Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit.  
This is interpreted into machine code.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
To produce machine code, the source code must either be compiled or transpiled.  
Techniques like Code refactoring can enhance readability.  
The following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Techniques like Code refactoring can enhance readability.  
 Computer programmers are those who write computer software.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
There exist a lot of different approaches for each of those tasks.