

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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Ideally, the programming language best suited for the task at hand will be selected. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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). 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). In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Many applications use a mix of several languages in their construction and use. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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.