Introduction to high level language-

In computer science, a **high-level programming language** is a programming language with strong abstraction from the details of the computer. In contrast to low-level programming languages, it may use natural language *elements*, be easier to use, or may automate (or even hide entirely) significant areas of computing systems (e.g. memory management), making the process of developing a program simpler and more understandable than when using a lower-level language. The amount of abstraction provided defines how "high-level" a programming language is. In the 1960s, high-level programming languages using a compiler were commonly called **autocodes** Examples of autocodes are COBOL and Fortran. The first high-level programming language designed for computers was Plankalkül, created by Konrad Zuse. However, it was not implemented in his time, and his original contributions were largely isolated from other developments due to World War II, aside from the language's influence on the "Superplan" language by Heinz Rutishauser and also to some degree Algol. The first significantly widespread high-level language was Fortran, a machine-independent development of IBM's earlier Autocode systems. Algol, defined in 1958 and 1960 by committees of European and American computer scientists, introduced recursion as well as nested functions under lexical scope. It was also the first language with a clear distinction between value and name-parameters and their corresponding semantics. Algol also introduced several structured programming concepts, such as the **while-do** and **if-then-else** constructs and its syntax was the first to be described in formal notation – "Backus–Naur form" (BNF). During roughly the same period, Cobol introduced records (also called structs) and Lisp introduced a fully general lambda abstraction in a programming language for the first time.

"High-level language" refers to the higher level of abstraction from machine language. Rather than dealing with registers, memory addresses, and call stacks, high-level languages deal with variables, arrays, objects, complex arithmetic or boolean expressions, subroutines and functions, loops, threads, locks, and other abstract computer science concepts, with a focus on usability over optimal program efficiency. Unlike low-level assembly languages, high-level languages have few, if any, language elements that translate directly into a machine's native opcodes. Other features, such as string handling routines, object-oriented language features, and file input/output, may also be present. One thing to note about high-level programming languages is that these languages allow the programmer to be detached and separated from the machine. That is, unlike low-level languages like assembly or machine language, high-level programming can amplify the programmer's instructions and trigger a lot of data movements in the background without their knowledge. The responsibility and power of executing instructions have been handed over to the machine from the programmer.

Introduction to Low/Middle/High Level Languages-

- **Low level languages / Machine Oriented Languages**

The language whose design is governed by the circuitry and the structure of the machine is known as the Machine language. This language is difficult to learn and use. It is specific to a given computer and is different for different computers i.e. these languages are machine-dependent. These languages have been designed to give a better machine efficiency, i.e. faster program execution. Such languages are also known as Low Level Languages.

- **High level Languages**

They are easy to learn and programs may be written in these languages with much less effort. However, the computer cannot understand them and they need to be translated into machine language with the help of other programs known as Compilers or Translators. Like C++, Java, Python

- **Middle level Languages**

It bridges gap between machine understandable machine level language and more conventional high-level language. This programming helps in writing system programming as well as application programming. Because C language has both the feature of high level and low-level language so it is often called middle level language.

Difference between high level and low level lang-

1.	It is programmer friendly language.	It is a machine friendly language.
2.	High level language is less memory efficient.	Low level language is high memory efficient.
3.	It is easy to understand.	It is tough to understand.
4.	It is simple to debug.	It is complex to debug comparatively.
5.	It is simple to maintain.	It is complex to maintain comparatively.
6.	It is portable.	It is non-portable.
7.	It can run on any platform.	It is machine-dependent.
8.	It needs compiler or interpreter for translation.	It needs assembler for translation.
9.	It is used widely for programming.	It is not commonly used now-a-days in programming