By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Some of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. Different programming languages support different styles of programming (called programming paradigms). 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. However, readability is more than just programming style. Different programming languages support different styles of programming (called programming paradigms). Programmable devices have existed for centuries. 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. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Computer programmers are those who write computer software. Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. 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). Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic.