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.  
There exist a lot of different approaches for each of those tasks.  
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.  
This is interpreted into machine code.  
Techniques like Code refactoring can enhance readability.  
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.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
One approach popular for requirements analysis is Use Case analysis.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
 The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems.  
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.  
 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.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
One approach popular for requirements analysis is Use Case analysis.  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.