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 computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. It is usually easier to code in "high-level" languages than in "low-level" ones. Programming languages are essential for software development. 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. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. It affects the aspects of quality above, including portability, usability and most importantly maintainability. Programmable devices have existed for centuries. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. 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. 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. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Unreadable code often leads to bugs, inefficiencies, and duplicated code. 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. 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. 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. 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. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Techniques like Code refactoring can enhance readability. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Different programming languages support different styles of programming (called programming paradigms).