

Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. 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. 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. There are many approaches to the Software development process. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. 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. Following a consistent programming style often helps readability. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in *A Manuscript on Deciphering Cryptographic Messages*. Integrated development environments (IDEs) aim to integrate all such help. Integrated development environments (IDEs) aim to integrate all such help. There exist a lot of different approaches for each of those tasks. In 1206, the Arab engineer Al-Jazari invented a programmable drum machine where a musical mechanical automaton could be made to play different rhythms and drum patterns, via pegs and cams. Whatever the approach to development may be, the final program must satisfy some fundamental properties. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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). High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users.