The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'. Different programming languages support different styles of programming (called programming paradigms). By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers. 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. Scripting and breakpointing is also part of this process. The academic field and the engineering practice of computer programming are both largely concerned with discovering and implementing the most efficient algorithms for a given class of problems. For example, COBOL is still strong in corporate data centers often on large mainframe computers, Fortran in engineering applications, scripting languages in Web development, and C in embedded software. After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug. It is very difficult to determine what are the most popular modern programming languages. Machine code was the language of early programs, written in the instruction set of the particular machine, often in binary notation. The Unified Modeling Language (UML) is a notation used for both the OOAD and MDA. Many applications use a mix of several languages in their construction and use. 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. 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. Code-breaking algorithms have also existed for centuries. Proficient programming usually requires expertise in several different subjects, including knowledge of the application domain, details of programming languages and generic code libraries, specialized algorithms, and formal logic. 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. Scripting and breakpointing is also part of this process. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Ideally, the programming language best suited for the task at hand will be selected. Ideally, the programming language best suited for the task at hand will be selected. Many applications use a mix of several languages in their construction and use. 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. 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.