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. Programs were mostly entered using punched cards or paper tape. Scripting and breakpointing is also part of this process. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Scripting and breakpointing is also part of this process. Programmable devices have existed for centuries. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Computer programmers are those who write computer software. Code-breaking algorithms have also existed for centuries. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. It affects the aspects of quality above, including portability, usability and most importantly maintainability. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. 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.