Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. 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. Many applications use a mix of several languages in their construction and use. Following a consistent programming style often helps readability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics. 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. 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. 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. It is very difficult to determine what are the most popular modern programming languages. Programmable devices have existed for centuries. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Code-breaking algorithms have also existed for centuries. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. Code-breaking algorithms have also existed for centuries. 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. Debugging is a very important task in the software development process since having defects in a program can have significant consequences for its users. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. One approach popular for requirements analysis is Use Case analysis. Techniques like Code refactoring can enhance readability. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. Integrated development environments (IDEs) aim to integrate all such help. It is usually easier to code in "high-level" languages than in "low-level" ones.