

Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. Integrated development environments (IDEs) aim to integrate all such help. Following a consistent programming style often helps readability. Many applications use a mix of several languages in their construction and use. Normally the first step in debugging is to attempt to reproduce the problem. 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. 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. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. 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. Computer programmers are those who write computer software. Scripting and breakpointing is also part of this process. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Following a consistent programming style often helps readability. 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. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Scripting and breakpointing is also part of this process. However, readability is more than just programming style. 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.