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. 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. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. A study found that a few simple readability transformations made code shorter and drastically reduced the time to understand it. Techniques like Code refactoring can enhance readability. Different programming languages support different styles of programming (called programming paradigms). 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, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. Normally the first step in debugging is to attempt to reproduce the problem. However, readability is more than just programming style. 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 first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). Readability is important because programmers spend the majority of their time reading, trying to understand, reusing and modifying existing source code, rather than writing new source code. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Programming languages are essential for software development. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. 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. Unreadable code often leads to bugs, inefficiencies, and duplicated code. It affects the aspects of quality above, including portability, usability and most importantly maintainability. He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm. However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.