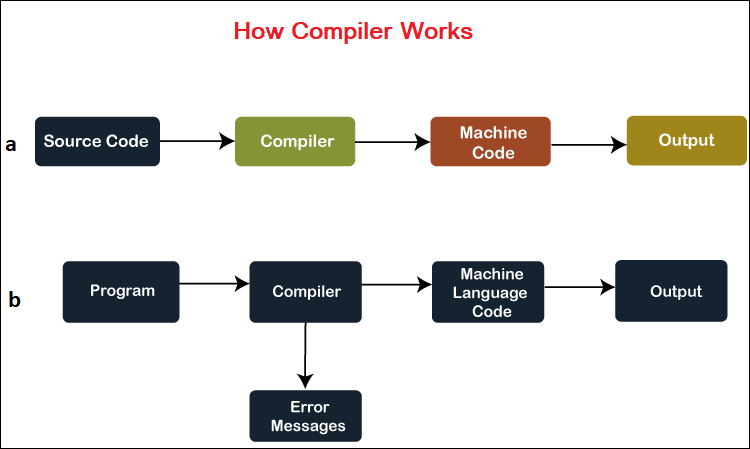
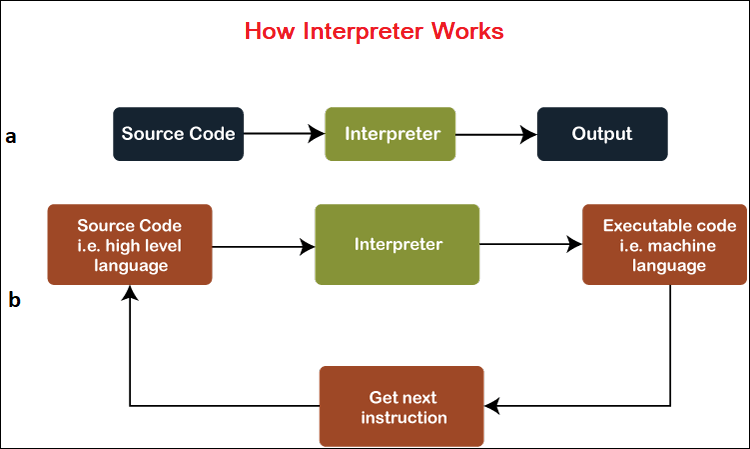
***Assignment for Day 1: Difference between Compiler & Interpreter***

***Definition of Compiler: -*** compiler is a computer software that translates source code written in a high-level language (e.g., C++) into a set of machine-language instructions that can be understood by a digital computer's CPU. Compilers are very large programs, with error-checking and other abilities.



***Definition of Interpreter: -*** An interpreter is a computer program that can analyse and executes instructions written high level language line by line. Interpreters enable other programs to run on a computer or server. They process program code at run time, checking the code for errors line by line.



***Difference Between the Compiler and Interpreter***

|  |  |  |
| --- | --- | --- |
| Analysis | The entire program is analysed in a compiler. | Line by line of the program is analysed in an interpreter. |
| Machine Code | Stores machine code in the disk storage. | Machine code is not stored anywhere. |
| Execution | The execution of the program happens only after the entire program is compiled. | The execution of the program takes place after every line is evaluated and hence the error is raised line by line if any. |
| Run Time | Compiled program runs faster | Interpreted program runs slower. |
| Generation | The compilation gives an output program that runs independently from the source file. | The interpretation does not give any output program and is thus evaluated on every execution. |
| Optimization | The compiler reads the entire program and searches multiple times for a time-saving execution. | No rigorous optimization takes place as code is evaluated line by line |
| Error and error execution | All the errors are shown at the end of the compilation and the program cannot be run until the error is resolved | Displays the errors from line to line. The program runs till the error is found and proceeds further on resolving. |
| Input | The compiler takes in the entire program for analysis. | The interpreter takes in lines of code for analysis. |
| Output | The compiler gives intermediate code forms or object code | The interpreter does not generate any intermediate code forms. |
| **Memory requirement** | A compiled program is generated into an intermediate object code, and it further required linking. So, there is a requirement for more memory. | An interpreted program does not generate an intermediate code. So, there is no requirement for extra memory. |
| **Later Execution** | It does not require source code for later execution. | It requires source code for later execution. |
| **Advantage** | As the source code is already converted into machine code, the code execution time becomes short. | As the source code is interpreted line-by-line, error detection and correction become easy. |
| **Disadvantage** | If you want to change your program for any reason, either by error or logical changes, you can do it only by going back to your source code. | Interpreted programs can run on only those computers which have the same interpreter. |
| Programming languages | C, C++, C#, Java are compiler-based programming languages | PHP, PERL, Ruby are interpreter-based programming languages. |