

Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Scripting and breakpointing is also part of this process. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. 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. Code-breaking algorithms have also existed for centuries. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Scripting and breakpointing is also part of this process. 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. Techniques like Code refactoring can enhance readability. There are many approaches to the Software development process. Whatever the approach to development may be, the final program must satisfy some fundamental properties. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. 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. Computer programmers are those who write computer software. 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. Following a consistent programming style often helps readability. Code-breaking algorithms have also existed for centuries. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability.