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..  
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.  
 Code-breaking algorithms have also existed for centuries.  
  
The first compiler related tool, the A-0 System, was developed in 1952 by Grace Hopper, who also coined the term 'compiler'.  
It is usually easier to code in "high-level" languages than in "low-level" ones.  
 Some languages are very popular for particular kinds of applications, while some languages are regularly used to write many different kinds of applications.  
 Programmable devices have existed for centuries.  
They are the building blocks for all software, from the simplest applications to the most sophisticated ones.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
FORTRAN, the first widely used high-level language to have a functional implementation, came out in 1957, and many other languages were soon developed—in particular, COBOL aimed at commercial data processing, and Lisp for computer research.  
 Programs were mostly entered using punched cards or paper tape.  
He gave the first description of cryptanalysis by frequency analysis, the earliest code-breaking algorithm.  
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.  
 High-level languages made the process of developing a program simpler and more understandable, and less bound to the underlying hardware.  
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).