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. Scripting and breakpointing is also part of this process. 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. 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. Computer programmers are those who write computer software. 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. 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. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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 is usually easier to code in "high-level" languages than in "low-level" ones. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Whatever the approach to development may be, the final program must satisfy some fundamental properties. However, readability is more than just programming style. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. 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. 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. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. One approach popular for requirements analysis is Use Case analysis. 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. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). Integrated development environments (IDEs) aim to integrate all such help. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.