However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
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.  
Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL).  
A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it.  
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.  
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.  
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.  
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.  
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.  
 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.  
 Different programming languages support different styles of programming (called programming paradigms).  
 It is very difficult to determine what are the most popular modern programming languages.  
 New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation).