Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. 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. Code-breaking algorithms have also existed for centuries. 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. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. Ideally, the programming language best suited for the task at hand will be selected. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. It is very difficult to determine what are the most popular modern programming languages. 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 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Use of a static code analysis tool can help detect some possible problems. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. 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. 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. There exist a lot of different approaches for each of those tasks. 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.