TABLE OF CONTENTS

LICENSE

LANGUAGE BASICS

Introduction

Variables
  Syntax

Arrays
  Syntax
  Example

Strings
  Concatenation
  String.Format
  New Line

CONDITIONAL STATEMENTS

If Statement
  Syntax
  Example

If Else Statement
  Example
  Operators

Select Case
  Syntax
  Example

LOOPS

While Loop
  Syntax
  Example

For Loops
  Syntax
  Example

For Each
  Syntax
  Example
ADVANCED – EXCEPTIONS, METHODS, CLASSES

Exceptions
Syntax
Example

Methods
Syntax
Example

Functions
Syntax
Example

Classes
Syntax
Example

ENJOYED IT?

Why not give us a like?
LICENSE

This work is licensed under the creative commons Attribution-NonCommercial-NoDerivs 3.0 Unported

- You may not alter, transform, or build upon this work.
- You may not use this work for commercial purposes.
- You are free to copy, distribute and transmit the work
**LANGUAGE BASICS**

**INTRODUCTION**

Visual Basic has a simple syntax much of the language is easily understandable (that's why it's called Basic, doh). A few points:

- The language is not case-sensitive (So A and a are the same)
- Lines do not terminate with semi-colons
- Code is in code blocks, but not your standard Java or C# code block {} (You will see in the examples)

**VARIABLES**

Variables are declared using the Dim keyword, Dim is short for (Dimension).

**SYNTAX**

```
Dim MyVariable As DataType
```

The above code creates a variable called MyVariable with no value. The example below creates two variables with data type of string and one of type integer I will use these variables throughout.

```
Dim Name As String = "thecodingguys"
Dim Year As Integer = 2013
```

**ARRAYS**

Arrays are similar to variables, however arrays can hold more than one value.

**SYNTAX**

```
Dim MyArray() As DataType = {Values Comma Separated}
```

**EXAMPLE**

```
Dim MyGamesOf2013() As String = {"GTAV", "Battlefield 3"}
Dim MyMoviesOf2013() As String = New String(3) {"The Amazing Spiderman", "The Expendables", "X-Men", "Rise of the planet of the apes"}
```

**STRINGS**

**CONCATENATION**

Concatenation is done through the & symbol, like the following:

```
Console.WriteLine("Hello " & "World")
```
**STRING FORMAT**

Formats a string, the following example prints out £5,00

```csharp
Console.WriteLine(String.Format("{0:C}", 5))
```

In the example above, we want to format the number 5 and show the currency symbol. The `{0:C}` is the formatting we want to do, in this case it means format the first argument (0) and apply the currency symbol. Many more formatting types are available see this [MSDN reference](https://docs.microsoft.com).

The formatting depends on your computer's regional settings, users of the UK will see the £ symbol, users of USA will see the $ symbol and so on.

**NEW LINE**

New lines are made using the `vbCrLf` word.

```csharp
Console.WriteLine("Hello " & vbCrLf & "World")
```
CONDITIONAL STATEMENTS

**IF STATEMENT**

Executes code based on a condition, the condition must evaluate true for the code to execute.

**SYNTAX**

If True Then
End If

**EXAMPLE**

If Year > 2010 Then
    Console.WriteLine("Hello World!")
End If

**IF ELSE STATEMENT**

The If Else Statement works similar to the if statement, however if the first condition is false the else condition will execute.

**EXAMPLE**

If Year < 2010 Then
    Console.WriteLine("Hello World!")
Else
    Console.WriteLine("Hello!")
End If

**OPERATORS**

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>Less than operator</td>
<td>if 19 &lt; 20 Then</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than operator</td>
<td>if 20 &gt; 19 Then</td>
</tr>
<tr>
<td>=</td>
<td>Equal to operator</td>
<td>if a = b Then</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Not equal to operator</td>
<td>if a &lt;&gt; b Then</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to operator</td>
<td>if 19 &lt;= b Then</td>
</tr>
<tr>
<td>=&gt;</td>
<td>Greater than or equal to operator</td>
<td>if 19 &gt;= b Then</td>
</tr>
</tbody>
</table>
**SELECT CASE**

The Select Case statement is similar to a switch statement found in many other programming languages. A few points:

