By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the 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. Integrated development environments (IDEs) aim to integrate all such help. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Also, specific user environment and usage history can make it difficult to reproduce the problem. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Programs were mostly entered using punched cards or paper tape. Code-breaking algorithms have also existed for centuries. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Scripting and breakpointing is also part of this process. 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. Following a consistent programming style often helps readability. Integrated development environments (IDEs) aim to integrate all such help. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. Programming languages are essential for software development. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Computer programmers are those who write computer software. 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.