PROGRAMMING LANGUAGES

What are Programming Languages?

Before diving into what a Programming Language is, let's understand what Language means. Language is a way we communicate, sharing ideas and opinions with others.

Now, a programming language is like a special way we talk to computers. Computers only get the language of 0s and 1s. But we can't talk to them like that. So, programming languages are made to be understood by both people and computers. They have certain rules, and if we follow and learn these rules, we can tell the computer what to do.

Here's the trick: Computers don't really get our programming language directly. There's a middle step. We use something called a compiler to change our human-friendly language into a computer-friendly one. It's like a translator, turning what we say into something the computer can understand and do.

Some of the most popular programming languages are-

C, C++, Java, Python, JavaScript, Ruby and Go are some of the most popular programming languages.

Each programming language is used for specific purposes. Like Java is used to develop enterprise-level applications, mobile applications, web development etc. Python is used for web-development, AIML, Data analytics etc.

How to choose which programming language to learn-

If you're unsure about which career domain to pursue—whether it's Mobile Application Development, Cybersecurity, Software Development, Full-stack development, backend development, frontend development, Testing, Database, AIML (Artificial Intelligence and Machine Learning), or Data

Analytics. I recommend starting by learning the basics of one programming language in which you find it easier to grasp concepts quickly. Options like Java, Python, or C++ are good choices.

Learning the fundamentals of one programming language serves as a solid foundation. The advantage is that once you understand the basics, transitioning to another language becomes easier because many programming languages share similar syntax. This flexibility allows you to explore different domains without having to start from scratch every time you switch languages. It's like building a versatile skill set that can be applied across various areas of technology.

Types of Programming Languages

High-Level vs Low-Level Programming Languages

High-Level Programming Language-

A high-level programming language is a programming language that is designed to be easily understood and written by humans.

Code written in high-level languages is generally more readable and resembles human language, making it easier for programmers to understand and maintain.

Examples of high-level programming languages include Python, Java, C++, C#, Ruby, Swift, and many others.

Low-Level Programming Language-

Low-level language is machine-dependent (0s and 1s) programming language. The processor runs low- level programs directly without the need of a compiler or interpreter, so the programs written in low-level language can be run very fast. They require a deeper understanding of computer architecture and can be more time-consuming to write and maintain compared to high-level languages.

Applications:

- **High-level languages:** Ideal for web development, mobile apps, data science, AI, and general-purpose programming.
- **Low-level languages:** Used for writing device drivers, operating systems, embedded systems, and applications requiring precise hardware control.

Types of High- Level Programming languages-

Difference between Procedural-oriented and Object-oriented programming languages-

In **procedural programming languages**, the program is divided into procedures or functions, which are sets of instructions that perform a specific task. These procedures can be called or invoked as needed.

Limited Code Reusability: While procedures can be reused, procedural programming has limited support for features like encapsulation and abstraction, which are more prominent in object-oriented programming (OOP).

Examples of procedural-oriented programming languages include:

- C: C is a classic procedural programming language known for its efficiency and low-level control over hardware.
- Fortran: Fortran is a procedural language commonly used for scientific and engineering applications.
- COBOL: Common Business-Oriented Language (COBOL) is a procedural language designed for business, finance, and administrative systems.

Object-Oriented Programming (OOP) is a programming paradigm that revolves around the concept of "objects," which are instances of classes. It provides a way to structure and organize code based on the principles of encapsulation, inheritance, and polymorphism.

It is used to implement real-world entities like inheritance, polymorphism, abstraction, etc in the program to makes the program resusable, efficient, and easy-to-use.

The main advantage of object-oriented programming is that OOP is faster and easier to execute, maintain, modify, as well as debug.

NOTE: We will discuss in-depth classes and objects later in OOPS TOPIC.

What are frameworks and What are IDE?

In software development, a **framework** is a pre-built, reusable set of tools, libraries designed to assist in development of software applications.

Frameworks saves time and effort of user by handling common tasks.

Frameworks provide a structured way to handle incoming requests from users.

Frameworks easily manage database interactions, provides user authentication and authorization features.

Provides common design patterns like MVC pattern.

Provides better security.

Examples of Frameworks:

- **Web Frameworks:**
  - Django (Python) for web development.
  - Ruby on Rails (Ruby) for web development.
  - Express.js (JavaScript/Node.js) for server-side JavaScript.
- **Application Frameworks:**
  - Spring (Java) for enterprise applications.
  - .NET (C#) for Windows-based applications.
- **Front-End Frameworks:**
  - React.js, Angular, Vue.js for building interactive user interfaces.

- **Testing Frameworks:**
  - JUnit (Java) for unit testing.
  - pytest (Python) for testing.

IDE stands for Integrated Development Environment. It is a software application that provides comprehensive facilities to programmers for software development. An IDE typically includes a code editor, a compiler or interpreter, build automation tools, and debugging features, all integrated into a unified user interface. IDEs aim to streamline the software development process by providing a centralized environment for coding, testing, and debugging.

Some popular IDEs are-

1.Eclipse

2.Visual Studio Code (VSCode)

3.PyCharm

4.NetBeans

5.Android Studio

6.Atom

Hardware vs Software-

Hardware refers to the physical components of a computer system that can be touched and seen. It encompasses all the physical devices and circuits that make up the computer.

Hardware is physically tangible. It includes components such as the central processing unit (CPU), memory, storage devices, input/output devices, and more.

Examples of hardware include processors, memory modules, hard drives, graphics cards, monitors, keyboards, and printers.

Hardware is responsible for executing software.

A **software** is a set of programs or instructions that is designed to perform a specific task.

Software is not physically tangible. It consists of code, data, and instructions that are stored electronically and executed by hardware.

Software requires hardware to execute. It needs a computer, mobile device, or other hardware platforms to run and perform its functions.

Join our growing community of tech enthusiasts! Subscribe to our YouTube channel, where we simplify programming languages in the easiest way possible. Gain a deeper understanding with our clear explanations and receive exclusive notes to enhance your learning journey. Don't miss out on valuable insights – hit that subscribe button now and embark on a programming adventure with us!

Subscribe to our channel:
https://www.youtube.com/@HTEASYLEARNING