NumPY

(A python library)

NumPy is a powerful library that provides easy access to efficient computational arrays from languages like C and Fortran. It offers many mathematical functions that include linear algebraic computing equations and random number generators along with Fourier transforms. Its application is widely accepted in data science, machine learning and scientific computing with Python. It’s not only fast but the broad concepts of how indexing is performed, and the vectorization makes it the best library to make use of in today’s era.

Installation Instructions:

There are two possible ways to install this library as of now.

* conda install numpy
* pip install numpy

Conda is an open-source tool that comes with Anaconda and Miniconda distribution. It’s used to manage packages and environments.

Pip is essentially the package installer for python which allows us to download packages from over the internet and make use of them in our python projects.

Using NumPy:

We can import NumPy to our environment using the following import statement:

* import numpy as np

Notice, np is an alias and it’s conventional for NumPy to be imported as np for our python projects.

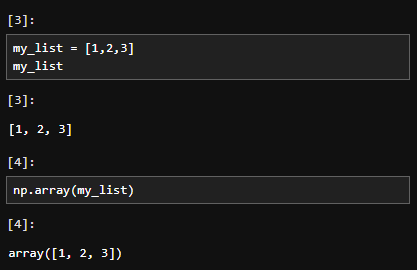
NumPy has many in-built functionalities and capabilities built into it, but we’ll focus on the important aspects such as:

* vectors
* arrays
* matrices
* number generation

NumPy Arrays:

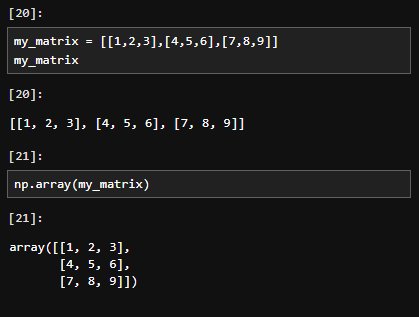
One of the widely used capabilities of this library is due to its arrays. These usually come as either matrices or vectors. Vectors are one-dimensional and matrices are two-dimensional.

Creating NumPy Arrays:



In the code snippet above, we use the array method from numpy library to generate a numpy array. It’s possible to convert a python list into a numpy array using this method.

Same goes for matrices.



Built-in Methods:

There are lots of built-in methods to generate arrays.

np.arange(start, stop, step) – works like the range function from python.

np.zeros(length) – generates an array with zeros

np.ones(length) – generates an array with ones

np.linspace(start, stop, num) - Return evenly spaced numbers over a specified interval.

np.eye(n) – creates a 2D array that has ones on its diagonal and zeros elsewhere

np.random.rand(n) – creates an array of the given shape with random samples from uniform distribution over [0, 1).

np.random.randn(n) – returns a sample from the Standard Normal Distribution of length n

np.random.randint(low, high) – Returns random integers from low (inclusive) to high (exclusive).

Array Attributes and Methods:

Some useful array methods and attributes are listed below.

arr.reshape(row, col) – returns an array containing the same data with a new shape.

arr.max(array) – returns the maximum value found in array

arr.min(array) – returns the minimum value found in array

arr.argmax(array) – returns the index of maximum value found in array

arr.argmin(array) – returns the index of minimum value found in array

arr.shape – Outputs the dimension size

arr.dtype – Outputs the data type

Bracket Indexing and Selection:

To get a value at an index, we use similar square bracket syntax, that we do for python lists.

* arr[8] gets the value at index 8
* arr[1:5] gets values in a range(1, 5)

Broadcasting:

Broadcasting allows us to set a value with index range in a numpy array.

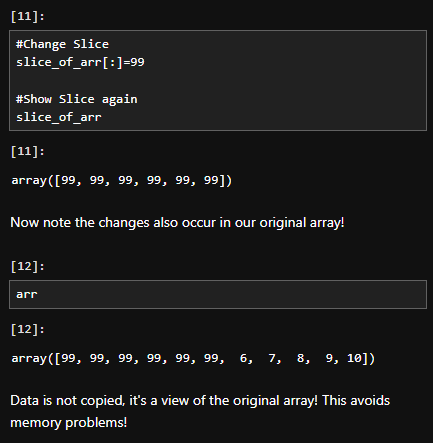
* arr[0:5] = 100 allows us to set a value of 100 for the range(5)

Consider the code snippet below:

We sliced the array and stored it in a variable slice\_of\_arr.

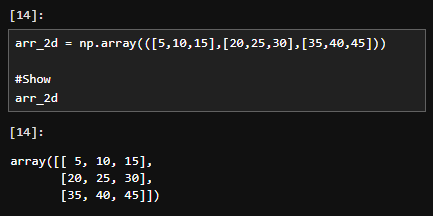
Now, if we broadcasted a value of 100 across the whole of slice\_of\_arr,

we’ll notice that the change has also occurred in our original array.



To get a copy, we should use the copy() method that allows us to make a copy of the array.

Indexing a 2D array (matrices):



The format is arr\_2d[row][col] or arr\_2d[row, col].

A screenshot of a computer program

Description automatically generated

We can also slice the 2D array. Consider the following code snippet.

A screenshot of a computer program

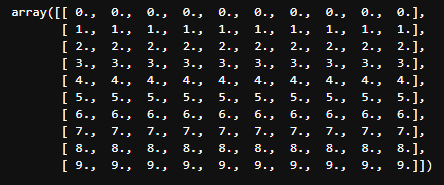
Description automatically generated

We use the colon to indicate the start/stop and the comma to differ rows from columns by indicating the row slice first and then the column slice.

Fancy Indexing:

It allows us to select entire rows or columns out of order.

Consider the following array.



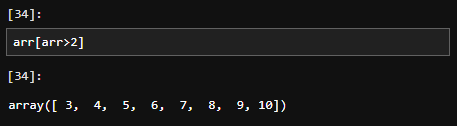
Fancy indexing allows the following.

A screenshot of a computer

Description automatically generated

Selection:

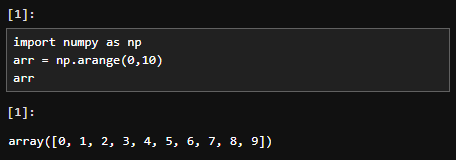
We can select contents based off comparison operators using such syntax.



NumPy Operations:

Arithmetic:

You can easily perform array with array arithmetic, or scalar with array arithmetic. Consider the following array.



We can perform arithmetic operations on arr and it’ll operate on all contents stored in arr.

A screenshot of a computer program

Description automatically generated

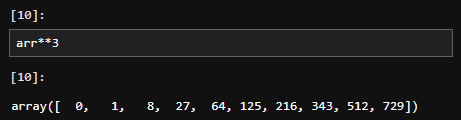
We know that arr[0] = 0, and when we consider arr / arr, the result must be not pretty. NumPy however, runs the statement and replaces the ZeroDivision content with nan. It also gives us a warning. The following code snippet shows this behavior.

A screenshot of a computer program

Description automatically generated

Similar behavior is encountered when we tried to divide 1 by arr. arr[0] becomes inf (short for infinity).

Exponents can also be taken this way.



Universal Array Functions:

Numpy comes with many universal array functions, which are essentially just mathematical operations you can use to perform the operation across the array.

* np.sqrt(array) – calculates the square root for each value in the array
* np.exp(array) – calculates the exponential for each value in the array
* np.max(array) – same as array.max()
* np.sin(array) – calculates the sin of values in array
* np.log(array) – calculates the logarithm of values in array

Further universal array functions can be found at [Universal functions (ufunc) — NumPy v2.0 Manual](https://numpy.org/doc/stable/reference/ufuncs.html).