# TUTORIAL 1

1)

Programming languages are essential tools for humans to communicate instructions to computers. They serve as a means to write software programs that control the behavior of computer systems and accomplish specific tasks. Programming languages provide a structured and systematic way to express algorithms, data manipulation, and logical operations. They enable developers to write code that is understandable by both humans and machines, making it easier to develop complex software systems, automate tasks, solve problems, and create innovative applications.

2)

* Source Code vs. Machine Code:

Source code is the human-readable form of a program written in a programming language. It consists of instructions and statements that are written using a specific syntax and structure. Machine code, on the other hand, is the binary representation of instructions that can be directly executed by the computer's processor. It consists of a series of binary digits (0s and 1s) that represent specific instructions and data.

* High-Level Language vs. Low-Level Language:

High-level languages are designed to be closer to human language and are more user-friendly. They provide abstractions and built-in functionality that make it easier to develop complex applications. Examples include Python, Java, and C++. Low-level languages are closer to the hardware and provide direct control over the computer's resources. They require a deeper understanding of the underlying hardware and are more difficult to program. Examples include assembly language and machine code.

* Compiler vs. Interpreter:

A compiler is a software tool that translates the entire source code into machine code before execution. It performs a complete analysis of the code, checks for errors, and generates an executable file. The resulting program can be executed directly by the computer's processor. An interpreter, on the other hand, translates and executes the source code line by line, without generating an executable file. It interprets and executes each statement as it encounters it. Interpreted languages typically have a shorter development cycle as changes in the code can be immediately tested without the need for compilation.

* Structured Language vs. Object-Oriented Language:

Structured programming languages focus on organizing code into logical structures such as sequences, loops, and conditionals. They emphasize modular design and procedural programming. Examples include C and Pascal. Object-oriented programming languages (OOP) provide features such as classes, objects, and inheritance, which allow for the organization of code around objects that encapsulate data and behavior. OOP promotes reusability, modularity, and easier maintenance. Examples include C++, Java, and Python.

* C vs. C++:

C is a procedural programming language that provides low-level control over the computer's resources. It is widely used for system programming, embedded systems, and performancecritical applications. C++ is an extension of the C language with added support for objectoriented programming. It includes features like classes, inheritance, and polymorphism. C++ combines the efficiency and low-level control of C with the benefits of object-oriented programming.

* C++ vs. Java:

C++ and Java are both high-level programming languages but differ in several aspects. C++ is a statically typed language that allows low-level control and direct memory manipulation. It provides manual memory management and supports procedural and object-oriented programming paradigms. Java, on the other hand, is a strongly typed language that runs on a

virtual machine (JVM). It has automatic memory management through garbage collection and focuses on platform independence, security, and simplicity. Java is mainly used for building enterprise applications, web applications, and Android apps.

* Syntax error vs. Logical error:

A syntax error occurs when the code violates the rules and structure of the programming language. It often results in a compilation error or runtime error and prevents the program from executing. Examples include missing semicolons, mismatched parentheses, or incorrect variable names. A logical error, on the other hand, occurs when the program compiles and runs without errors but produces incorrect results. Logical errors are flaws in the design or implementation of the code and require debugging and logical reasoning to identify and fix.