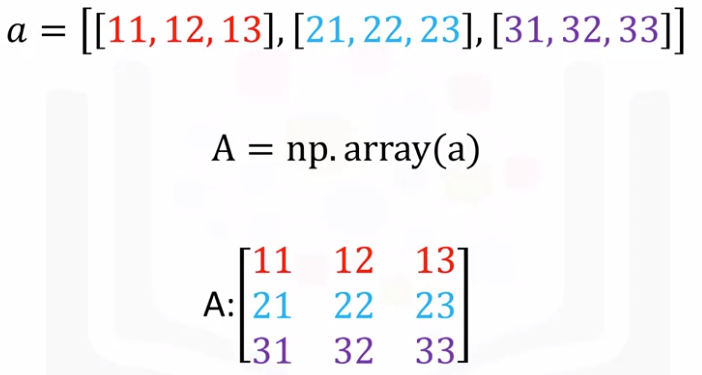
#### **Two Dimensional Numpy**

We can create numpy arrays with more than one dimension. This section will focus only on 2D arrays but you can use numpy to build arrays of much higher dimensions. In this video, we will cover the

* basics and array creation in 2D,
* indexing and slicing in 2D, and
* basic operations in 2D.

Consider the list a, the list contains three nested lists each of equal size. Each list is color-coded for simplicity. We can cast the list to a numpy array as follows.



It is helpful to visualize the numpy array as a rectangular array each nested lists corresponds to a different row of the matrix.

**Nidm - Number of Nested Lists**

We can use the attribute ndim to obtain the number of axes or dimensions referred to as the rank.

The term rank does not refer to the number of linearly independent columns like a matrix. It's useful to think of ndim as the **number of nested lists**.

The first list represents the first dimension (yellow). This list contains another set of lists (blue).



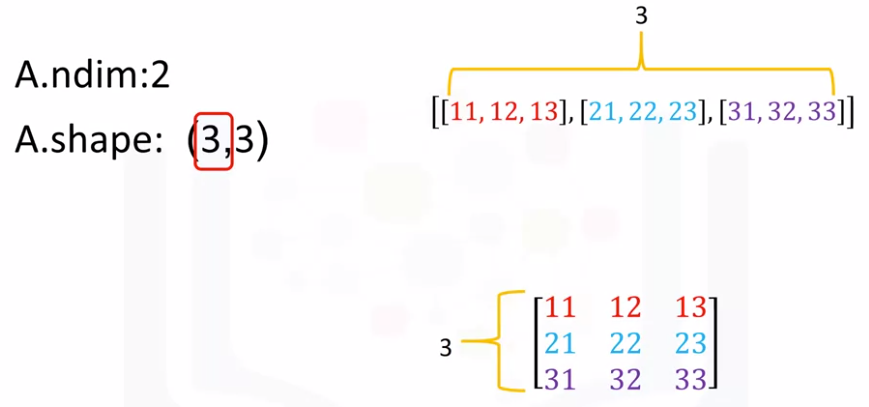
This represents the second dimension or axis.

The number of lists the list contains does not have to do with the dimension but the shape of the list.

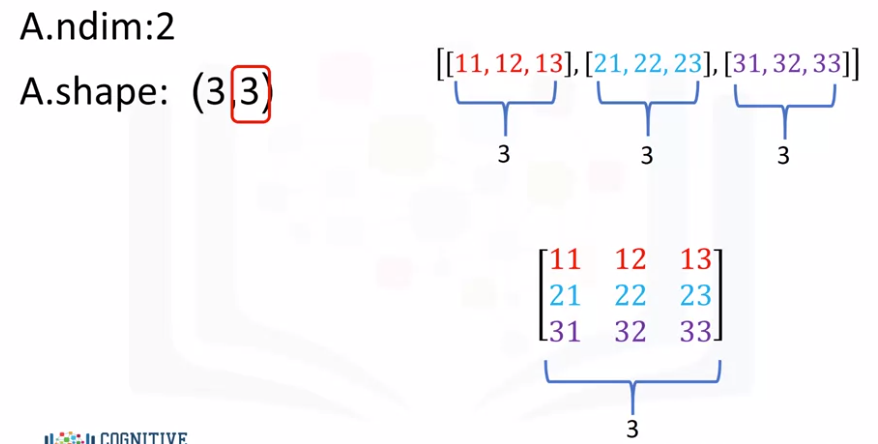
**Shape**

As with a 1D array, the attribute shape **returns a tuple**. It's helpful to use the rectangular representation as well.

