Programming languages are essential for software development. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. 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. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Normally the first step in debugging is to attempt to reproduce the problem. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. Code-breaking algorithms have also existed for centuries. 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. 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. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Auxiliary tasks accompanying and related to programming include analyzing requirements, testing, debugging (investigating and fixing problems), implementation of build systems, and management of derived artifacts, such as programs' machine code. Use of a static code analysis tool can help detect some possible problems. 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. 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. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Code-breaking algorithms have also existed for centuries. Programs were mostly entered using punched cards or paper tape. Sometimes software development is known as software engineering, especially when it employs formal methods or follows an engineering design process.