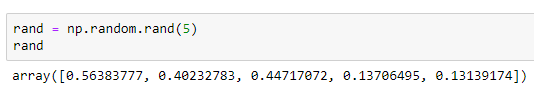


**Creating an array of random numbers in NumPy**

We can create an array of random numbers using rand(), randn() or randint() functions.

* Using random.rand(), we can generate an array of random numbers of the shape we pass to it from uniform distribution over 0 to 1.
* For example, say we want a one-dimensional array of 5 objects that are uniformly distributed from 0 to 1, we can do this:

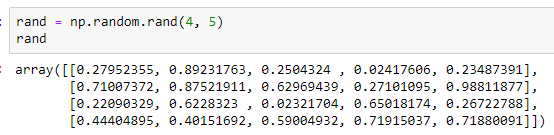
rand = np.random.rand(5)



* And if we want a two-dimensional array of 4rows and 5columns:

rand = np.random.rand(4, 5)

rand

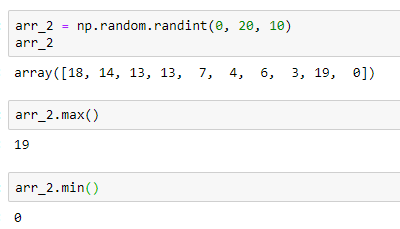
**Locating the maximum and minimum values of a NumPy Array**

* Using the max() and min(), we can get the maximum or minimum values in an array.

arr\_2 = np.random.randint(0, 20, 10)

arr\_2.max() #This provides the highest value in the array arr\_2.

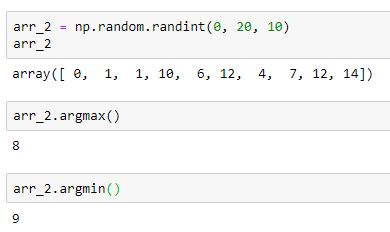
arr\_2.min() #This provides the lowest value in the array.



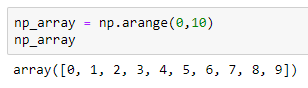
* Using the argmax() and argmin() functions, we can find the index of the maximum or minimum values in an array.

arr\_2.argmax() #This indicates the index of the highest value in the array.

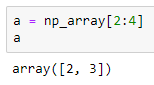
arr\_2.argmin() #This indicates the index of the lowest value in the array.



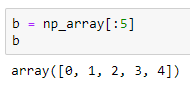
* To get a range of values in an array, we will use the slice notation ‘***:*’** just like in Python



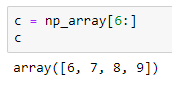
* np\_array[2:4] #This returns everything from index 2 to 4(exclusive).



* np\_array[:5] #This returns everything from index 0 to 5(exclusive).



* np\_array[6:] #This returns everything from index 6 to the end of the array.

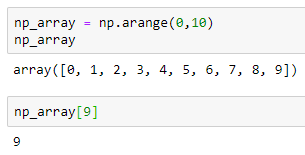


**Indexing/Selecting elements or groups of elements from a NumPy array.**

* Indexing NumPy arrays is related to that of Python. You simply pass in the index you want.

np\_array = np.arange(0,10)

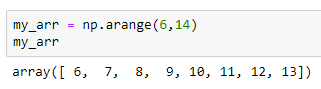
np\_array[5] #This gives us the value of element at index 5.



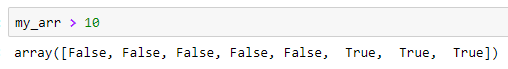
**Conditional Operators.**

* We can also perform conditional and logical selections on arrays using & (AND), | (OR), <, > and == operators to compare the values in the array with the given value. Here’s how:

my\_arr = np.arange(6,14)



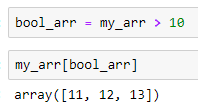
my\_arr > 10 #This returns TRUE where the elements are greater than 10



* Now we can print out the actual elements that were TRUE in the above conditional using:

bool\_arr = my\_arr > 10

my\_arr[bool\_arr] #This returns elements greater than 10 [11, 12, 13]

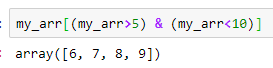


my\_arr[my\_arr>10] #A shorter way to do what we have just done.

* Using a combination of conditional and Logical & (AND) operators, we can get elements that are greater than 6 but less than 10.

my\_arr[(my\_arr>5) & (my\_arr<10)]

Our expected result is: ([6,7, 8, 9])

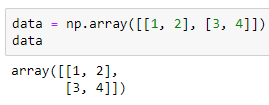


**Creating matrices**

You can pass Python lists of lists to create a 2-D array (or “matrix”) to denote them in NumPy.

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

data



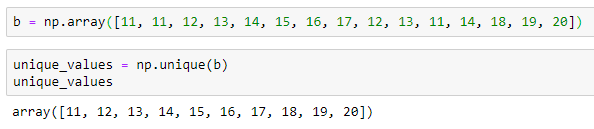
**How to get unique items.**

You can find the unique elements in an array easily with **np.unique.**

b = np.array([11, 11, 12, 13, 14, 15, 16, 17, 12, 13, 11, 14, 18, 19, 20])

unique\_values = np.unique(b)

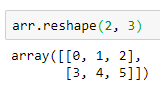
print(unique\_values)



**Transposing and reshaping a matrix**

It’s common to need to transpose your matrices. NumPy arrays have the property T that allows you to transpose a matrix.

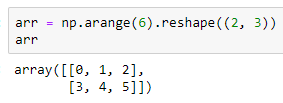
* arr.reshape(2, 3)



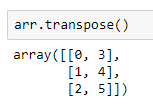
You can also use **.transpose** to reverse or change the axes of an array according to the values you specify.

* arr = np.arange(6).reshape((2, 3))

arr

****

* arr.transpose()

****

**How to create an array from existing data**

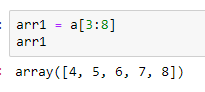
You can easily use create a new array from a section of an existing array.

Let’s say you have this array:

a = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])

arr1 = a[3:8]

arr1



You can also stack two existing arrays, both vertically and horizontally. Let’s say you have two arrays, a1 and a2:

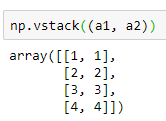
a1 = np.array([[1, 1],

[2, 2]])

a2 = np.array([[3, 3],

[4, 4]])

* np.vstack((a1, a2))



* np.hstack((a1, a2))

