

In alphabetical order by first name, assign someone to each of the following roles.

Manager: keeps the group on task. Makes sure that everyone gets a chance to participate: if someone hasn't spoken up, asks them what they think. Rotates who speaks first for each question.

Reader: reads each question out loud, when asked by the Manager to do so. The reader can choose to skip over long-winded details, but should definitely read all of the main portions of the questions out loud.

Recorder: writes in the answers that the group agrees on for each question.

Presenter: reports answers back to the main group when we come back together. Each group won't report back every question; I'll pick one or two groups to report for each question, and the Presenter will speak for the group.

(If 3 people, reader/presenter are same person.)

Fill in your names here:

- Manager: _____
- Reader: _____
- Recorder: _____
- Presenter: _____

1. Explain the difference between execution speed of a compiler and execution speed of a compiled program. Also explain which one is generally considered to be more important with regards to an end user, if a program is compiled and then distributed.

2. Suppose I had both an interpreter and a compiler for Scheme, and I asked you to compare their speeds of translation. Why would this be difficult to do?

3. A compiler is usually used to translate from a high-level language to a lower-level one. Why? Why not the other way around?

4. Why does a program typically execute more slowly if it is running through an interpreter, than if it had been compiled?

Typically a compiler generates low level code w/ hopefully runs quickly. Interpreting is running through layers in between

5. Why can you typically start running a program that you've just modified more quickly with an interpreter rather than a compiler?

A compiler translates the code before you can run it.
An interpreter starts running right away

6. It is often said that interpreters process a program one line at a time, translating each line and then running it. While an interpreter could be designed to work that way in some very limited circumstances, this is in general not true. Why is it less than accurate to say that an interpreter is a system that translates and runs code line-by-line?

7. Here is a ~~paraphrase of a statement~~ that was found on a blog a few years ago: “One of the biggest advantages of compiled languages over interpreted languages is their execution speed. For example, a program written in C typically runs considerably faster than an equivalent program written in Python.” Though the second sentence of this quote may be true in practice, there’s something fundamentally problematic with the boldfaced sentence above. Explain.

Compiled vs
interpreted
is about the
implementation.

This is a nonsense distinction.

8. Here is an interaction I just had with DeepSeek.

Dave: Is C a compiled language, or an interpreted language?

DeepSeek: C is a compiled language. This means that C code is translated into machine code by a compiler before it is executed. The compilation process converts the human-readable source code into binary instructions that the computer’s processor can directly execute. This is different from interpreted languages, where the code is executed line-by-line by an interpreter at runtime without a separate compilation step.

I then tried ChatGPT.

Dave: Is C a compiled language, or an interpreted language?

ChatGPT: C is a compiled language. This means that C code is written in a human-readable format, then translated into machine code by a compiler (such as GCC, Clang, or MSVC) before it can be executed. The compiled machine code runs directly on the hardware, making C programs fast and efficient.

Unlike interpreted languages (such as Python or JavaScript), C does not require an interpreter at runtime. Instead, once the program is compiled, it can be executed without further translation.

C: clang, gcc,
dozens of others
(most are compilers)

The most popular Python
implementation is a compiler/
interpreter hybrid.

PyPy
is a
JIT
compiler.

Those responses are both incorrect, in a number of ways. (I admit that my question was something of an intentional set up.) Explain what the mistakes are in the responses and why they are incorrect. Just circle or underline the problematic text, and write in the margins what the problems are.

9. Here’s another interaction between me and ChatGPT. Again, I set it up with a (mis)leading question, but it nonetheless could have been a real question from someone hoping to learn. The response was long, but I selected some highlights.

Dave: Is it true that a compiler generates code that is larger in size than an interpreter? *meaningless*

ChatGPT: ~~Not necessarily.~~ The size of the generated code depends on multiple factors, including optimization techniques, the complexity of the program, and the efficiency of the compiler or interpreter...

Compiled code:

- A compiler translates the entire source code into machine code before execution, often applying optimizations to improve performance.

Interpreted code:

- An interpreter does not generate a separate machine code file; instead, it executes the source code line by line.
- Some interpreted languages (like Python) use bytecode (e.g., .pyc files) for execution, which can still take up space.

Conclusion:

- If you’re comparing compiled binaries vs. interpreted source code, compiled binaries may be larger.
- However, interpreted languages require an interpreter at runtime, which can make the overall system heavier.
- Compilers often generate optimized code that can be more efficient in space and execution time than interpreted code.

ChatGPT then asked if I wanted an example, so I said yes. It ran an experiment (apparently) compiling a C and Python “Hello world” program. Lots of details appeared that I’ve cut here, but it ultimately came to these conclusions:

Comparison:

Feature	Compiled C (hello)	Interpreted Python (hello.py)
File Size	Larger (~10-20 KB)	Smaller (~25 bytes)
Interpreter Needed?	No (standalone)	Yes (Python ~30 MB)
Execution Speed	Faster (machine code)	Slower (interpreted)

Conclusion

- **Compiled C produces a larger binary**, but it runs independently without an interpreter.
- **Interpreted Python has smaller source/bytecode files**, but the Python interpreter is required for execution, adding overhead.

So, while compiled code is larger **per file**, an interpreted language **needs an entire runtime environment**, which can be much larger overall.

Identify all the problems with the above. Circle or underline problematic text, then explain in the margins what the problems are.

10. Many modern language implementations use a hybrid approach: they use a compiler to translate to an intermediate language (often called bytecode, or portable intermediate code), and then use either an interpreter or another compiler to operate on the bytecode. Why has this approach become so popular? What advantages does it offer?

11. Just-in-time (JIT) compilers are an innovation that have become very popular. What advantages do they offer over both interpreters and traditional compilers?