Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Techniques like Code refactoring can enhance readability. For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Different programming languages support different styles of programming (called programming paradigms). Scripting and breakpointing is also part of this process. It is usually easier to code in "high-level" languages than in "low-level" ones. 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. 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. 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. 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. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. 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. Following a consistent programming style often helps readability. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. Following a consistent programming style often helps readability. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Many applications use a mix of several languages in their construction and use. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. For this purpose, algorithms are classified into orders using so-called Big O notation, which expresses resource use, such as execution time or memory consumption, in terms of the size of an input. As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices. 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.