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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Also, specific user environment and usage history can make it difficult to reproduce the problem. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. 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. Many applications use a mix of several languages in their construction and use. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. Scripting and breakpointing is also part of this process. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Normally the first step in debugging is to attempt to reproduce the problem. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Programmable devices have existed for centuries. 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. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Following a consistent programming style often helps readability. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. 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.