What is difference between Compiler and Interpreter?

Note: Please upload assignment in PDF format.

Aspect	**Compiler**	**Interpreter**
Execution	Translates the entire source code into machine code before execution.	Translates and executes the source code line-by-line or statement-by-statement.
Speed	Generally, it is faster execution after compilation because the code is directly translated to machine code.	Slower execution compared to compiled code, as translation occurs during execution.
Error Detection	Errors are detected and must be corrected after the entire program is compiled.	Errors are detected and must be corrected line-by-line during execution.
Memory Consumption	Typically, it requires more memory initially to store the executable code.	Uses less memory as it directly executes the instructions without creating an executable file.
Development Cycle	Longer development cycle due to the compile-link-execute steps.	Shorter development cycle, as code can be executed directly, making it ideal for rapid testing and debugging.
Portability	The compiled code is platform-specific. Separate compilations are required for different platforms.	Interpreted code is more portable, as the same code can run on any machine with a compatible interpreter.
Examples	C, C++, Rust, and Swift are typically compiled languages.	Python, JavaScript, and Ruby are examples of languages that are commonly interpreted.