

Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Techniques like Code refactoring can enhance readability. 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). Many applications use a mix of several languages in their construction and use. However, readability is more than just programming style. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic 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. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). 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. Code-breaking algorithms have also existed for centuries. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. 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. Different programming languages support different styles of programming (called programming paradigms). Scripting and breakpointing is also part of this process. It is very difficult to determine what are the most popular modern programming languages. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.