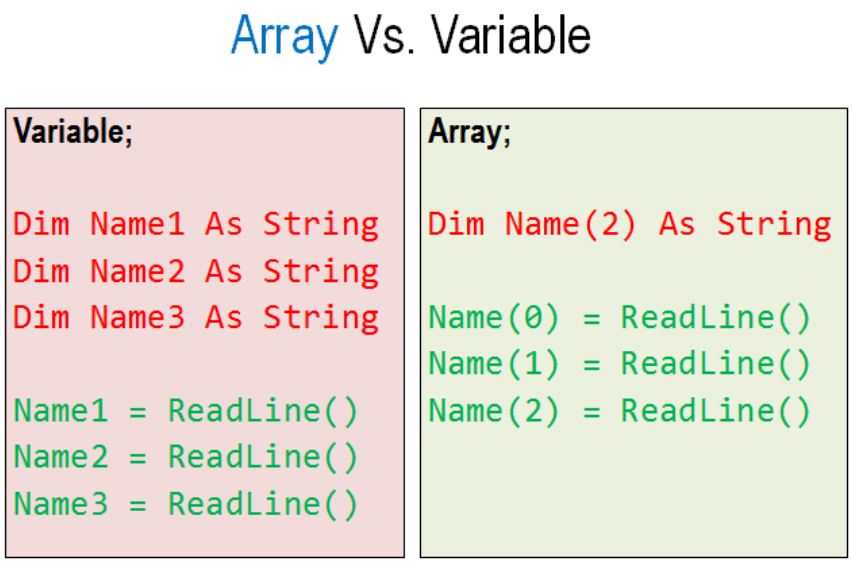
Using Arrays

What is an Array?

Currently you have been using variables to store one piece of information at a time; for example:

```
Dim X As Integer = 5
```

Although variables are useful, storing only one piece of information at each instance will make some tasks very tedious. Fortunately, an array is a type of variable that allows a programmer to store multiple values as opposed to a single value. Arrays are available in most programming languages and there are also many different types; we shall focus on native arrays.



Open Notepad or Notepad++ and type the following code (pay attention to the indentation used). Once the code has been typed up, compile and run the program. In this example, a one-dimensional array is used to store three name items. A one-dimensional array could be visualised as a single column in a spreadsheet. Notice that arrays always start at position 0, not position 1.

```
Imports System.Console
Public Class UsingArrays
    Shared Sub Main()
        Dim FirstNames(2) As String
        FirstNames(0) = "Joe"
        FirstNames(1) = "James"
        FirstNames(2) = "Alex"
        WriteLine(FirstNames(0))
        WriteLine(FirstNames(1))
        WriteLine(FirstNames(2))
    End Sub
End Class
```

Test Your Skills

- ✓ *Write a short application that stores your favourite films in a one-dimensional array and then writes your list of favourite films back to the user.*

Arrays and Loops

Outputting the results of an entire array individually (i.e. displaying each array item using separate **WriteLine** commands) can be a tedious process, and is inefficient and costly (in terms of time and LOC (Lines Of Code)). Fortunately, iteration (loops) can be used to output multiple items of an array in just a few lines of code; most programming languages support this functionality. An example of this is shown below; now try it yourself!

```
Shared Sub Main()
        Dim FirstNames(3) As String
        FirstNames(0) = "Joe"
        FirstNames(1) = "James"
        FirstNames(2) = "Alex"
        FirstNames(3) = "Bobby"
        For X = 0 To 3
                WriteLine(FirstNames(X))
        Next X
End Sub
```

Determining the length of an array is important because there are occasions when a programmer will not know the specific number of items in an array. There is a function that is part of the **array** class in VB.NET (and the same principle in other languages) that allows a programmer to determine the number of items in an array. The function in VB.NET is called **UBound()**. The first argument in **UBound()** specifies the name of the array; the second specifies the dimension. An example of the function is shown below:

```
Dim X As Integer = UBound(ArrayName, 1)
```

This example shows the **UBound()** function implemented in code:

```
Shared Sub Main()
        Dim FirstNames(2) As String
        FirstNames(0) = "Joe"
        FirstNames(1) = "James"
        FirstNames(2) = "Alex"
        For X = 0 To UBound(FirstNames, 1)
            WriteLine(FirstNames(X))
        Next X
End Sub
```

The following example uses two loops: one to read the values from the user to the array, and another to write the responses back to the screen; try this one yourself!

```
Shared Sub Main()
        Dim FirstNames(4) As String
        Dim X As Integer = 0
        WriteLine("Please enter five first names;")
            For X = 0 To UBound(FirstNames, 1)
                FirstNames(X) = ReadLine()
                X = X + 1
            Next X
        WriteLine("These are your first names;")
            For X = 0 To UBound(FirstNames, 1)
                WriteLine(FirstNames(X))
            Next X
End Sub
```

Two-Dimensional Arrays

Arrays do not have to be one-dimensional; in fact, they can have numerous dimensions. A two-dimensional array is similar to that of a spreadsheet; it is like a grid in memory that consists of rows and columns, and each cell has a unique reference. As arrays start at 0, a 3-by-3 two-dimensional array could be visualised similar to that of the image below. A coded example has also been provided.

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3
3,0	3,1	3,2	3,3

```
    Shared Sub Main()
        Dim Details(2, 2) As String
        WriteLine("Please enter two first names;")
                Details(0, 0) = ReadLine()
                Details(1, 0) = ReadLine()
        WriteLine("Please enter two last names;")
                Details(0, 1) = ReadLine()
                Details(1, 1) = ReadLine()
        WriteLine("Please enter two ages;")
                Details(0, 2) = ReadLine()
                Details(1, 2) = ReadLine()
        WriteLine("Here are your details;")
        WriteLine(Details(0, 0) & " " & Details(0, 1) & " " & Details(0, 2))
        WriteLine(Details(1, 0) & " " & Details(1, 1) & " " & Details(1, 2))
End Sub
```