language

Language is a special system of communication that people use to express their thoughts, feelings, ideas, and information to others. It uses words, sounds, signs, or symbols in an organized way so that people can understand each other.

Key Features of Language

1. **Communication tool** – It helps people talk, write, or signal to share messages.
2. **Uses rules (grammar)** – Every language follows rules about how to form words and sentences.
3. **Made of symbols** – These symbols can be spoken sounds, written letters, or even hand signs (like in sign language).
4. **Changes over time** – Language grows and changes as society changes.
5. **Learned, not born with** – We learn language from our family and society.

Languages required for in-between

1. Human to human communication - Hindi, English, etc.
2. Human to machine communication – programming languages(c, c++, java, python, php, etc)

What is a Programming Language?

A programming language is a syntex or formate for giving instructions to a computer so it can perform specific tasks. Just like humans use natural languages to communicate, developers use programming languages to communicate with machines.

Why Do We Need Programming Languages?

Computers understand only binary (0s and 1s). It is very difficult to write code in binary formats. Programming languages provide an easier way to write instructions in human-readable formats, which are then translated into machine code that computers can understand.

Classification of Programming Languages

Programming languages can be grouped in various ways depending on their abstraction level, execution style, application area, and programming approach. Here's a comprehensive breakdown with examples and explanations.

Classification by Abstraction Level

Low-Level Languages

These languages operate close to the hardware and offer minimal abstraction. They’re faster but harder for humans to read or write.

Machine Language

1. Composed entirely of binary digits (0s and 1s).
2. Understood and executed directly by the CPU.
3. Hardware-specific and requires no translation.
4. Not user-friendly for programmers.

Assembly Language

1. Uses symbolic instructions or mnemonics like MOV, ADD, etc.
2. Slightly more readable than machine code but still platform-specific.
3. Requires an assembler to translate to binary.
   Example:

```
MOV AX, 5
ADD AX, 3
```

High-Level Languages
These are designed for ease of use, focusing more on solving problems than managing hardware.

1. Human-readable syntax.
2. Portable across different systems.
3. Needs a compiler or interpreter for execution.

Examples: Python, Java, C++, JavaScript, Ruby

Classification by Execution Method

Compiled Languages

1. Entire program is converted into machine code before running.
2. Offers fast execution speed.
3. Typically platform-dependent unless compiled for multiple platforms.

Examples: C, C++, Go, Rust

Interpreted Languages

1. Code is executed line by line by an interpreter.
2. Easier for debugging and development.
3. Generally slower but platform-independent.

Examples: Python, JavaScript, Ruby, PHP

Hybrid Languages (Bytecode + Virtual Machine)

1. Source code is first compiled to an intermediate bytecode.
2. This bytecode is then run by a virtual machine (e.g., JVM, CLR).
3. Combines the advantages of both compiled and interpreted languages.

Examples: Java (JVM), Python (PVM), .NET languages (CLR)

Classification by Programming Paradigm

Procedural Programming

1. Focuses on procedures or routines.
2. Programs are structured as a series of step-by-step instructions.

Examples: C, Pascal, Fortran

Object-Oriented Programming (OOP)

1. Centers around objects and classes.
2. Emphasizes reuse through features like inheritance, encapsulation, and polymorphism.

Examples: Java, C++, Python

Functional Programming

1. Based on pure functions and immutability.
2. Avoids side effects and mutable state.

Examples: Haskell, Lisp, Scala, F#

Logic Programming

Built on formal logic and declarative statements.
Defines what should happen rather than how.

Example: Prolog

Scripting Languages

Designed for automating tasks and writing short programs.
Typically interpreted.

Examples: Python, Bash, Perl, JavaScript

Classification by Domain of Use

System Programming Languages

Ideal for building operating systems, embedded systems, and drivers.

Examples: C, Rust, Assembly

Web Development Languages

1. Frontend: HTML, CSS, JavaScript
2. Backend: PHP, Python, Node.js, Ruby, Java

Scientific and Numerical Computing Languages

Used for complex calculations, simulations, and data processing.
Examples: MATLAB, R, Julia, Fortran

Artificial Intelligence & Machine Learning Languages

Offer powerful libraries and frameworks for AI/ML development.
Examples: Python, R, Julia