Using Iteration

If you were asked to write a simple program that displays 1 to 5 in the console window, with each number on a separate line, what would your code look like? Would it look like this?

```
Imports System.Console
Public Class UsingIFS
    Shared Sub Main()
        WriteLine("1")
        WriteLine("2")
        WriteLine("3")
        WriteLine("4")
        WriteLine("5")
    End Sub
End Class
```

If your code would be similar to that above, how would you make your program write 1 to 100 or even 1 to 10,000? Let us agree that the previous example is not very efficient! So instead, amend your code so that it is the same as the code below. Once the code has been typed up, compile and run the program.

```
Imports System.Console
Public Class UsingIteration
    Shared Sub Main()
        For X = 1 To 5
            WriteLine(X)
        Next X
    End Sub
End Class
```

Program Output:

Congratulations – you just used a 'for loop' to reduce the amount of code written! Most software languages have some form of 'for loop'; remember, the principles are the same – only the syntax changes. 'For loops' are helpful if the programmer is aware of the number of times they need to execute the same code. A 'for loop' has a starting number and an ending number/condition (that marks the end of the loop); this example will loop from 1 to 5 (in total five times). A 'for loop' also has an increment value; by default this is set to 1. In this example, the integer variable X will initially have the numeric value 1; once the loop executes its containing code once, X will then increment to 2 and will repeat this process until the end condition is met (in this case when X reaches the value 5).

While Loop

Another useful loop is the 'Do While' loop; again, this type of loop is found in most software languages. This loop works by repeating the contained code while a condition is true; the moment the condition becomes false, the loop is ended. The logic of a 'While Loop' is shown below:

```
Do While [this condition is true]
   Repeat this code
Loop
```

'Do While' loops are helpful if the programmer is unsure how many times the code will need to be repeated, as a condition can be used instead of a fixed numeric value. An example of a 'Do While' loop is shown below; try it yourself!

```
Imports System.Console
Public Class UsingIteration
    Shared Sub Main()
        Dim x As Integer = 1
        Do While x < 5
            WriteLine(x)
            x = x + 1
        Loop
    End Sub
End Class
```

This example has a condition that states the variable x must be less than 5. As x has the value 1 to begin with, the given condition is true and thus the code will execute. The second line of code inside the loop (after the **WriteLine** instruction) will increment the value of x by 1, before the loop returns to the start and checks the condition again. This loop will continue until x has the value of 5, because at this point the condition becomes false (if x equals 5 then this is no longer less than 5) and the code ceases to execute. This example above will only loop four times; if the programmer wanted it to run five times then the condition needs to be amended to '**Do While x<=5**'.

Program Output:

Test Your Skills

- ✓ *Replace the keyword **While** with **Until** and replace the less than symbol (<) with the more than symbol (>). What is the difference between **While** and **Until**?*