**Interpreted Languages**

Interpreters run through a program line by line and execute each command. Here, if the author decides he wants to use a different kind of olive oil, he could scratch the old one out and add the new one. Your translator friend can then convey that change to you as it happens.

Interpreted languages were once significantly slower than compiled languages. But, with the development of [just-in-time compilation](https://guide.freecodecamp.org/computer-science/just-in-time-compilation), that gap is shrinking.

Examples of common interpreted languages are PHP, Ruby, Python, and JavaScript.

**A Small Caveat**

Most programming languages can have both compiled and interpreted implementations – the language itself is not necessarily compiled or interpreted. However, for simplicity’s sake, they’re typically referred to as such.

Python, for example, can be executed as either a compiled program or as an interpreted language in interactive mode. On the other hand, most command line tools, CLIs, and shells can theoretically be classified as interpreted languages.

**Advantages and disadvantages**

**Advantages of compiled languages**

Programs that are compiled into native machine code tend to be faster than interpreted code. This is because the process of translating code at run time adds to the overhead, and can cause the program to be slower overall.

**Disadvantages of compiled languages**

The most notable disadvantages are:

* Additional time needed to complete the entire compilation step before testing
* Platform dependence of the generated binary code

**Advantages of interpreted languages**

Interpreted languages tend to be more flexible, and often offer features like dynamic typing and smaller program size. Also, because interpreters execute the source program code themselves, the code itself is platform independent.

**Disadvantages of interpreted languages**

The most notable disadvantage is typical execution speed compared to compiled languages.