Compiler and Interpreter

Compiler and Interpreter are two different ways to execute a program written in a programming or scripting language.

A **compiler** takes entire program and converts it into object code which is typically stored in a file. The object code is also refereed as binary code and can be directly executed by the machine after linking. Examples of compiled programming languages are C and C++.

An **Interpreter** directly executes instructions written in a programming or scripting language without previously converting them to an object code or machine code. Examples of interpreted languages are Perl, Python and Matlab.

Interpreter

- Translates program one statement at a time.
- It takes less amount of time to analyze the source code but the overall execution time is slower.
- No intermediate object code is generated, hence are memory efficient.

- Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.
- Programming language like Python, Ruby use interpreters.

Compiler

- Scans the entire program and translates it as a whole into machine code.
- It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
- Generates intermediate object code which further requires linking, hence requires more memory.
- It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
- Programming language like C, C++ use compilers.

Machine level Programming

Imagine them as the “native tongue” of the computer, the language closest to the hardware itself. Each unique computer has a unique machine language. A machine language program is made up of a series of binary patterns (e.g., 01011100) which represent simple operations that can be accomplished by the computer (e.g., add two operands, move data to a memory location). Machine language programs are executable, meaning that they can be run directly. Programming in machine language requires memorization of the binary codes and can be difficult for the human programmer.

Assembly level Programming

They represent an effort to make programming easier for the human. The machine language instructions are replaced with simple pneumonic abbreviations (e.g., ADD, MOV). Thus assembly languages are unique to a specific computer (machine). Prior to execution, an assembly language program requires translation to machine language. This translation is accomplished by a computer program known as an Assembler. Assemblers are written for each unique machine language.

High level Programming

High-level languages, like C,C++, JAVA etc., are more English-like and, therefore, make it easier for programmers to “think” in the programming language. High-level languages also require translation to machine language before execution. This translation is accomplished by either a compiler or an interpreter. Compilers translate the entire source code program before execution. Interpreters translate source code programs one line at a time. Interpreters are more interactive than compilers.