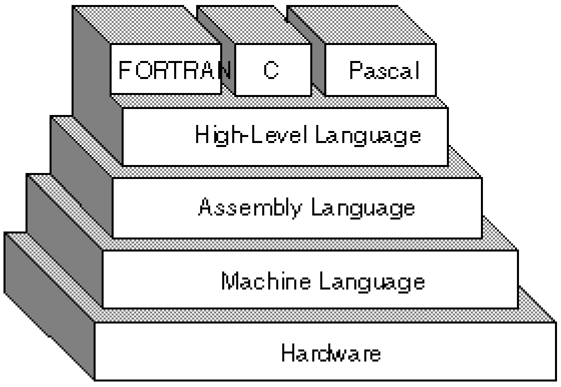
Language-----------------
programmers use ------

converted into ----------

converted into ----------



*

- Programs written in high-level languages are translated into assembly language or machine language by a compiler.
- Assembly language programs are translated into machine language by a program called an assembler.

*

What is high level and low level language ? :-

High level language:-

A h*igh-level* languag (**HLL**) is a programming language such as C, FORTRAN, or Pascal that enables a programmer to write programsthat are more or less independent of a particular type of computer. Such languages are considered high-level because they are closer to human languages and further from machine languages.

High-level programming languages, while simple compared to human languages, are more complex than the languages the computer actually understands, called machine languages

When we think about computer programmers, we are probably thinking about people who write in high-level programming languages.

High level languages are written in a form that is close to our human language, enabling to programmer to just focus on the problem being solved.

No particular knowledge of the hardware is needed as high level languages create programs that are portable and not tied to a particular computer or microchip.

Examples include: C++, Java, Pascal, Python, Visual Basic.

Low level language :-

Low level languages are used to write programs that relate to the specific architecture and hardware of a particular type of computer.
They are closer to the native language of a computer (binary), making them harder for programmers to understand.
Ex- assembly language and Machine language etc.

programming language:-

A programming language is a vocabulary and set of grammatical rules for instructing a **computer** or computing device to perform specific tasks. The term programming language usually refers to high-level languages, such as BASIC, C, C++, COBOL, Java, FORTRAN, Ada, and Pascal.

Each programming language has a unique set of keywords (words that it understands) and a special syntax for organizing program instructions.

Assembly language:-

Lying between machine languages and high-level languages are languages called assembly languages. Assembly languages are similar to machine languages, but they are much easier to program in because they allow a programmer to substitute names for numbers. Machine languages consist of numbers only.

Machine Language:-

Machine language is the lowest-level programming language(except for computers that utilize programmable microcode). Machine languages are the only languages understood by computers.

Why Humans Don't Use Machine Language ?

While easily understood by computers, machine languages are almost impossible for humans to use because they consist entirely of numbers. Programmers, therefore, use either a high-level programming language or an assembly language. An assembly language contains the same **instructions** as a machine language, but the instructions and **variables** have **names** instead of being just numbers.

interpreted Language:-

An **interpreted language** is a type of programming language for which most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions. The interpreter executes the program directly, translating each statement into a sequence of one or more subroutines, and then into another language (often machine code).

Scripting Language:-

A scripting or script language is a programming language for a special run-time environment that automates the execution of tasks; the tasks could alternatively be executed one-by-one by a human operator. Scripting languages are often interpreted (rather than compiled). ex- javascript , php etc.

Compiled Language:-

A compiled language is a **programming language** whose implementations are typically **compilers** (translators that generate machine code from source code), and not interpreters (step-by-step executors of source code, where no pre-runtime translation takes place).

Language differentiation according to Generation--

1. **First-generation languages:** machine code

2. Second-generation languages: assembly languages

3. Third-generation languages: largely “standalone” languages like C, C++, and Fortran, as well as languages that use VMs and runtimes like C#, Python, or Java. The basic idea is you can implement a complete executable application in a third-generation language (although it may require some sort of VM/interpreter/runtime). Part of the problem with the idea of “third-generation language” is it’s rather over-broad and quite likely could be broken into several “sub-generations”, with (as an example) 3a languages being C, C++, and Fortran, 3b languages being C# and Java, and 3c languages being Python, LISP, Haskell, and similar languages.

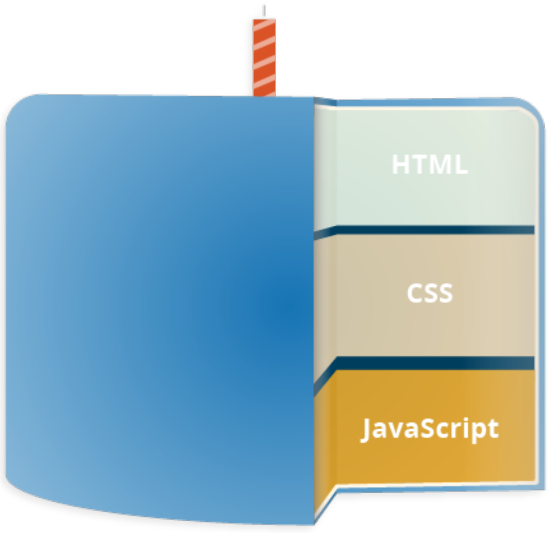
4. Fourth-generation languages: Languages like SQL as well as R, SAS, Bash and other shell languages, and similar special-purpose languages. The basic idea here is these are languages that are used to instruct some other piece of software to do a complex task, such as a DB engine in the case of SQL, the shell in the case of Bash, or the R or SAS applications in the case of these languages. I’d probably put most web languages such as Javascript as 4th-generation languages.

5. Fifth-generation languages: AI description-oriented languages of various types. “Fifth-generation language” was a hot idea in the late 1980s and early 1990s, but is rarely used nowadays outside of marketing.

JavaScript

an object-oriented computer programming language commonly used to create interactive effects within web browsers.

Javascript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript,



HTML is the markup language that we use to structure and give meaning to our web content, for example defining paragraphs, headings, and data tables, or embedding images and videos in the page.

CSS is a language of style rules that we use to apply styling to our HTML content, for example setting background colors and fonts, and laying out our content in multiple columns.

JavaScript is a scripting language that enables you to create dynamically updating content, control multimedia, animate images, and pretty much everything else. (Okay, not everything, but it is amazing what you can achieve with a few lines of JavaScript code.)

Node.js is an open-source, cross-platform JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web application development around a single programming language, rather than different languages for server side and client side scripts.