

For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Integrated development environments (IDEs) aim to integrate all such help. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. 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). Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. Integrated development environments (IDEs) aim to integrate all such help. Different programming languages support different styles of programming (called programming paradigms). Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Whatever the approach to development may be, the final program must satisfy some fundamental properties. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. There exist a lot of different approaches for each of those tasks. The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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).