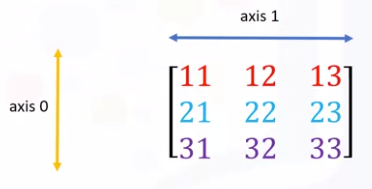
The first element in the tuple corresponds to the number of nested lists contained in the original list or the number of rows in the rectangular representation, in this case three.



The second element corresponds to the size of each of the nested list or the number of columns in the rectangular array zero.

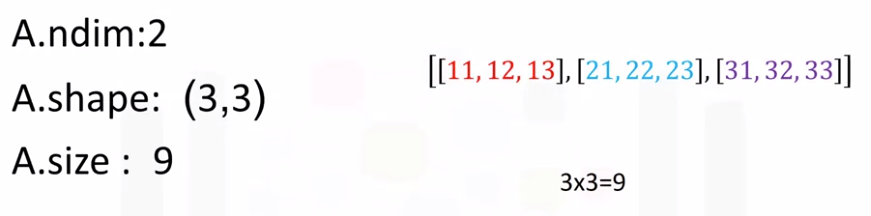


The convention is to label this axis zero and this axis one as follows.



We can also use the attribute size to get the size of the array. We see there are three rows and three columns.

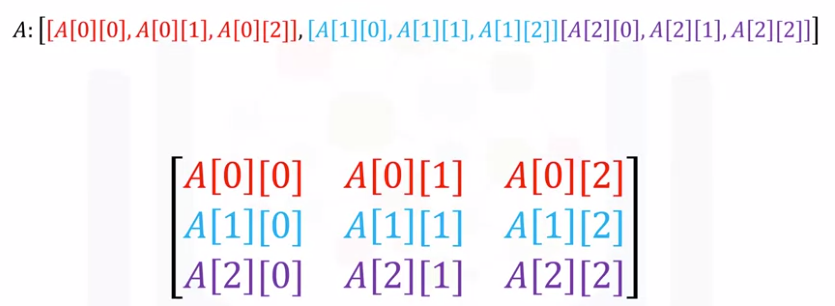
Multiplying the number of columns and rows together, we get the total number of elements, in this case nine.



Check out the labs for arrays of different shapes and other attributes.

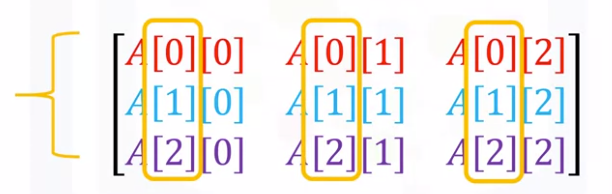
**Indexing**

We can use rectangular brackets to access the different elements of the array.

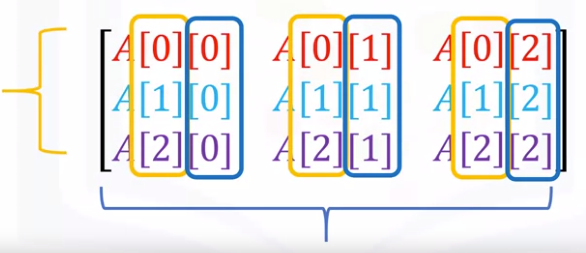


The following image demonstrates the relationship between the indexing conventions for the lists like representation.

The index in the first bracket corresponds to the different nested lists each a different color.



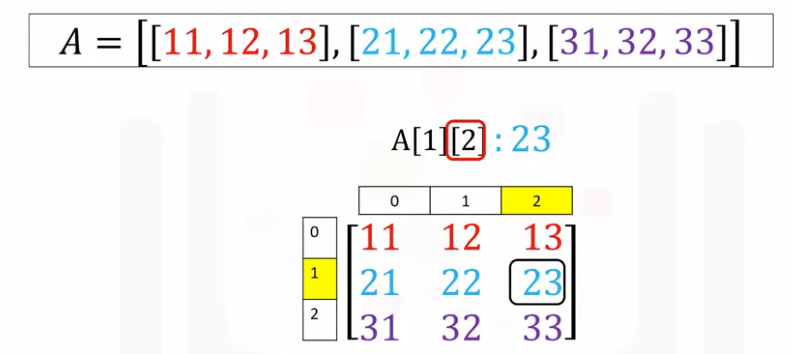
The second bracket corresponds to the index of a particular element within the nested list.



Using the rectangular representation, the first index corresponds to the row index. The second index corresponds to the column index.

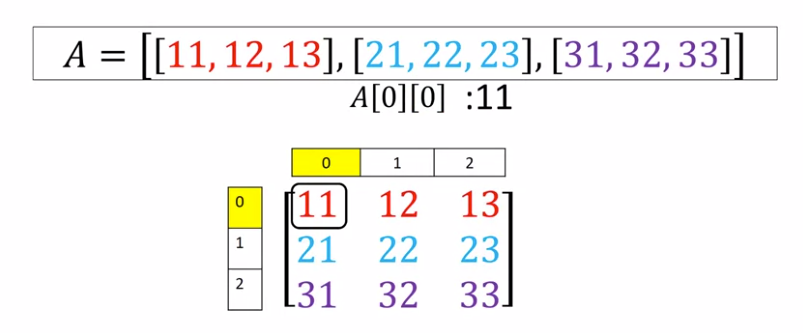
We could also use a single bracket to access the elements as follows.

Consider the following syntax.



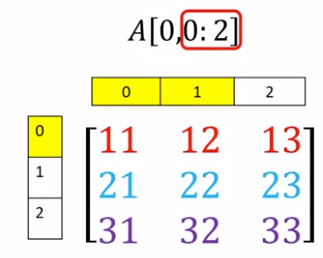
This index corresponds to the second row, and this index the third column, the value is 23.

Consider this example, this index corresponds to the first row and the second index corresponds to the first column, and a value of 11.

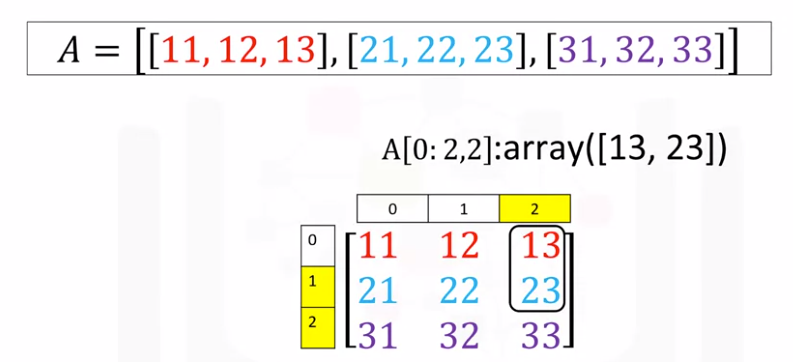


**Slicing Numpy Arrays**

We can also use slicing in numpy arrays. The first index corresponds to the first row. The second index accesses the first two columns.



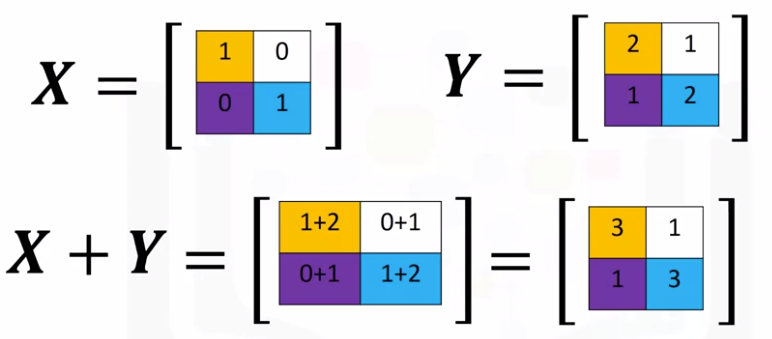
Consider this example, the first index corresponds to the last two rows. The second index accesses the last column.



**Addition**

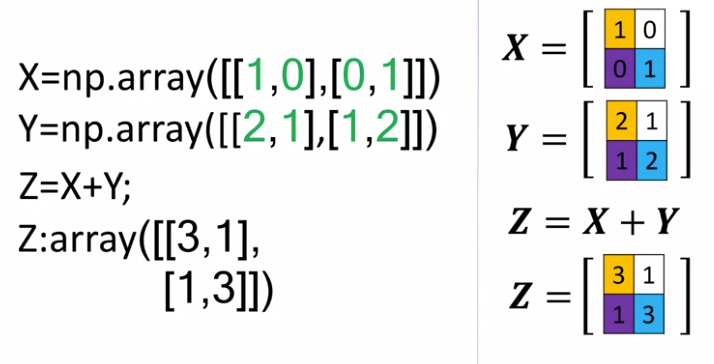
We can also add arrays, the process is identical to matrix addition. Consider the matrix X, each element is colored differently. Consider the matrix Y. Similarly, each element is colored differently.

We can add the matrices. This corresponds to adding the elements in the same position, i.e adding elements contained in the same color boxes together. The result is a new matrix that has the same size as matrix Y or X.



Each element in this new matrix is the sum of the corresponding elements in X and Y.

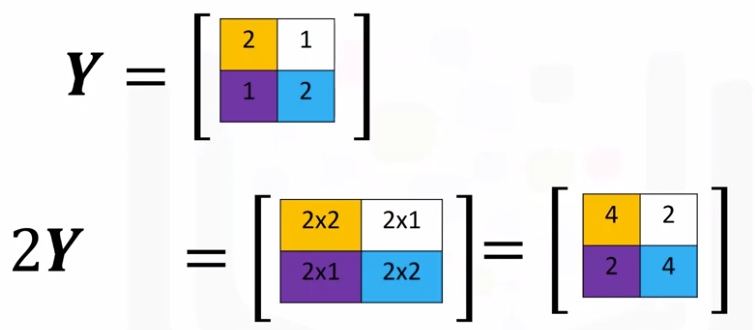
To add two arrays in numpy, we define the array in this case X. Then we define the second array Y, we add the arrays. The result is identical to matrix addition.



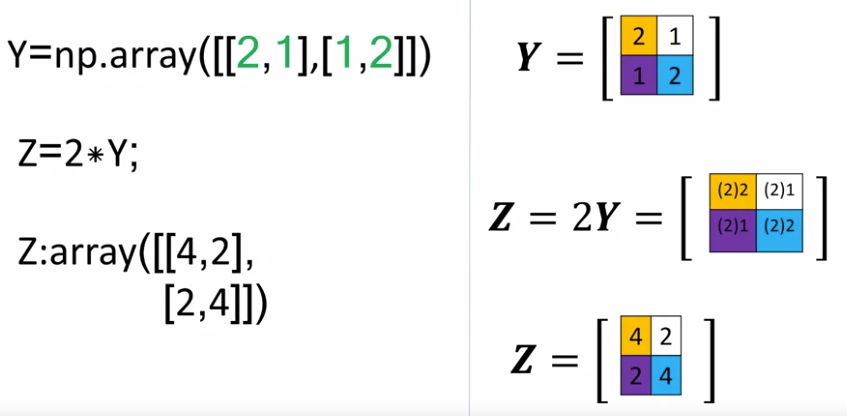
**Multiplication**

Multiplying a numpy array by a scalar is identical to multiplying a matrix by a scalar.

Consider the matrix Y. If we multiply the matrix by this scalar two, we simply multiply every element in the matrix by two. The result is a new matrix of the same size where each element is multiplied by two.



