Overview

**What are the roles of the compiler?**

**What are the advantages of compiled languages?**

**List all the different programming paradigms.**

**What are the various ways we can implement a compiler?**

**How does compilation work?**

**How does interpretation work?**

**What are the pros and cons of interpretation?**

**What is a hybrid implementation system?**

**List all the steps in the compilation process.**

**What does the linker/loader do?**

**What does the front-end module of a compiler do?**

**What does the back-end module of a compiler do?**

Translator (source code to target program – machine/byte code), validator (check syntax and semantics of code), and optimizer (time and space).

Efficiency, convenience (reduce low-level complexity), complexity (support for object-oriented programming), and retargetability (one source language to multiple target languages - platforms).

Imperative, functional, logical, and object-oriented programming.

Compilation, interpretation, hybrid.

Compile source code once into machine code and execute multiple times (may need to port source code for different machines).

Translate source code directly and generate executable code.

Creates highly portable code but interpretation is typically slower in execution and must be done every time.

Compile code from source to intermediate. Just in time interpreter to execute into machine code (still compile once, execute everywhere).

Preprocessing, compiling, assembling, linking.

Creates a single executable from different modules.

Focuses on language-specific issues, analysis, and is systematic (using tools).

Machine-specific and focus on generating and optimizing assembly code.