What is Programming?

What is Programming?

Writing software (computer programs) is a lot like writing down the steps it takes to complete a process. These written steps, often referred to as instructions, are passed to a computer which then processes each command in sequence. Often, in computing, these instructions manipulate things called objects; objects may be numbers, words and/or graphical interfaces.

Computer Rules

Always remember that a computer is dumb, but very obedient. In other words, a computer will do exactly what the user tells it to do… even if it is not necessarily what the user intended it to do. Below are three rules to consider when programming:

1. Computers do not make mistakes – programmers do
2. Programming will highlight the importance of 'clarity of expression'
3. Programming instructions are processed in sequence and one at a time

High-Level Programming Languages

Most computer programming today is achieved using high-level programming languages. There are lots of these languages available on the market and some are quite old, i.e. COBOL which was devised in the 1950s! More modern languages include Java, VB.NET, Python, C#, JavaScript, PHP, etc.

A high-level language basically makes it easier to write programs by allowing advanced programs to be written without any concerns in regard to computer architecture – i.e. specific CPU instructions. These languages also come with pre-written, reusable common code – known as libraries – which help to reduce development times. An example of a high-level language is shown below:

Code:	**Explanation:**

If x >= 5 Then
WriteLine("Hello World")
End IF
	If x is greater than or equal to 5, then write the line 'Hello World' to the console window.

All you need to remember about high-level programming languages (including VB.NET) is that:

- *The syntax is sort of like English*
- *They have pre-written code called libraries*
- *They 'sit on top' of an operating system*
- *They are **not** languages that a CPU understands*

Assembler Languages

Assembler language is 'one step above' a computer's native language, machine language (binary). In an assembler language, instructions are given human-friendly symbolic names. Unlike high-level languages, the programmer works with basic operations/instructions that the CPU can directly perform, such as bitwise operations (i.e. AND, OR, NOT). Remember that assembler language is very basic and therefore is impractical when writing large programs (which are normally achieved using high-level languages). Below is an example of assembler language (notice it is not as interpretable as the high-level language):

Code:	**Explanation:**
MOV EAX, [EBX]	Move the 4 bytes in memory at the address contained in EBX into EAX
MOV [ESI+EAX], CL	Move the contents of CL into the byte at address ESI+EAX

Machine Language

Computers only understand 'bits' – 0s and 1s, often referred to as binary; binary is machine language. A computer system uses these bits to represent information, whether that is numbers, characters, pixels, etc. The computer also uses bits to represent computer instructions; this is very difficult to do, although the pioneers of computer science once did this! However, today most programs are written in high-level languages which are then compiled into machine code using a compiler.

Compilers

As stated earlier, computers do not understand programs written in high-level languages, such as VB.NET and Java (they only understand binary, 0s and 1s). High-level languages must be compiled using a compiler to convert the source code into machine code. Every high-level programming language has a specific compiler; a compiler is a small computer utility program usually packaged with the SDK (Software Development Kit) of the programming language.