Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. In the 9th century, the Arab mathematician Al-Kindi described a cryptographic algorithm for deciphering encrypted code, in A Manuscript on Deciphering Cryptographic Messages. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Also, specific user environment and usage history can make it difficult to reproduce the problem. 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. 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. Programming languages are essential for software development. Following a consistent programming style often helps readability. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Techniques like Code refactoring can enhance readability. It is usually easier to code in "high-level" languages than in "low-level" ones. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. 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. However, readability is more than just programming style. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Methods of measuring programming language popularity include: counting the number of job advertisements that mention the language, the number of books sold and courses teaching the language (this overestimates the importance of newer languages), and estimates of the number of existing lines of code written in the language (this underestimates the number of users of business languages such as COBOL). Following a consistent programming style often helps readability. 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. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. Many applications use a mix of several languages in their construction and use. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages.