```
# create a 3x3 array contain random normal numbers
my_3D = np.random.randn(3,3)
'Output':
array([[ 0.99709882, -0.41960273,  0.12544161],
       [-0.21474247,  0.99555079,  0.62395035],
       [-0.32453132,  0.3119651 , -0.35781825]])
# select a particular cell (or element) from a 2-D array.
my_3D[1,1]    # In this case, the cell at the 2nd row and column
'Output': 0.99555079000000002
# slice the last 3 columns
my_3D[:,1:3]
'Output':
array([[-0.41960273,  0.12544161],
       [ 0.99555079,  0.62395035],
       [ 0.3119651 , -0.35781825]])
# slice the first 2 rows and columns
my_3D[0:2, 0:2]
'Output':
array([[ 0.99709882, -0.41960273],
       [-0.21474247,  0.99555079]])
```

Matrix Operations: Linear Algebra

Linear algebra is a convenient and powerful system for manipulating a set of data features and is one of the strong points of NumPy. Linear algebra is a crucial component of machine learning and deep learning research and implementation of learning algorithms. NumPy has vectorized routines for various matrix operations. Let's go through a few of them.

Matrix Multiplication (Dot Product)

First let's create random integers using the method **np.random.randint(low, high=None, size=None,)** which returns random integers from low (inclusive) to high (exclusive).

```
# create a 3x3 matrix of random integers in the range of 1 to 50
A = np.random.randint(1, 50, size=[3,3])
B = np.random.randint(1, 50, size=[3,3])
# print the arrays
A
'Output':
array([[15, 29, 24],
       [ 5, 23, 26],
       [30, 14, 44]])
B
'Output':
array([[38, 32, 22],
       [32, 30, 46],
       [33, 47, 24]])
```

We can use the following routines for matrix multiplication, **np.matmul(a,b)** or **a @ b** if using Python 3.6. Using **a @ b** is preferred. Remember that when multiplying matrices, the inner matrix dimensions must agree. For example, if A is an $m \times n$ matrix and B is an $n \times p$ matrix, the product of the matrices will be an $m \times p$ matrix with the inner dimensions of the respective matrices n agreeing (see Figure 10-1).

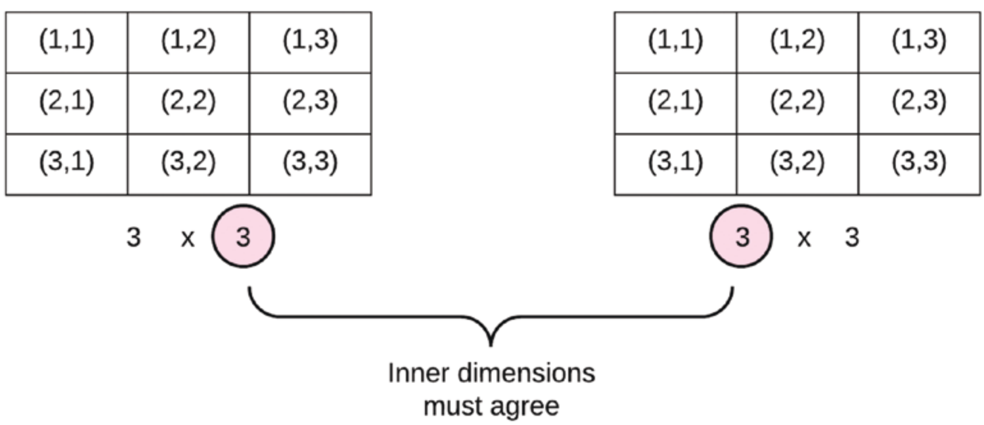

Figure 10-1. *Matrix multiplication*

```
# multiply the two matrices A and B (dot product)
A @ B    # or np.matmul(A,B)
```

```
'Output':
array([[2290, 2478, 2240],
       [1784, 2072, 1792],
       [3040, 3448, 2360]])
```

Element-Wise Operations

Element-wise matrix operations involve matrices operating on themselves in an element-wise fashion. The action can be an addition, subtraction, division, or multiplication (which is commonly called the Hadamard product). The matrices must be of the same shape. **Please note** that while a matrix is of shape $n \times n$, a vector is of shape $n \times 1$. These concepts easily apply to vectors as well. See Figure 10-2.

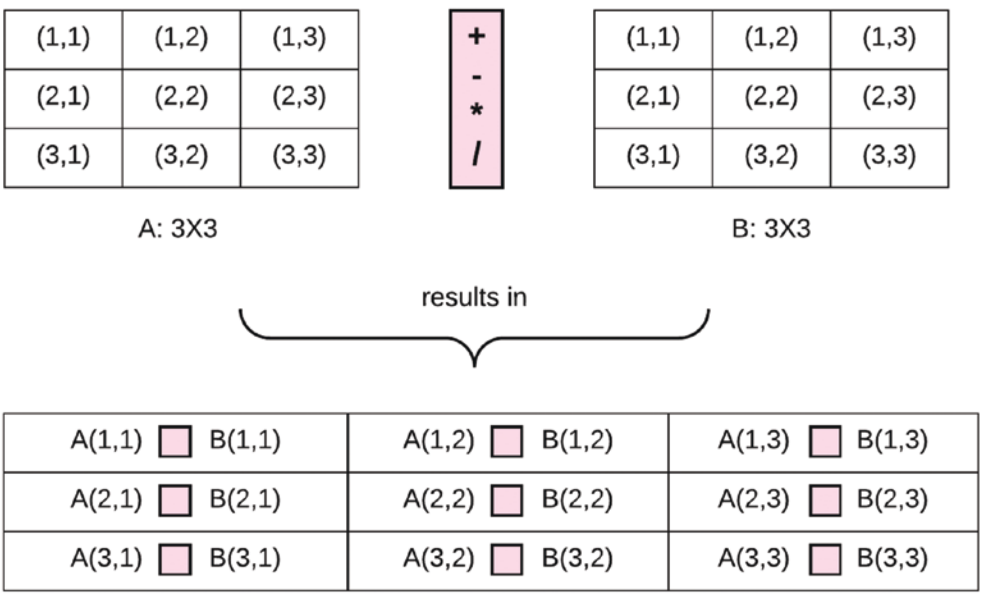

***Figure 10-2.** Element-wise matrix operations*

Let's have some examples.

```
# Hadamard multiplication of A and B
A * B
'Output':
array([[ 570,  928,  528],
       [ 160,  690, 1196],
       [ 990,  658, 1056]])
```