# Types of Programming languages

## Classification by Level: Low-Level and High-Level

| Classification by Level | Low-Level | High-Level |
|---|---|---|
| **Definition** | Low-level programming languages are close to machine language (Machine Language) and interact directly with hardware. These languages require a deep understanding of the internal architecture of the computer. | High-level programming languages are far removed from machine language and use human-readable instructions. They are converted into machine language using programs like compilers or interpreters. |
| **Types and Examples** | Machine and Assembly Language | Python, C++ and JavaScript. |
| **Advantages** | 1. High efficiency in resource usage (e.g., memory and processor).<br>2. Full control over hardware. | 1. Easy to write and understand.<br>2. Portable across different systems.<br>3. Built-in tools simplify development. |
| **Disadvantages** | 1. Difficult to write and understand.<br>2. Not portable across different systems. | 1. Less efficient than low-level languages.<br>2. Depending on intermediate programs to convert code into machine language. |

## Classification by Execution Method: Compiled Languages and Interpreted Languages

| Classification by Execution Method | Compiled Languages | Interpreted Languages |
|---|---|---|
| **Definition** | In these languages, the entire source code is converted into machine code before execution using a program called a "compiler." | In these languages, the source code is executed directly without prior compilation. This is done using a program called an "interpreter." |
| **Examples** | C, C++ and Go | Python, JavaScript and PHP. |
| **Advantages** | 1. High performance efficiency as the final code is optimized.<br>2. Faster execution since no recompilation is needed at runtime. | 1. Easy testing and rapid development.<br>2. Highly portable across different systems. |
| **Disadvantages** | 1. Long recompilation time when changes are made.<br>2. May not be portable across different systems. | 1. Lower performance compared to compiled languages.<br>2. Requires the interpreter to be present on the target machine. |

## Classification by Type of Programming: Programming Languages and Scripting Languages

| Classification by Type of Programming | Programming Languages | Scripting Languages |
|---|---|---|
| **Definition** | These languages are used to create standalone, executable programs. | These languages are used to write scripts that run within other programs or environments, such as web browsers or server management systems. |
| **Examples** | C, C++, Java, Python. | JavaScript, PHP, Ruby. |
| **Advantages** | 1. Provide full control over software.<br>2. Used to build complex applications. | 1. Easy to write and execute.<br>2. Used for automating simple tasks. |

## Classification by Licensing: Open-Source Languages and Not Open-Source Languages

| Classification by Licensing | Open-Source Languages | Not Open-Source Languages |
|---|---|---|
| **Definition** | Open-source programming languages allow free access to their source code, enabling modification and redistribution. | Non-open-source programming languages are controlled by a specific company or entity, and their source code is inaccessible. |
| **Examples** | Python, Java, PHP, Ruby. | some versions of Visual Basic and MATLAB |
| **Advantages** | 1. A large community contributes to the language's development.<br>2. Wide customization options. | **I did not find** |
| **Disadvantages** | **I did not find** | 1. Limited customization.<br>2. May require paid licenses. |

## Classification by Support for Object-Oriented Programming (OOP):

| | Languages Supporting OOP | Languages Not Supporting OOP |
|---|---|---|
| **Definition** | These languages allow programmers to use object-oriented programming concepts like classes and objects to provide better organization of programs. | These languages rely on other programming paradigms, such as procedural programming. |
| **Examples** | Java, Python, C++, Ruby. | C, Assembly. |
| **Advantages** | 1. Better organization of data and instructions.<br>2. Code reuse. | **I did not find** |
| **Disadvantages** | **I did not find** | 1. Difficulty in handling large projects.<br>2. Lack of organization. |