Alright, Let's start this section by looking at the characteristics of good unit tests:

The first thing I want you to remember is that unit tests are

First-class citizens. So they are as important, if not more than the production code. And that means all the best practices we have learned about writing clean, and maintainable code have applied to your test methods as well. More specifically, each test should have a single responsibility, and should ideally be less than ten lines of code. In fact, one of the reasons that people fail with unit testing is because their tests are messy.

They have large and fat test methods that are really unmaintainable.

When these tests break, they spend so much time on debugging.

So as you are writing tests, you need to keep them clean, and refactor if necessary. I'll show you several examples of these later in this course. Another characteristic of good unit tests, is they should not have any logic. So you should not have any conditional statements, loops and other kinds of logic in your tests. Why? Because when you write logic in unit (?) test, it is possible that you will make a mistake. So, your test will end up having a bug, and it will fail. When a test fails,

first you think there's something wrong with the production code.

So you spend 20 minutes just to figure out that there is a bug in your test. So no logic in your tests! You simply call a method

and make an assertion. Also, each test should be written and executed as if it's the only test in the world. So your tests methods should not call each other and they should not assume any state created by another test. And finally, your test should not be too specific. Or too general.

If they are too general, they may not give you much confidence that your production code is working. Again, we'll look at examples of this shortly.