The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Many programmers use forms of Agile software development where the various stages of formal software development are more integrated together into short cycles that take a few weeks rather than years. 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. They are the building blocks for all software, from the simplest applications to the most sophisticated 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. Ideally, the programming language best suited for the task at hand will be selected. In the 1880s, Herman Hollerith invented the concept of storing data in machine-readable form. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. In 1801, the Jacquard loom could produce entirely different weaves by changing the "program" – a series of pasteboard cards with holes punched in them. 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. 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. It is usually easier to code in "high-level" languages than in "low-level" ones. Expert programmers are familiar with a variety of well-established algorithms and their respective complexities and use this knowledge to choose algorithms that are best suited to the circumstances. However, Charles Babbage had already written his first program for the Analytical Engine in 1837. 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. When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear. 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). 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. Also, specific user environment and usage history can make it difficult to reproduce the problem. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Normally the first step in debugging is to attempt to reproduce the problem. Programs were mostly entered using punched cards or paper tape. Programs were mostly entered using punched cards or paper tape.