**Scripting Languages:**

Scripting languages are not compiled to machine code by the user (python, perl, shell, etc.). Rather, another program (called the interpreter, runs the program and simulates its behavior)

A scripting language is used to write code that is going to target a software system. It's going to automate operations on that software system. The script is going to be a sequence of instructions to the target software system

In the other hand, the target software system of an script may compile the code or interpret it. Is up to the software system.

**Programming Languages:**

Some programming languages that are not scripting (C, C++, Haskell, and other 'compiled' languages), are compiled to machine code, and is subsequently run.

A programming language targets the computing system, which can be a real or virtual machine. The instructions are executed by the machine.

Of course, a real machine understands only binary code so you need to compile the code of a programming language. But this is a consequence of targeting a machine instead of a program.

**To Understand:**

If we say that the real difference is whether it is compiled or not, then we have a problem because when Javascript runs in V8 is compiled and when it runs in Rhino is not.

This line is getting more and more blurry since compilation can be so fast with modern hardware and modern compilation techniques. For instance, V8, the JavaScript engine in Google Chrome and used a lot outside of the browser as well, actually compiles the JavaScript code on the fly into machine code, rather than interpreting it. (In fact, V8's an optimizing two-phase compiler.)

Also note that **whether a language is a "scripting" language or not can be more about the environment than the language**. There's no reason you can't write a C interpreter and use it as a scripting language (and people have). There's also no reason you can't compile JavaScript to machine code and store that in an executable file (and people have). The language Ruby is a good example of this: The original implementation was entirely interpreted (a "scripting" language), but there are now multiple compilers for it.

Some examples of "scripting" languages (e.g., languages that are *traditionally* used without an explicit compilation step):

* Lua
* JavaScript
* VBScript and VBA
* Perl

And a small smattering of ones *traditionally* used with an explicit compilation step:

* C
* C++
* D
* Java *(but note that Java is compiled to bytecode, which is then interpreted and/or recompiled at runtime)*
* Pascal

...and then you have things like Python that sit in both camps: Python is widely used without a compilation step, but the main implementation (CPython) does that by compiling to bytecode on-the-fly and then running the bytecode in a VM, and it *can* write that bytecode out to files (.pyc, .pyo) for use without recompiling.