Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.  
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.  
Relatedly, software engineering combines engineering techniques and principles with software development.  
There exist a lot of different approaches for each of those tasks.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
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.  
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.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.