Assembly languages were soon developed that let the programmer specify instruction in a text format (e..g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses.  
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.  
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.  
 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.  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
There are many approaches to the Software development process.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
Programming languages are essential for software development.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.