The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. Scripting and breakpointing is also part of this process. Scripting and breakpointing is also part of this process. 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. 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. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. 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. 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. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. 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. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Ideally, the programming language best suited for the task at hand will be selected. 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. Integrated development environments (IDEs) aim to integrate all such help. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Normally the first step in debugging is to attempt to reproduce the problem. FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Code-breaking algorithms have also existed for centuries. There are many approaches to the Software development process.