They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
Many applications use a mix of several languages in their construction and use.  
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.  
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.  
Techniques like Code refactoring can enhance readability.  
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.  
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.  
Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language.  
This is interpreted into machine code.  
Programming languages are essential for software development.  
 Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA).  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
Scripting and breakpointing is also part of this process.