Transpiling on the other hand, takes the source-code from a high-level programming language and converts it into bytecode.  
Normally the first step in debugging is to attempt to reproduce the problem.  
Compilers harnessed the power of computers to make programming easier by allowing programmers to specify calculations by entering a formula using infix notation.  
By the late 1960s, data storage devices and computer terminals became inexpensive enough that programs could be created by typing directly into the computers.  
As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices.  
Also, specific user environment and usage history can make it difficult to reproduce the problem.  
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.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
For example, when a bug in a compiler can make it crash when parsing some large source file, a simplification of the test case that results in only few lines from the original source file can be sufficient to reproduce the same crash.  
As early as the 9th century, a programmable music sequencer was invented by the Persian Banu Musa brothers, who described an automated mechanical flute player in the Book of Ingenious Devices.  
There exist a lot of different approaches for each of those tasks.  
The purpose of programming is to find a sequence of instructions that will automate the performance of a task (which can be as complex as an operating system) on a computer, often for solving a given problem.  
This is interpreted into machine code.  
However, Charles Babbage had already written his first program for the Analytical Engine in 1837.  
 These compiled languages allow the programmer to write programs in terms that are syntactically richer, and more capable of abstracting the code, making it easy to target varying machine instruction sets via compilation declarations and heuristics.