

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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. There are many approaches to the Software development process. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. One approach popular for requirements analysis is Use Case analysis. Different programming languages support different styles of programming (called programming paradigms). The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. 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. 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. Use of a static code analysis tool can help detect some possible problems. 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. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Computer programmers are those who write computer 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).