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.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
There are many approaches to the Software development process.  
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.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Normally the first step in debugging is to attempt to reproduce the problem.  
As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices.  
Programming languages are essential for software development.  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
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).  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
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.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.