Code-breaking algorithms have also existed for centuries. Following a consistent programming style often helps readability. 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. 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. Scripting and breakpointing is also part of this process. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). It is very difficult to determine what are the most popular modern programming languages. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. 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. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. There exist a lot of different approaches for each of those tasks. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Whatever the approach to development may be, the final program must satisfy some fundamental properties. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language.