Techniques like Code refactoring can enhance readability.  
To produce machine code, the source code must either be compiled or transpiled.  
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.  
Normally the first step in debugging is to attempt to reproduce the problem.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
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.  
Techniques like Code refactoring can enhance readability.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Programming languages are essential for software development.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
However, while these might be considered part of the programming process, often the term software development is more likely used for this larger overall process – whereas the terms programming, implementation, and coding tend to be focused on the actual writing of code.