

Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. Different programming languages support different styles of programming (called programming paradigms). Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. It is very difficult to determine what are the most popular modern programming languages. 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. 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. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Debugging is often done with IDEs. Standalone debuggers like GDB are also used, and these often provide less of a visual environment, usually using a command line. 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. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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). Following a consistent programming style often helps readability. They are the building blocks for all software, from the simplest applications to the most sophisticated ones.