Compiling takes the source code from a low-level programming language and converts it into machine 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.  
To produce machine code, the source code must either be compiled or transpiled.  
Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly.  
To produce machine code, the source code must either be compiled or transpiled.  
Scripting and breakpointing is also part of this process.  
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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Normally the first step in debugging is to attempt to reproduce the problem.  
 Programmable devices have existed for centuries.  
Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
 Code-breaking algorithms have also existed for centuries.  
  
 Computer programming is the process of performing particular computations (or more generally, accomplishing specific computing results), usually by designing and building executable computer programs.  
In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them.