They are the building blocks for all software, from the simplest applications to the most sophisticated ones. Languages form an approximate spectrum from "low-level" to "high-level"; "low-level" languages are typically more machine-oriented and faster to execute, whereas "high-level" languages are more abstract and easier to use but execute less quickly. 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. Programming languages are essential for software development. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. 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, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Many applications use a mix of several languages in their construction and use. 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. 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. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Computer programmers are those who write computer software. Their jobs usually involve: Although programming has been presented in the media as a somewhat mathematical subject, some research shows that good programmers have strong skills in natural human languages, and that learning to code is similar to learning a foreign language. It is very difficult to determine what are the most popular modern programming languages. Code-breaking algorithms have also existed for centuries. 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. One approach popular for requirements analysis is Use Case analysis. 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. 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. 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. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. 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.