Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. 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. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Code-breaking algorithms have also existed for centuries. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Programming languages are essential for software development. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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). Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Also, specific user environment and usage history can make it difficult to reproduce the problem. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. Unreadable code often leads to bugs, inefficiencies, and duplicated code. One approach popular for requirements analysis is Use Case analysis. Computer programmers are those who write computer software. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Programs were mostly entered using punched cards or paper tape. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code.