The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Programming languages are essential for software development. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. 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. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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. However, readability is more than just programming style. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. 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. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. 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. 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.