Programming languages are essential for software development.  
This is interpreted into machine code.  
This can be a non-trivial task, for example as with parallel processes or some unusual software bugs.  
Ideally, the programming language best suited for the task at hand will be selected.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
One approach popular for requirements analysis is Use Case analysis.  
Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards.  
 Whatever the approach to development may be, the final program must satisfy some fundamental properties.  
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.  
 Tasks accompanying and related to programming include testing, debugging, source code maintenance, implementation of build systems, and management of derived artifacts, such as the machine code of computer programs.  
Integrated development environments (IDEs) aim to integrate all such help.  
 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.  
It affects the aspects of quality above, including portability, usability and most importantly maintainability.