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..  
It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages.  
 Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code.  
Normally the first step in debugging is to attempt to reproduce the problem.  
 Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.  
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.  
For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
  
 Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.  
Techniques like Code refactoring can enhance readability.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.