Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
Normally the first step in debugging is to attempt to reproduce the problem.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
This is interpreted into machine code.  
However, readability is more than just programming style.  
Programming involves tasks such as analysis, generating algorithms, profiling algorithms' accuracy and resource consumption, and the implementation of algorithms (usually in a particular programming language, commonly referred to as coding).  
Also, those involved with software development may at times engage in reverse engineering, which is the practice of seeking to understand an existing program so as to re-implement its function in some way.  
To produce machine code, the source code must either be compiled or transpiled.  
The source code of a program is written in one or more languages that are intelligible to programmers, rather than machine code, which is directly executed by the central processing unit.  
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.  
Normally the first step in debugging is to attempt to reproduce the problem.  
There exist a lot of different approaches for each of those tasks.  
When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.