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.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
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.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
Techniques like Code refactoring can enhance readability.  
Normally the first step in debugging is to attempt to reproduce the problem.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
Many applications use a mix of several languages in their construction and use.  
 Different programming languages support different styles of programming (called programming paradigms).  
 Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation.  
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.  
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.