Consider the array Y. We first define the array, we multiply the array by a scalar as follows and assign it to the variable Z. The result is a new array where each element is multiplied by two.



Y=np.array([[2,1],[1,2]])

Z=2\*Y

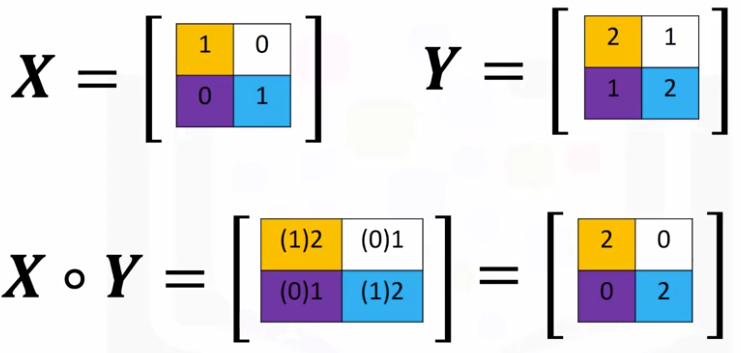
= array([[4, 2], [2, 4]])

**Hadamard product**

Multiplication of two arrays corresponds to an element-wise product, or Hadamard product.

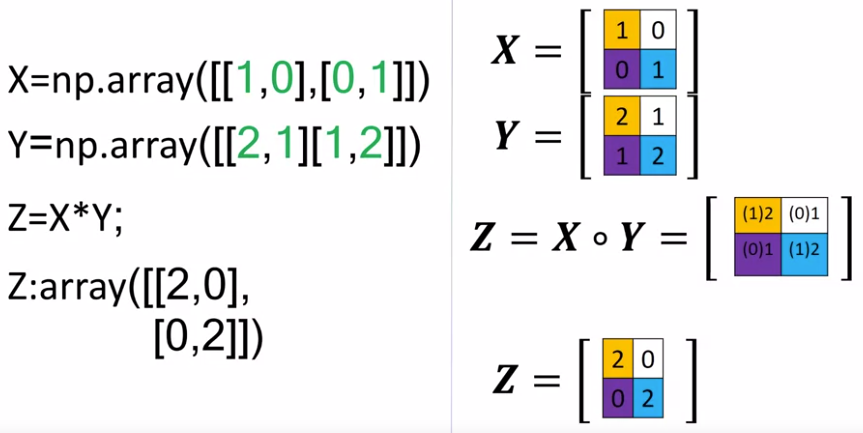
Consider array X and array Y.

Hadamard product corresponds to multiplying each of the elements in the same position i.e multiplying elements contained in the same color boxes together.



The result is a new matrix that is the same size as matrix Y or X. Each element in this new matrix is the product of the corresponding elements in X and Y.

Consider the array X and Y. We can find the product of two arrays X and Y in one line, and assign it to the variable Z as follows. The result is identical to Hadamard product.

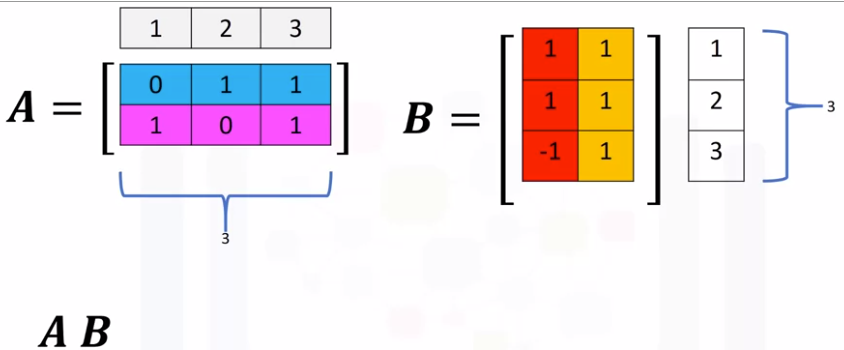


**Matrix Multiplication**

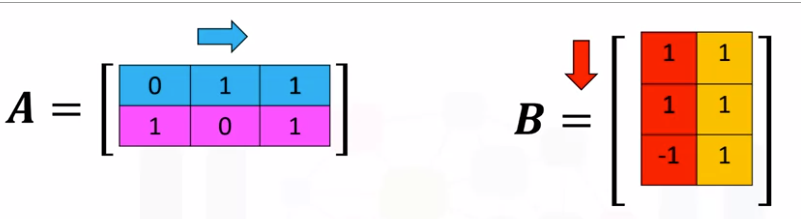
We can also perform matrix multiplication with Numpy arrays. Matrix multiplication is a little more complex but let's provide a basic overview.

Consider the matrix A where each row is a different color. Also, consider the matrix B where each column is a different color.

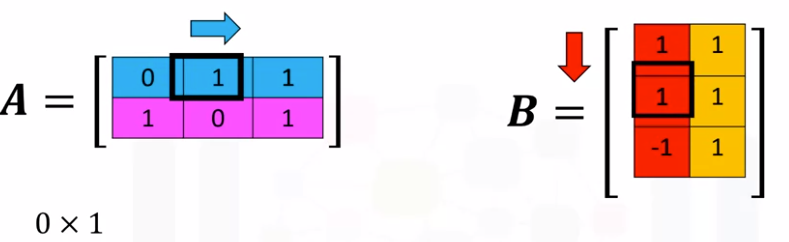
In linear algebra, before we multiply matrix A by matrix B, **we must make sure that the number of columns in matrix A in this case three is equal to the number of rows in matrix** B, in this case three.

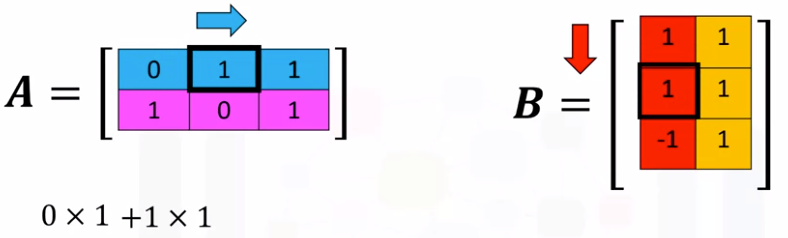


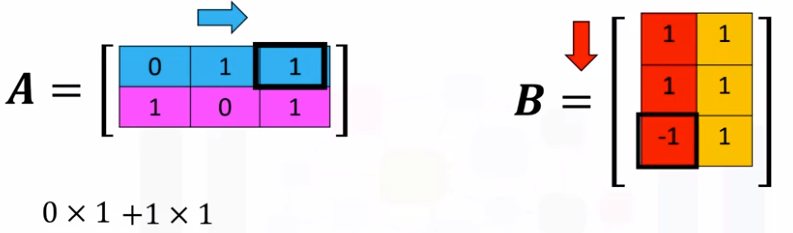
From matrix multiplication, to obtain the **ith** row and **jth** column of the new matrix, we take the dot product of the **ith** row of a with the **jth** columns of B.



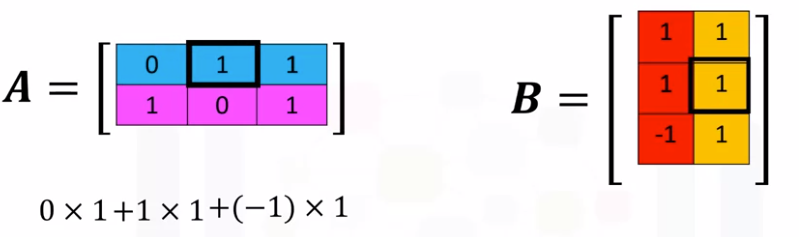
For the **first column, first row** we take the dot product of the first row of A with the first column of B as follows. The result is zero.

