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.  
Programming languages are essential for software development.  
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.  
Techniques like Code refactoring can enhance readability.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Integrated development environments (IDEs) aim to integrate all such help.  
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.  
One approach popular for requirements analysis is Use Case analysis.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances.  
 The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine.  
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.  
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.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.