

However, Charles Babbage had already written his first program for the Analytical Engine in 1837. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Programming languages are essential for software development. 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. It is very difficult to determine what are the most popular modern programming languages. Normally the first step in debugging is to attempt to reproduce the problem. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. It is usually easier to code in "high-level" languages than in "low-level" ones. Integrated development environments (IDEs) aim to integrate all such help. 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. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. 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. However, readability is more than just programming style. There are many approaches to the Software development process. Unreadable code often leads to bugs, inefficiencies, and duplicated code. Various visual programming languages have also been developed with the intent to resolve readability concerns by adopting non-traditional approaches to code structure and display. However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages. Use of a static code analysis tool can help detect some possible problems. 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. Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se.