It is usually easier to code in "high-level" languages than in "low-level" ones. By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Integrated development environments (IDEs) aim to integrate all such help. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Programming languages are essential for software development. One approach popular for requirements analysis is Use Case analysis. Programming languages are essential for software development. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. 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). Scripting and breakpointing is also part of this process. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Programming languages are essential for software development. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic 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. Integrated development environments (IDEs) aim to integrate all such help. Normally the first step in debugging is to attempt to reproduce the problem. It affects the aspects of quality above, including portability, usability and most importantly maintainability. 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).