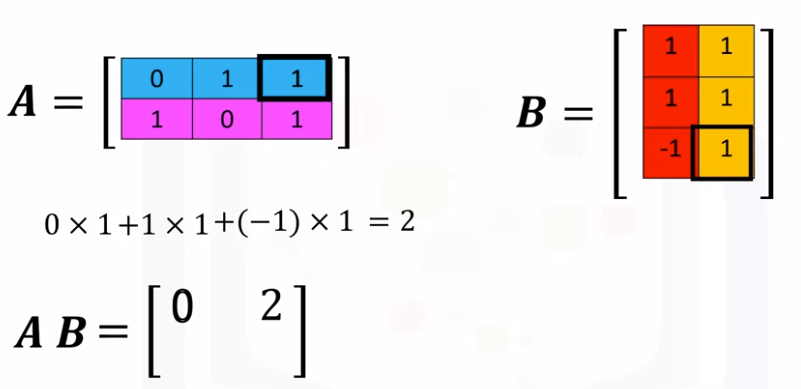




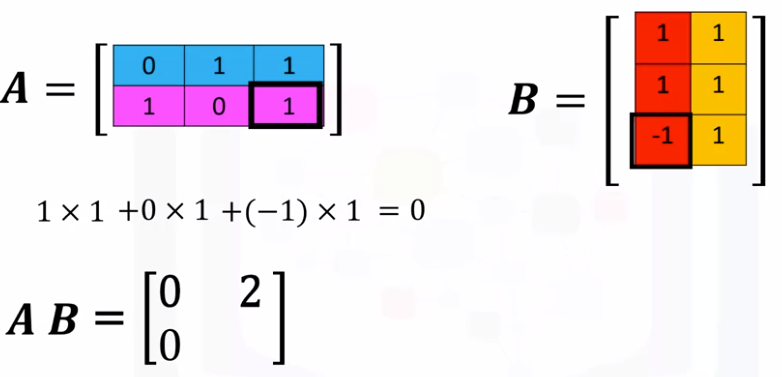


For the **first row and the second column** of the new matrix, we take the dot product of the first row of the matrix A, but this time we use the second column of matrix B, the result is two.

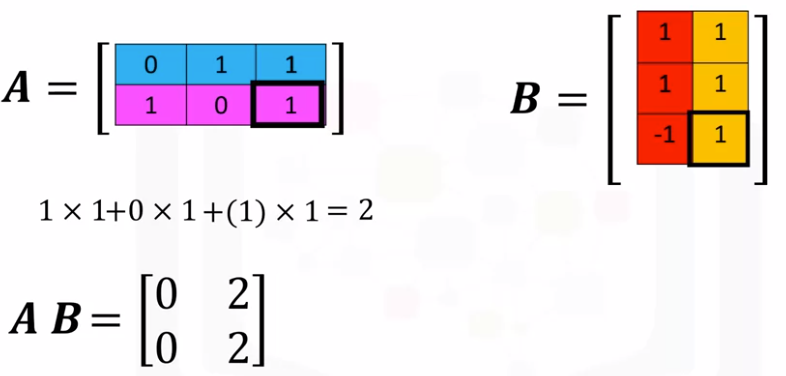




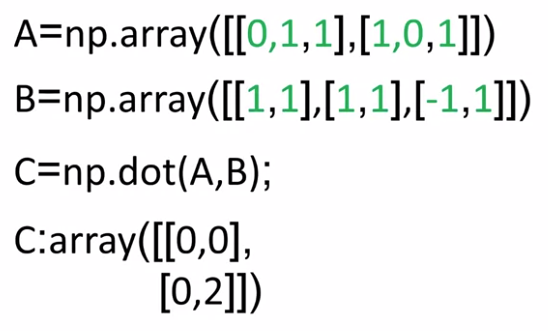
For the **second row and the first column** of the new matrix, we take the dot product of the second row of the matrix A. With the first column of matrix B, the result is zero.



Finally, for the second row and the second column of the new matrix, we take the dot product of the second row of the matrix A with the second column of matrix B, the result is two.



In numpy, we can define the numpy arrays A and B.



We can perform matrix multiplication and assign it to array C.

The result is the array C. It corresponds to the matrix multiplication of array A and B.

There is a lot more you can do with it in numpy. Checkout numpy.org. Thanks for watching this video.