Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications. Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment. Ideally, the programming language best suited for the task at hand will be selected. It involves designing and implementing algorithms, step-by-step specifications of procedures, by writing code in one or more programming languages. 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. New languages are generally designed around the syntax of a prior language with new functionality added, (for example C++ adds object-orientation to C, and Java adds memory management and bytecode to C++, but as a result, loses efficiency and the ability for low-level manipulation). Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation. Provided the functions in a library follow the appropriate run-time conventions (e.g., method of passing arguments), then these functions may be written in any other language. This can be a non-trivial task, for example as with parallel processes or some unusual software bugs. The first computer program is generally dated to 1843, when mathematician Ada Lovelace published an algorithm to calculate a sequence of Bernoulli numbers, intended to be carried out by Charles Babbage's Analytical Engine. Text editors were also developed that allowed changes and corrections to be made much more easily than with punched cards. Computer programming or coding is the composition of sequences of instructions, called programs, that computers can follow to perform tasks. Whatever the approach to development may be, the final program must satisfy some fundamental properties. Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic languages. Popular modeling techniques include Object-Oriented Analysis and Design (OOAD) and Model-Driven Architecture (MDA). Assembly languages were soon developed that let the programmer specify instruction in a text format (e.g., ADD X, TOTAL), with abbreviations for each operation code and meaningful names for specifying addresses. A similar technique used for database design is Entity-Relationship Modeling (ER Modeling). Programs were mostly entered using punched cards or paper tape. 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. Different programming languages support different styles of programming (called programming paradigms). Whatever the approach to development may be, the final program must satisfy some fundamental properties. 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. High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware. 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. 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.