To start using numpy, import it as such: import numpy as np

To initialize an array, can say: a = np.array([1, 2, 3, 4, 5, 6]).
When this is printed, will get “array([1, 2, 3, 4, 5, 6])

Can access various elements using square brackets. So a[0] = 1.
Numpy arrays are mutable (so a[0] = 10 can be applied and a visible change will be present)

Can also use python slice notation for indexing:

a[:3] will print ([10, 2, 3])

However, it’s important to note that making a copy of an array with a slice notation will just reference back the original array. So if b was a copy of a and included the last 3 elements of a, b[0] = (value) would surely modify b[0], but it would also modify the third-to-last element of a.

2D arrays can be initialized pretty easily - ex: a = np.array([1, 2, 3], [4, 5, 6], [7, 8, 9])

You can get the dimension of an array using a.ndim

Can get the shape of an array with a.shape. Here a shape depicts the number of elements in each dimension. So a.shape for the above would return 3, 3.

A.size: returns the number of elements (would return 9 for the 2d array above)

There are a ton of other basic array elements, like zeros(int s) (create an array with all s zeros), or ones, which functions the same.

There are easy sorting functions, like np.sort(), which will sort an array (so 5, 2, 3, 4, 1 will be sorted to 1, 2, 3, 4, 5)

np.concatenate() concatenates two arrays

Harder example: if you have the 2D array sample and concatenate it with [9, 4, 2] with np.concatenate((x, y), axis = 0), you’ll then have array([1, 2, 3], [4, 5, 6], [7, 8, 9], [9, 4, 2])

You can reshape an array. Using arr.reshape() gives a new shape to an array without changing the data. If you had an array that was [1, 2, 3, 4, 5, 6], you could make translate that into a 3x2 array

Np.reshape parameters: array, shape (new dimensions), and order

- C means to read/write using C-like index order
- F means to do it with Fortan like index order

- A means to do it with Fortian like index order if a is Fortran contiguous in memory, otherwise do it with C-like order

You can print arrays that are below a specific size
print (a[a < 5]) would print all numbers below 5.
You could have a new array and filter it based on elements that are at least a certain size, or even if it's divisible by some number.

You can select elements that satisfy two conditions using the & and | operator

If you have an array, you can make a new array based on the original array but it's filled with true/false based on if the specific element meets a condition.

So if you had the array (1, 2, 3, 4, 5) and you wanted to look for elements greater than 2, you could say new_arr = arr > 2. new_arr would then have (false, false, true, true, true)

You can add two arrays together that are the same dimension (just like how you'd do it in matrix algebra). If you have an array [1, 2] and an array [1, 1], you can add these two arrays by saying "a+b" (where a and b are the two arrays above) and it'll return [2, 3].