Computers are used to perform a seemingly endless variety of tasks. However, no matter what the task, all computer software requires the same thing to operate... data. In fact, in the simplest terms, all computer tasks consist of the reading, manipulating, storing, or displaying of data.

**Data Structures:**

The term data can take on many meanings, but for our purposes it is information that is in some form that is suitable for processing by a computer. So, what form can information take that is convenient for computers to process? There are three different forms that data is commonly found in. These three data structures are **scalars**, **arrays**, and **matrices**. As the term "structure" suggests, the difference between these three types is how the data is arranged.

A **scalar** is a single figure or quantity. The quantity may be a number, a text string, or any other acceptable quantity of data. For example, consider the following:

\[
a = 1 \\
b = 500 \\
n = Bob
\]

Each of these variables, \(a\), \(b\), and \(n\) are scalar variables whose values are 1, 500, and Bob, respectively. As you might guess, the scalar is the most fundamental of all the data structures.

An **array** is a one-dimensional collection or list of quantities. Again, these quantities may be numbers, text strings, or any other acceptable quantity of data. For example, consider the following:

\[
\begin{array}{ccc}
\text{names} & = & \begin{bmatrix} \text{Bob} & \text{Sue} & \text{Joe} \end{bmatrix} \\
\text{ages} & = & \begin{bmatrix} 21 & 18 & 19 \end{bmatrix} \\
\text{cars} & = & \begin{bmatrix} \text{Ford} & \text{Chevy} & \text{Dodge} \end{bmatrix}
\end{array}
\]

Each of these variables, \(\text{names}\), \(\text{ages}\), and \(\text{cars}\) are array variables. The array variable \(\text{names}\) contains three values: Bob, Sue, and Joe. The array variable \(\text{ages}\) contains three values: 21, 18, and 19. The array variable \(\text{cars}\) contains three values: Ford, Chevy, and Dodge.

A **matrix** is a two-dimensional collection of quantities. As before, these quantities may be numbers, text strings, or any other acceptable quantity of data. For example, consider the following:

\[
\text{data} = \begin{bmatrix} \\
\text{Bob} & \text{Sue} & \text{Joe} \\
21 & 18 & 19 \\
\text{Ford} & \text{Chevy} & \text{Dodge} \\
\end{bmatrix}
\]

The variable \(\text{data}\) is a matrix that has three rows and three columns. This type of matrix, where the number of rows equals the number of columns, is referred to as a square matrix.
Data Structure Relationships:

The relationship between the three types of data structures is interesting if not trivial. For starters, an array is a collection of scalars. However, an array may also be thought of as a matrix with either only one row or one column.

A matrix may be thought of as a collection of scalars as well as a collection of arrays. Think of it in these terms, since an array is a collection of scalars, and a matrix is a collection of arrays, then a matrix is also a collection of scalars.

Interestingly enough, the reverse logic is true also. For example, a scalar may be thought of as an array with only one item in it. A scalar may also be thought of as a matrix with only one row and one column.

To sum things up, here are some question and answer pairs to illustrate these ideas. You should be able to understand the following:

- Is a vector a matrix? Yes
- Is a scalar a vector? Yes
- Is a scalar a matrix? Yes
- Are you confused? Probably

Subscripts:

Now that we know the forms in which data is stored, we need to learn how to get individual pieces of data from arrays and matrices. This is accomplished with subscripting or indexing of the array or matrix variables.

Arrays only require one subscript or index because they are a simple one-dimensional list, either a column or a row, unlike a matrix. Take the following array for example:

\[
c = \begin{bmatrix}
  1 & 2 & 3 & 4 \\
  5 & 6 & 7 & 8 \\
  9 & 10 & 11 & 12
\end{bmatrix}
\]

The array variable \( c \) contains 4 elements or entries. If you want to do something with the first element in the array, you can reference it by placing a subscript or index of 1 on the variable as shown.

Matrices, since they are two-dimensional, require two indices or subscripts. One index is for the column and one index is for the row. There is a particular order: the first index is the row index and the second index is the column index. Consider the following matrix for example:

\[
d = \begin{bmatrix}
  1 & 2 & 3 & 4 \\
  5 & 6 & 7 & 8 \\
  9 & 10 & 11 & 12
\end{bmatrix}
\]

\[
\begin{align*}
d(1, 2) &= 2 \\
d(2, 1) &= 5 \\
d(1, 1) + d(1, 2) &= 1 + 2 = 3 \\
d(2, 4) &= 8 \\
d(2, 3) + d(3, 1) &= 7 + 9 = 16 \\
d(3, 4) &= 12
\end{align*}
\]
Transpose:

One common operation performed on a vector or matrix is called the transpose of the vector or matrix. When a matrix is transposed, the first row in the matrix becomes the first column in the matrix, the second row in the matrix becomes the second column in the matrix, and so on. So, if you have a vector, a row vector becomes a column vector and a column vector becomes a row vector when transposed. The transpose operator is an apostrophe placed after the matrix or vector. See the following examples of the transpose operator:

\[
\begin{bmatrix}
2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
\r
\text{if: } e = \r
\begin{bmatrix}
2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9
\end{bmatrix}
= \left[
\begin{array}
\begin{bmatrix}
2 \\
4 \\
7
\end{bmatrix}
\end{array}
\begin{bmatrix}
3 \\
5 \\
8
\end{bmatrix}
\begin{bmatrix}
6
\end{bmatrix}
\end{bmatrix}
\text{then: } e' = \begin{bmatrix}
4 & 5 & 6
\end{bmatrix}
\]

\[
\begin{bmatrix}
2 & 3 & 4 \\
5 & 6 & 7 \\
8 & 9 & 1
\end{bmatrix}
\r
\text{if: } f = \r
\begin{bmatrix}
2 & 3 & 4 \\
5 & 6 & 7 \\
8 & 9 & 1
\end{bmatrix}
\text{then: } f' = \left[
\begin{array}
\begin{bmatrix}
2 \\
5 \\
8
\end{bmatrix}
\begin{bmatrix}
3 \\
6 \\
9
\end{bmatrix}
\begin{bmatrix}
4
\end{bmatrix}
\end{array}
\right]