Comparison Tables

Low-Level vs. High-Level Languages

Feature	Low-Level Language	High-Level Language
Abstraction	Minimal	High
Readability	Hard to read	Easy to read
Performance	Very fast	Slower than low-level
Portability	Low	High
Examples	Assembly, Machine Code	Python, Java, C++

Interpreted vs. Compiled Languages

Feature	Interpreted Language	Compiled Language
Execution	Line by line	Full compilation before execution
Speed	Slower	Faster
Error Detection	During execution	Before execution
Portability	High	Platform-dependent
Examples	Python, JavaScript	C, C++

Programming vs. Scripting Languages

Feature	Programming Language	Scripting Language
Main Purpose	General-purpose apps	Automation, web scripts
Compilation	Often compiled	Often interpreted
Performance	Faster	Slower
Usage	Complex software, OS	Web development, automation
Examples	C, Java, C++	Python, JavaScript, Bash

Open Source vs. Not Open Source

Feature	Open Source Software	Not Open Source Software

Code Availability | Publicly available | Restricted access

Modification | Can be modified | Cannot be modified

Cost | Often free | Usually paid or licensed

Security | Can be reviewed | Security depends on vendor

Examples | Linux, Python, MySQL | Windows, MATLAB, Oracle

Support OOP vs. Not Supporting OOP

Feature | Supports OOP | Does Not Support OOP

---|---|---

OOP Principles | Yes | No

Code Reusability | High | Low

Modularity | Strong | Weak

Examples | Java, Python, C++ | C, Assembly, Bash

Generated by ChatGPT