**Lesson 1: The Modern Programming Language.**

The native language of the modern computer is ***machine* *code***. The CPU of the computer receives and sends instructions in the form of a series of binary codes. Each code represents a certain simple instruction such as to store a number in a memory cell, to retrieve a number from a memory cell or to add two numbers. The code might look something like this:

1101 0111 1001 0000 ‘retrieve a number from cell #9

1101 0111 1010 0000 ‘retrieve a number from cell #10

1100 0001 1011 0000 ‘add them and put the answer in cell #11

etc.

An application such as a word processor or a game would require many thousands or millions of these binary instructions. It would be impossible to manually write most applications today in machine code for the following reasons:

* Entering millions of 1s and 0s would take an unreasonably long time
* Errors would be common and finding errors would be extremely difficult
* Every computer platform is different, therefore code must be written specifically for the hardware in the system.

Various programming languages have been developed to make programming easier. One of the earliest languages, only a small step better than machine code is ***assembler*** language. Assembler is a collection of three-letter codes to represent each binary code. An assembler version of the above machine code program would be something like this:

MOV ax, 09

MOV bx, 0A

ADD cx

MOV cx, 0B

Assembler is a little easier to read and write than binary, but it is still a direct translation of machine code, meaning that doing something as simple as adding two numbers together takes several lines of code. As well, any assembler program must be written for a specific platform with certain hardware components. Both assembler and machine code are considered **low level languages**.

It was not long until programmers created **high level languages** that were closer to English than machine code. Examples of high level languages are BASIC, Perl, Python, C++ and Java. In these languages, to add two numbers and store them in a variable, the code would be this:

x = 5 + 7

A program written in a high level language is translated into machine code by a program called a ***compiler or interpreter***. This has several advantages:

1. A simple command can replace many lines of machine code.
2. The programmer doesn’t have to worry about writing specifically for a particular **platform** (the MacBook Air using OSX, for example). Each platform has its own compiler, which will create the binary code specific to that platform.
3. A program, called the source code, once written, can be compiled for many different systems. This is called the ***portability*** of the code.

There are two different ways that the machine code is created. In one way, an ***executable* *file*** is **compiled**. This file contains the machine code that is read and executed by the CPU. On the Windows platform, the executable file is given the .exe extension. On a Mac it is given the .app or .pkg extension. The most common example of a language that is compiled this way is C++.

In a second method, an executable is not created. Instead, the code is interpreted at runtime using an ***interpreter***. The computer that is used to run the application must have a copy of the interpreter. Examples of interpreted languages are BASIC, Python and Java.

One advantage of a compiled executable is that it will run on its own. You don’t need an interpreter to run it.

One advantage of an interpreted code is that it can be checked for malicious code before it is executed. It is safer to run this way. Java, for example, is an interpreted language that was created for use through the internet. When you run a Java program on a webpage, your computer must have a Java interpreter installed (called the JVM for Java Virtual Machine). This interpreter makes sure you have all the permissions in order and will deny any malicious code from accessing your computer’s hard drive or other components. Before Java became the standard, people had to download executables and run them with the hope that they did not contain any malicious code.

In addition, Python is considered a ***scripting*** ***language***. Scripting languages are designed to create short but powerful programs (called scripts) that are usually embedded inside of larger applications.

Key words: **machine code, assembler, low-level language, high-level language, compiler, platform, portability, executable file, interpreter, scripting language.**