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..  
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.  
 Following a consistent programming style often helps readability.  
There are many approaches to the Software development process.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
 Code-breaking algorithms have also existed for centuries.  
Many applications use a mix of several languages in their construction and use.  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
 Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code.  
Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).  
 In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form.  
For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.