Aspect	High-Level Languages	Low-Level Languages
Abstraction	High-level of abstraction from machine hardware	Close to the hardware, minimal abstraction
Human Readability	Easier to read and write, uses natural language elements (e.g., `if`, `else`, `for`)	Harder to read, uses symbols and mnemonics (e.g., `MOV`, `ADD`)
Memory Management	Automatic (e.g., garbage collection in Java)	Manual memory management (e.g., pointers in C)
Portability	Platform-independent (e.g., Java, Python)	Platform-dependent (e.g., Assembly, machine code)
Execution Speed	Generally slower due to abstraction and interpretation/compilation overhead	Fast execution since it's closer to machine code
Error Handling	Built-in exception handling mechanisms	Less built-in error handling, errors must be managed by the programmer
Development Time	Faster development due to simpler syntax and rich libraries	Slower development due to complexity and need for manual control
Example Languages	Java, Python, C#, Ruby	Assembly, Machine code (binary), C (considered low-level compared to modern languages)
Machine Dependency	No (Can run on different platforms with minimal changes)	Yes (Machine-specific, needs specific hardware instructions)
Control Over Hardware	Limited direct control over hardware resources	Full control over hardware and system resources
Translation Process	Compiled or interpreted into machine code via compiler/interpreter	Directly translated into machine code or written in machine code itself