- Select Case evaluate one variable
- You can use some operators
- Select Case Statements are must easier to maintain then using nested if else

**SYNTAX**

```csharp
Select Case variableName
    Case 1
    Case 2
    Case Else
End Select
```

**EXAMPLE**

```csharp
Select Case Year
    Case 2012
        Console.WriteLine("It's 2012!")
    Case 2013
        Console.WriteLine("The current year!")
    Case Year > DateTime.Now.Year
        Console.WriteLine("Year is greater than 2013")
    Case Else
        Console.WriteLine("....")
End Select
```
**Loops**

**While Loop**
Continuously loops around code until the condition becomes false.

**Syntax**
```
While True
    End While
```

**Example**
```
While Year >= 2013
    Year += 1
    If Not Year = 2100 Then
        Console.WriteLine(Year)
    Else
        Exit While
    End If
End While
```
Visual Basic does not have an increment operator, however Year += 1 will increment by 1 until it reaches 2100. *Always make sure your loop comes to a stop at some point otherwise it becomes endless and can result in errors.*

**For Loops**
Similar to the while statement, but you specify when the loop should end.

**Syntax**
```
For index = 1 To 10
    Next
```

**Example**
```
For i = 1 To 100
    Console.WriteLine(i)
Next
```
This will output 1 – 100. Once it reaches 100 it will stop.
**FOR EACH**

Loops through elements in a collection.

**SYNTAX**

```
For Each element As DataType In Group
    Next
```

**EXAMPLE**

```
For Each item As String In MyGamesOf2013
    Console.WriteLine(item)
    Next
```

The above example prints out all the elements in the MyGamesOf2013 array created earlier.
Advanced – Exceptions, Methods, Classes

Exceptions
To catch possible exceptions which may occur we use a Try Catch Block.

Syntax
Try
    Catch ex As Exception
    End Try

Example
Try
    Console.WriteLine(Year + 2147483641)
    Catch oEx As OverflowException
        Console.WriteLine("Result: Overflow: " + oEx.Message)
    Catch fEx As ArithmeticException
        Console.WriteLine("Result: Arithmetic Exception: " + fEx.Message)
    Catch ex As Exception
        Console.WriteLine(ex.Message)
    End Try

The example above catches Overflow Exception, Arithmetic Exception or if all fails it will catch any exception. The above however only results in an Overflow Exception.

Methods

Syntax
Public Sub MyMethodName()
    ‘No Arguments
End Sub

Public Sub MyMethodName(ByVal Parameter As DataType)
    ‘Arguments
End Sub

A Sub (Subroutine) does not return value back. A Public method can be accessed outside the current class, if you declare it private it can only be accessed within the current class.
EXAMPLE

```vbnet
Public Sub WelcomeUser()
    Console.WriteLine("Welcome Guest")
End Sub

Public Sub WelcomeUser(ByVal Name As String)
    Console.WriteLine("Welcome " + Name)
End Sub
```

The above example is also an overloaded method this is where two methods have the same name but you pass different arguments of different data type. The first method has no arguments and the second one requires an argument.

(Just be careful not to make it ambiguous, for example having a method where one you require a double and other and Integer, double can be an integer so this results in an error)

FUNCTIONS

A function similar to a Subroutine, requires that you return some data back, you can pass arguments just like above.

SYNTAX

```vbnet
Public Function MyFunc() As DataType
    Return Data
End Function
```

EXAMPLE

```vbnet
Public Function Tomorrow() As Date
    Dim CurrentDate As Date = DateTime.Now
    Return CurrentDate.AddDays(1)
End Function
```
**CLASSES**

A class contains methods and events.

**SYNTAX**

```csharp
Public Class MyClassName
    'Your Methods
End Class
```

**EXAMPLE**

```csharp
Public Class MyCar

    Public Sub CarManufacturer(ByVal Manufacturer As String)
        Console.WriteLine(Manufacturer)
    End Sub

End Class
```

We can then call this method from another class or module, first you must initialize the class.

```csharp
Dim Car As New MyCar()

Car.CarManufacturer("Audi")
```

This will output Audi to the console.
ENJOYED IT?

WHY NOT GIVE US A LIKE?