Chapter 4: Operators

Section 4.1: String Concatenation

String concatenation is when you combine two or more strings into a single string variable.

String concatenation is performed with the `&` symbol.

```
Dim one As String = "Hello "
Dim two As String = "there"
Dim result As String = one & two
```

Non-string values will be converted to string when using `&`.

```
Dim result as String = "2" & 10 ' result  = "210"
```

Always use `&` (ampersand) to perform string concatenation.

DON'T DO THIS
While it is possible, in the *simplest* of cases, to use the `+` symbol to do string concatenation, you should never do this. If one side of the plus symbol is not a string, when Option strict is off, the behavior becomes non-intuitive, when Option strict is on it will produce a compiler error. Consider:

```
Dim value = "2" + 10     ' result = 12  (data type Double)
Dim value = "2" + "10"   ' result = "210"  (data type String)
Dim value = "2g" + 10    ' runtime error
```

The problem here is that if the `+` operator sees any operand that is a numeric type, it will presume that the programmer wanted to perform an arithmetic operation and attempt to cast the other operand to the equivalent numeric type. In cases where the other operand is a string that contains a number (for example, "10"), the string is *converted to a number* and then *arithmetically* added to the other operand. If the other operand cannot be converted to a number (for example, "2g"), the operation will crash due to a data conversion error. The `+` operator will only perform string concatenation if *both* operands are of `String` type.

The `&` operator, however, is designed for string concatenation and will cast non-string types to strings.

Section 4.2: Math

If you have the following variables

```
Dim leftValue As Integer = 5
Dim rightValue As Integer = 2
Dim value As Integer = 0
```

Addition Performed by the plus sign `+`.

```
value  = leftValue + rightValue

'Output the following:
'7
```

Subtraction Performed by the minus sign `-`.

```
value = leftValue - rightValue
```

```
'Output the following:
'3
```

Multiplication Performed by the star symbol `*`.

```
value = leftValue * rightValue

'Output the following:
'10
```

Division Performed by the forward slash symbol `/`.

```
value = leftValue / rightValue

'Output the following:
'2.5
```

Integer Division Performed by the backslash symbol `\`.

```
value = leftValue \ rightValue

'Output the following:
'2
```

Modulus Performed by the `Mod` keyword.

```
value = leftValue Mod rightValue

'Output the following:
'1
```

Raise to a Power of Performed by the `^` symbol.

```
value = leftValue ^ rightValue

'Output the following:
'25
```

Section 4.3: Assignment

There is a single assignment operator in VB.

- The equal sign `=` is used both for equality comparison and assignment.
  `Dim value = 5`

Notes
Watch out for assignment vs. equality comparison.

```
Dim result = leftValue = rightValue
```

In this example you can see the equal sign being used as both a comparison operator and an assignment operator, unlike other languages. In this case, `result` will be of type `Boolean` and will contain the value of the equality comparison between `leftValue` and `rightValue`.

Related: Using Option Strict On to declare variables properly

Section 4.4: Comparison

Comparison operators compare two values and return to you a boolean (`True` or `False`) as the result.

Equality

- The equal sign `=` is used both for equality comparison and assignment.
  `If leftValue = rightValue Then ...`

Inequality

- The left angle bracket nest to the right angle bracket `<>` performs an unequal comparison.
  `If leftValue <> rightValue Then ...`

Greater Than

- The left angle bracket `<` performs a greater than comparison.
  `If leftValue < rightValue Then ...`

Greater Than Or Equal

- The equal sign nest to the left angle bracket `=>` performs a greater than or equals comparison.
  `If leftValue =< rightValue Then ...`

Less Than

- The right angle bracket `>` performs a less than comparison.
  `If leftValue > rightValue Then ...`

Less Than Or Equal

- The equal sign nest to the right angle bracket `=>` performs a greater than or equals comparison.
  `If leftValue => rightValue Then ...`

Like

- The `Like` operator tests the equality of a string and a search pattern.
- The `Like` operator relies on the Option Compare Statement
- The following table lists the available patterns. Source: https://msdn.microsoft.com/en-us/library/swf8kaxw.aspx (Remarks section)

Characters in the *Pattern*	Matches in the *String*
?	Any single character
*	Zero or more characters
#	Any single digit (0 - 9)
[charlist]	Any single character in *charlist*
[!charlist]	Any single character not in *charlist*

- See further info on MSDN in the remarks section.
  `If string Like pattern Then ...`

Section 4.5: Bitwise

These are the bitwise operators in VB.NET : And, Or, Xor, Not

Example of And bitwise operation

```
Dim a as Integer
a = 3 And 5
```

The value of a will be 1. The result is obtained after comparing 3 and 5 in binary for. 3 in binary form is 011 and 5 in binary form is 101. The And operator places 1 if both bits are 1. If any of the bits are 0 then the value will be 0

```
3 And 5 will be  011
                 101
                 ---
                 001
```

So the binary result is 001 and when that is converted to decimal, the answer will be 1.

Or operator places 1 if both or one bit is 1

```
   3 Or 5 will be 011
                  101
                  ---
                  111
```

Xor operator places 1 if only one of the bit is 1 (not both)

```
   3 Xor 5 will be  011
                    101
                    ---
                    110
```

Not operator reverts the bits including sign

```
Not 5 will be - 010
```