By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
Many applications use a mix of several languages in their construction and use.  
Later a control panel (plug board) added to his 1906 Type I Tabulator allowed it to be programmed for different jobs, and by the late 1940s, unit record equipment such as the IBM 602 and IBM 604, were programmed by control panels in a similar way, as were the first electronic computers.  
Many applications use a mix of several languages in their construction and use.  
Some languages are more prone to some kinds of faults because their specification does not require compilers to perform as much checking as other languages.  
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.  
Normally the first step in debugging is to attempt to reproduce the problem.  
However, with the concept of the stored-program computer introduced in 1949, both programs and data were stored and manipulated in the same way in computer memory.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Compiling takes the source code from a low-level programming language and converts it into machine code.  
There exist a lot of different approaches for each of those tasks.  
 The first step in most formal software development processes is requirements analysis, followed by testing to determine value modeling, implementation, and failure elimination (debugging).  
Relatedly, software engineering combines engineering techniques and principles with software development.  
Some text editors such as Emacs allow GDB to be invoked through them, to provide a visual environment.  
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.