Code-breaking algorithms have also existed for centuries. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Also, specific user environment and usage history can make it difficult to reproduce the problem. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Different programming languages support different styles of programming (called programming paradigms). Programming languages are essential for software development. 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. 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. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Use of a static code analysis tool can help detect some possible problems. 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. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. There are many approaches to the Software development process. There exist a lot of different approaches for each of those tasks. 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. Normally the first step in debugging is to attempt to reproduce the problem. 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 of these factors include: The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills. Following a consistent programming style often helps readability.