Difference Between High-Level and Low-Level Languages

The major difference between High-Level and Low-Level languages is that the high-level language runs at a lower speed and will require a translator program. In contrast, no such requirement is for the low-level language.

Key Differences Between High-Level and Low-Level Languages with Example

High-Level Languages	Low-Level Languages
High-Level Language is a human-friendly language that is easy to learn and understand.	Low-Level Languages are quite challenging for humans to learn and understand.
They are executed at a slower speed as they require a translator program.	This language can execute at high speed.
These languages allow much more abstraction.	These languages have negligible abstraction.
These languages have a very low memory efficiency and consume more memory	These languages have a very high memory efficiency and consume less energy.
These languages are unable to provide many facilities at the hardware level.	These languages are machine friendly and very close to the hardware so it helps to write a program at the hardware level.
No need for hardware knowledge for writing programs.	Hardware knowledge is necessary for writing programs.
In these languages, modification of programs is easy for humans.	In this language, modification of programs is quite difficult for humans.
A single statement in this language may execute several instructions.	The statements in these languages can be directly redirected to processor instructions.
Programming, these languages are very common	For programming, these languages

and widely used nowadays

Example: Java, JavaScript, Python, PHP, C#, Objective C, C++, Cobol, Pascal, Perl, FORTRAN, LISP, and Swift programming languages are examples of High-Level Languages.

are not very common nowadays.

Example: Machine language and Assembly language are Low-Level Languages.

High-level Language vs Low-level Language

High-level language and Low-level language are the types of programming languages. It is the set of instructions for the Central Processing Unit (CPU) to perform any specific task. High-level language and Low-level language are different from each other in the following ways:

High-Level Language

- It can easily run on different platforms.
- It consumes more memory.
- It has low efficiency of memory.
- It is easy to maintain.
- It is simple and manageable.

Low-Level Language

- It runs on various platforms.
- Machines easily understand it.
- Maintenance is complex.
- It does not have a wide application at present.
- Debugging is difficult.

What are High-Level Languages?

Before knowing about the difference between high-level and low-level languages, let us explore high-level language. A high-level language is easy to read, write, and maintain by the human or programmer. These High-level languages are less memory efficient, and these languages are simple to debug. The High-level programming language (HLL) is used for developing user-friendly software programs and websites. A high-level language is further divided into three parts first one is Procedural Oriented Programming (POP) language, the second one is Object-Oriented Programming (OOP) language, and the third one is Natural language or human language.

Examples of High-level programming languages are Java, JavaScript, Python, PHP, C#, Objective C, C++, Cobol, Pascal, Perl, FORTRAN, LISP, and Swift programming language.

What are Low-Level Languages?

Low-Level Languages are machine-friendly languages that are quite difficult to understand by human beings but very easy to interpret by machines. The Low-level language is a machine-dependent programming language. The processor runs the low-level programs directly without the compiler or interpreter, so the programs written in low-level language can be run very fast. This language is difficult to write, understand and debug due to this person moving toward high-level language.

A low-level language is further divided into two parts, the first one is Machine Language, and the second one is Assembly language (ASM). It is a series of bits of 0s and 1s, or it performs short instructions like ADD, MOV, etc.