When debugging the problem in a GUI, the programmer can try to skip some user interaction from the original problem description and check if remaining actions are sufficient for bugs to appear.  
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.  
Unreadable code often leads to bugs, inefficiencies, and duplicated code.  
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 factors, having little or nothing to do with the ability of the computer to efficiently compile and execute the code, contribute to readability.  
There exist a lot of different approaches for each of those tasks.  
There exist a lot of different approaches for each of those tasks.  
Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Techniques like Code refactoring can enhance readability.  
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.  
However, because an assembly language is little more than a different notation for a machine language, two machines with different instruction sets also have different assembly languages.  
To produce machine code, the source code must either be compiled or transpiled.  
Normally the first step in debugging is to attempt to reproduce the problem.  
Ideally, the programming language best suited for the task at hand will be selected.  
The choice of language used is subject to many considerations, such as company policy, suitability to task, availability of third-party packages, or individual preference.