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.  
 Implementation techniques include imperative languages (object-oriented or procedural), functional languages, and logic 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.  
 Different programming languages support different styles of programming (called programming paradigms).  
Scripting and breakpointing is also part of this process.  
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).  
  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
 Programs were mostly entered using punched cards or paper tape.  
 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.  
 After the bug is reproduced, the input of the program may need to be simplified to make it easier to debug.  
 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.  
Some of these factors include:  
 The presentation aspects of this (such as indents, line breaks, color highlighting, and so on) are often handled by the source code editor, but the content aspects reflect the programmer's talent and skills.  
One approach popular for requirements analysis is Use Case analysis.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).