Integrated development environments (IDEs) aim to integrate all such help.  
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.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
To produce machine code, the source code must either be compiled or transpiled.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
Use of a static code analysis tool can help detect some possible problems.  
One approach popular for requirements analysis is Use Case analysis.  
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.  
Programming languages are essential for software development.  
There exist a lot of different approaches for each of those tasks.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
However, readability is more than just programming style.