Whatever the approach to development may be, the final program must satisfy some fundamental properties..  
 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.  
 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.  
 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.  
There exist a lot of different approaches for each of those tasks.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
 Different programming languages support different styles of programming (called programming paradigms).  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
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.