You can find both a object-oriented and function-based solution in `exercise_2_solution_oo.py` and `exercise_2_solution_fn.py` respectively. For the object-oriented version, I added a method to the class called `reset_to_factory` that does the job. In the functional version, I added a function.

The funny thing is that the difference between these two solutions is minimal. If you look at the function/method itself, they're exactly the same, except that in the object-oriented version the argument is called `self` and in the functional version it's called `laptop`. And in the main function, we either call the method with the dot syntax (in the object-oriented version), or we pass the object as an argument (in the functional version).

The syntax around classes and object is syntactical sugar. You can achieve the same thing with functions. In fact, even functions are syntactical sugar for lower-level coroutines. And if you go even lower than that, you'll end up at the basic Von Neumann computer architecture of the CPU that manipulates memory. That something is syntactical sugar doesn't mean that it isn't useful though. Sometimes classes are useful because you can structure information with them and group methods together with that information. Sometimes, functions are easier because they lead to shorter code and are often easier to test, especially if they're pure functions.