Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). However, Charles Babbage had already written his first program for the Analytical Engine in 1837. The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. Trade-offs from this ideal involve finding enough programmers who know the language to build a team, the availability of compilers for that language, and the efficiency with which programs written in a given language execute. 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. Trial-and-error/divide-and-conquer is needed: the programmer will try to remove some parts of the original test case and check if the problem still exists. Different programming languages support different styles of programming (called programming paradigms). 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. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Programming languages are essential for software development. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. The following properties are among the most important: In computer programming, readability refers to the ease with which a human reader can comprehend the purpose, control flow, and operation of source code. It is usually easier to code in "high-level" languages than in "low-level" ones. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Following a consistent programming style often helps readability. There exist a lot of different approaches for each of those tasks. 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. Many factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Following a consistent programming style often helps readability. 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.