He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
The following properties are among the most important:  
  
 In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code.  
Normally the first step in debugging is to attempt to reproduce the problem.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.  
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.  
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.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
However, while these might be considered part of the programming process, often the term software development is more likely used for this larger overall process – whereas the terms programming, implementation, and coding tend to be focused on the actual writing of code.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute.  
Scripting and breakpointing is also part of this process.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 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.  
Proficient programming thus usually requires expertise in several different subjects, including knowledge of the application domain, specialized algorithms, and formal logic.  
 Allen Downey, in his book How To Think Like A Computer Scientist, writes:  
 Many computer languages provide a mechanism to call functions provided by shared libraries.