By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. While these are sometimes considered programming, often the term software development is used for this larger overall process – with the terms programming, implementation, and coding reserved for the writing and editing of code per se. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. 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. It is very difficult to determine what are the most popular modern programming languages. Programming languages are essential for software development. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. However, readability is more than just programming style. One approach popular for requirements analysis is Use Case analysis. Programmers typically use high-level programming languages that are more easily intelligible to humans than machine code, which is directly executed by the central processing unit. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. Integrated development environments (IDEs) aim to integrate all such help. The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging). 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. 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. Programmable devices have existed for centuries. Different programming languages support different styles of programming (called programming paradigms). There are many approaches to the Software development process. 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. 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. Whatever the approach to development may be, the final program must satisfy some fundamental properties.