

Allen Downey, in his book *How To Think Like A Computer Scientist*, writes: Many computer languages provide a mechanism to call functions provided by shared libraries. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. One approach popular for requirements analysis is Use Case analysis. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Programming languages are essential for software development. There are many approaches to the Software development process. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Techniques like Code refactoring can enhance readability. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. 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. Many applications use a mix of several languages in their construction and use. 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. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Many applications use a mix of several languages in their construction and use. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. 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. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.