They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. Scripting and breakpointing is also part of this process. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Ideally, the programming language best suited for the task at hand will be selected. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. Scripting and breakpointing is also part of this process. Integrated development environments (IDEs) aim to integrate all such help. One approach popular for requirements analysis is Use Case analysis. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. 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. However, readability is more than just programming style. 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. 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. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. One approach popular for requirements analysis is Use Case analysis. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Following a consistent programming style often helps readability.