This video will explore the capabilities of numpy arrays.
The numpy module is installed automatically by ArcGIS. The module needs to be imported when working with numpy arrays.

Arcpy’s RasterToNumPyArray method converts a raster to an array format.

An array can also be created from a list.
Arrays can provide basic statistics on the data they contain.

These statistics include the maximum value, the minimum value, the mean, the standard deviation, and the variance.
Other methods of arrays include:

1) calculating the product of all items, 2) the cumulative product of the items, 3) the sum of all items, and 4) the cumulative sum of the items.
Arrays work with all standard math operators. The math operation is applied to each value in the array.
Arrays can be added, subtracted, multiplied, or divided.

When multiple arrays are used with a map operator, the data from the arrays are paired by their location in the array. Arrays must have compatible sizes in order to be used in math expressions.
This slide will compare processing speed of using an array with a math operator to equivalent methods using lists. The initial statement creates an array with 10 million random numbers.

The array is converted to a list using the `tolist` method. For the tests, we’ll triple each value in the list or array.

For the first list approach, we’ll use the list in a standard for loop to multiple each item by 3. This approach took 4.4 seconds.

The second list approach used a list comprehension – the speed was twice as fast as the for loop method.

The array approach was by far the fastest – 60 times faster than the 1st method and 30 times faster than the list comprehension.
This slide will compare the processing speed of a list and an array in a loop.

Iterating through the list took 0.3 seconds.

Iterating through an equivalent array took 16 times as long as for the list.

Arrays are much faster at math operations but lists are much faster when you need to iterate through the data collection.
Arrays can be transposed which switches the rows and columns.

The dot product of two arrays can be calculated.
The data in an array can be conveniently stored in a text file – this can be useful for temporary data storage when the memory in the array needs to be freed up. On the next slide, we’ll see an array method that can turn the text file back to an array.

The `tofile` method converts the array to a text file. All data in the text file are stored on a single line.
Text files can be converted to an array.

All values in the text file should be on a single line.

The `fromfile` method converts the text file to an array. This method along with the `tofile` method allows data to be easily transferred back and forth between internal memory and hard drive storage.
The array has methods similar to the range function for creating sequential values. Unlike the range or xrange functions, the array functions can create decimal numbers.

The `arange` function is equivalent to the `range` function except that it can create decimal numbers.

The `linspace` method creates a specific number of values between the start and stop value.
Numpy has a number of methods for creating new arrays. Numpy’s **ones** method creates an array in which all values are 1.

The **ones_like** method uses the shape of an existing list or array to create a new array in which all values are 1.

The **zeros** and **zeros_like** methods are equivalent to the previous methods except that they create arrays filled with zeros.
The **reshape** method changes the shape (i.e. number of rows and columns) of an array without changing the data.

The **unique** method creates a new array that contains only the unique values from the input array. This is equivalent to using the **set** function on a list.

The **trim_zeros** method will remove all zero values from the front or back end of a 1-dimensional array. A 1-dimensional array contains only a single row.
Arrays can be appended to an existing array using the `numpy.append` method. Note that a single value or a list or array of values can be appended. If an axis value of zero is specified, then the value will be appended to the end of the row; an axis value of 1 will append the value to the bottom of a column.

The `numpy.insert` method allows values to be inserted into any location in the array.

Note that for either the `append` or `insert` methods, the resulting array will be flattened if the axis is not specified.
Conditional expressions can be used to retrieve only the values from an array that satisfy the expression.

The **select** method allows expressions to be selectively applied to items in the array. The **condlist** parameter specifies one or more conditional expressions. The **choicelist** parameter contains one operation that corresponds to each expression in the **condlist** parameter. In this example, values in the input array that are <= 1 remain unchanged in the output array while values >=3 are squared.