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.  
There are many approaches to the Software development process.  
Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists.  
Scripting and breakpointing is also part of this process.  
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.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
Use of a static code analysis tool can help detect some possible problems.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
Relatedly, software engineering combines engineering techniques and principles with software development.  
 It is very difficult to determine what are the most popular modern programming languages.  
Normally the first step in debugging is to attempt to reproduce the problem.  
Integrated development environments (IDEs) aim to integrate all such help.  
 Tasks accompanying and related to programming include testing, debugging, source code maintenance, implementation of build systems, and management of derived artifacts, such as the machine code of computer programs.