Introduction Part 2:

Programming languages are tools that allow humans to communicate instructions to computers, which only understand binary (0s and 1s). These languages are categorized into low-level, middle-level, and high-level languages, each with different levels of hardware interaction and readability. Understanding these types helps in choosing the right language for a specific task, from direct hardware control to building complex applications.

What is a Programming Language?

A programming language is a structured way to give instructions to a computer, following specific rules and syntax. Computers understand only binary code (0 and 1) because they operate using electric signals that represent on and off states.

Types of Programming Languages

Programming languages can be categorized in multiple ways, but the main distinction is by their level of abstraction: low-level, middle-level, and high-level languages. This classification depends on how closely the language interacts with computer hardware and how easy it is for humans to read and write.

Low-Level Programming Languages

Low-level languages are machine-friendly and closely tied to hardware, often requiring different code for different processors. They are hard for humans to read and write but are directly understood by computers. The main low-level languages are machine language (pure binary) and assembly language.

Machine Language: The First Programming Language

Machine language uses only 0s and 1s and is the earliest form of programming. It is extremely fast and directly understood by computers, but very hard for humans to work with. Ada Lovelace is recognized as the first programmer, creating the first machine algorithm in the 1800s.

Assembly Language and Mnemonics

Assembly language was developed to make programming easier by introducing human-readable commands, called mnemonics (like ADD or MOVE), instead of pure binary. However, it still remained machine-dependent and required separate codes for different hardware.

The Role of Assemblers

Since computers still only understand binary, a tool called an assembler translates assembly language into machine language. This makes it possible for humans to write somewhat readable code that computers can execute.

The Rise of High-Level Languages

As programming needs grew, high-level languages like Algol, Fortran, LISP, BASIC, and eventually C were developed. These languages are more human-friendly, less dependent on hardware, and can be used for a wide range of applications.

The Impact of C Language

C became the first widely used general-purpose programming language in 1972, allowing developers to create many types of applications (games, business, scientific) with one language. This was a significant shift from earlier languages, which were often designed for specific tasks.

What Makes a High-Level Language?

High-level languages offer features that make software development easier, such as operators, structures, classes, and exception handling. They use English-like statements, making them easier to learn and use, but they are not suitable for direct hardware interaction.

Limitations of High-Level Languages

While high-level languages are easy for humans, they cannot interact directly with hardware or perform low-level tasks efficiently. For specialized hardware control or high-speed requirements, low-level languages are still used in areas like research centers and embedded systems.

Middle-Level Languages: C and C++

Middle-level languages like C and C++ combine features of both low-level and high-level languages. They allow for both hardware interaction and easy software development, making them suitable for tasks like writing device drivers or embedded systems.

Summary of Language Levels

Low-level languages (machine and assembly) are hardware-focused; high-level languages (Java, Python, etc.) are user-focused; and middle-level languages (C, C++) balance both. Choosing the right type depends on the task and required hardware interaction.