CHAPTER 1

Matrices

You have already gained experience with matrices earlier in this module, as well as when you have used spreadsheets. In this chapter, you will learn the types of matrices and get an introduction to some of the special matrices used for various calculations.

As you know, a matrix is a rectangular array of numbers arranged in rows and columns. The individual numbers in the matrix are called elements or entries. Matrices can be described by their dimensions. For example, a matrix with 2 rows and 3 columns is a 2 by 3 matrix.

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

More generally, a matrix with m rows and n columns is referred to as an $m \times n$ matrix, or simply an m-by-n matrix; m and n are its dimensions.

The general form of a 2×3 matrix A is:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \end{bmatrix}$$

1.1 Types of Matrices

Matrices can be described by their shape:

- **Row Matrix:** Has only one row.
- **Column Matrix:** Has only one column.
- **Square Matrix:** Has the same number of rows and columns.
- **Rectangular Matrix:** Has an unequal number of rows and columns.

They can also be described by their unique numerical properties. Special matrices that come in handy for certains types of computations. These are a few of the most common special matrices:

- **Zero Matrix:** Only contains entries that are zero.

- **Identity Matrix:** Sometimes called the unit matrix, it is a square matrix whose diagonal entries are 1 and all other entries are 0.
- **Symmetric Matrix:** A square matrix that equals its transpose. The next section shows how to create the transpose of a matrix,
- **Diagonal Matrix:** Has nonzero elements on the main diagonal, but all other elements are zero
- **Triangular Matrix:** This is a special square matrix that can be upper triangular or lower triangular. If upper, the main diagonal and all entries above it are nonzero while the lower entries are all zero. If lower, the main diagonal and all the entries below it are nonzero, while the upper entries are all zero.

1.1.1 Symmetric Matrices

If you want to find out if a square matrix is symmetric, you need to transpose it. If the transpose is equal to the original matrix, then the matrix is symmetric.

To transpose a matrix, flip it over its diagonal so that the rows and columns are switched, like this:

$$A = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

After transposing:

$$A^T = \begin{bmatrix} a_{1,1} & a_{2,1} & a_{3,1} \\ a_{1,2} & a_{2,2} & a_{3,2} \\ a_{1,3} & a_{2,3} & a_{3,3} \end{bmatrix}$$

Note that A^T means the transpose of A.

Let's see how this works for the following square matrix, A.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

Switch the rows and columns to get the transpose:

$$A^T = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

Notice that $A = A^T$, so the matrix is symmetric.

What about matrix B?

$$B = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 7 & 8 & 9 \end{bmatrix}$$

Switch the rows and columns to get the transpose:

$$B = \begin{bmatrix} 1 & 2 & 7 \\ 2 & 4 & 8 \\ 3 & 5 & 9 \end{bmatrix}$$

Note that $B \neq B^T$, so B is not symmetrical.

Exercise 1 Matrix Transposition

Find the transpose of this matrix. Is it symmetric?

$$A = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$

Working Space

Answer on Page 7

1.1.2 Creating Matrices in Python

Create a file called `matrices_creation.py` and enter this code:

```
# import the python module that supports matrices
import numpy as np
# Use the function np.array to define a matrix that
# contains specific values that you supply.
A = np.array([[ 5, 1, 3],
              [ 1, -1, 8],
              [ 6, 2, 1]])
# The transpose function returns
A.transpose()
```

When you run it, you should see:

```
array([[ 5, 1, 6],
       [ 1, -1, 2],
       [ 3, 8, 1]])
```

As you can see, $A \neq A^T$, so A is not symmetric. Try another:

```
# create a matrix, B
B = np.array([[ 5, 1, 6],
              [ 1, -1, 2],
              [ 6, 2, 1]])
B.transpose()
```

When you run it, you should see:

```
array([[ 5, 1, 6],
       [ 1, -1, 2],
       [ 6, 2, 1]])
```

B is symmetric. You can actually transpose any matrix using this function, but a matrix cannot be symmetric unless it is square.

Try this code to see what happens when you transpose a rectangular matrix.

```
# create a matrix, J
J = np.array([[ 5, 1, 3, 0],
              [ 1, -1, 8, 11],
              [ 6, 2, 1,-7]])
J.transpose()
```

Note that transposing a rectangular matrix changes its dimension from 3 by 4 to 4 by 3. You should see a transposed matrix, but it's not symmetric.

```
array([[ 5,  1,  6],
       [ 1, -1,  2],
       [ 3,  8,  1],
       [ 0, 11, -7]])
```

1.1.3 Creating Special Matrices in Python

Use the same file to add this code for creating a zero matrix.

```
# create an 8 by 10 Zero matrix.
 C = np.zeros((8, 10))
 C
```

When you run it, you should see:

```
array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

Add the following code to create an 8 by 8 Identity matrix.

```
# create an 8 by 8 Identity matrix
 D = np.eye(8)
 D
```

When you run it, you should see:

```
array([[1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1.]])
```

As you progress in your studies, you will learn the importance of diagonal matrices and of extracting the diagonal of a matrix. Let's see how to extract a diagonal, then create a diagonal matrix.

```
# create a  matrix
W = np.array([[1, 2, 3, 4],
              [5, 6, 7, 8],
              [-8, -7, -6, -5],
              [-4, -3, -2, -1]])
```

Extract the main diagonal using np.diag(<array>,<diagonal to extract>). Passing 0 as the second parameter specifies the main diagonal. A positive value extracts a diagonal from the upper part. A negative value extracts a diagonal from the lower part. Run this code then experiment passing other values to see what you get.

```
print(np.diag(W,0))
```

When you run it, you should see:

```
array([ 1,  6, -6, -1])
```

You can also use np.diag() to create a diagonal matrix from a 1D array. In this case, do not pass a second paramenter.

```
Q = np.array([1, 2, 3])
DiagArray = np.diag(Q))
print(DiagArray)
```

When you run it you should see;

```
[[1 0 0]
 [0 2 0]
 [0 0 3]]
```

Python has functions for extracting upper and lower triangular matrices. Try these:

```
print(np.triu(W))
print(np.tril(W))
```

You should see:

```
[[ 1  2  3  4]
 [ 0  6  7  8]
 [ 0  0 -6 -5]
 [ 0  0  0 -1]]

[[ 1  0  0  0]
 [ 5  6  0  0]
 [-8 -7 -6  0]
 [-4 -3 -2 -1]]
```

Appendix A

Answers to Exercises

Answer to Exercise 1 (on page 3)

$$A = A^t = \begin{bmatrix} 3 & -2 & 4 \\ -2 & 6 & 2 \\ 4 & 2 & 3 \end{bmatrix}$$