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.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
Ideally, the programming language best suited for the task at hand will be selected.  
 Different programming languages support different styles of programming (called programming paradigms).  
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.  
 Programmable devices have existed for centuries.  
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.  
 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.  
 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.  
Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
 Following a consistent programming style often helps readability.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.