Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.  
Techniques like Code refactoring can enhance readability.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
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.  
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.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
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.