At the beginning of this section, I told you the characteristics of good unit tests. One characteristic that I didn't tell you about earlier, is that your unit tests should be trustworthy. I didn't tell you about this earlier, because I want to show you with a real example. What do I mean by a trustworthy test? A trustworthy test is the kind of test we can rely on.

So if the test passes, you know that your code is working, and if it fails, you know that there is something wrong with your code. But how can we write trustworthy tests? Well, there are two ways, one way is to use test-driven development, or TDD. With test-driven development, we start by writing a failing test, then we write enough production code, to make that test pass. So if the test passes that means you have written the right production code to make that test pass.

And if the test fails, that means that something was wrong with the production code. So, using TDD is one way to write trustworthy tests. But as I told you before, TDD can get complex in some real world scenarios. So far we have been writing tests after the production code. This approach is a little bit risky, because you may test the wrong thing. But if you test the wrong thing, your test may pass, but your production code

might have a bug, so this is an example of untrustworthy tests. I'm going to simulate this scenario in this lecture, and then show you this technique to write trustworthy tests, even if you write your tests after the production code,

So, back in our math class, I'm going to create a bug in this add method. Instead of returning the sum of A and B.

I'm going to return 0. So, back to our math tests, you currently have a test with the add method, I want to disable the ignore attribute so we run this test, we give this method one and two and we expect three.

But our method is returning zero, so the test should fail. Let's verify this. So command and T, and command and R.

Okay, our test method failed, you can see we expect a three, but we got zero. Beautiful. So, this is a trustworthy test, because it is testing the right thing. However, when you write your tests, after the production code, it is possible you may make a mistake in the implementation of that test. In other words, your test might have a bug. You can't test your test, right?

So, let me simulate this scenario, and then show you the solution

to make sure your test is trustworthy. I'm going to comment out this line, and rewrite my assertion like this. AssertThat\_math,as our private field, Is not null. So you know that our math object is not null. So this test should always pass.

Now this is an exaggerated example, of course in a real world

scenario, you're not going to write an assertion like this. But what I'm pointing out here, is that when you write your tests after the production code, it is possible that you may make a mistake in the implementation of your test.

Now this test is not testing the right thing. It's not testing

the result of the add method. So when you run this, it passes. It's not a trustworthy test. Our production code has a bug, but our test is passing. So you might ask, how can we prevent this scenario from happening when writing tests after the production code? Here's the technique that I'm going to share with you. You go to your production code, and make a simple change on the line that is supposed to make that line pass.

So, I'm going to delete this line temporarily, imagine this is the implementation of our ad method. So this line, is supposed to make our test pass. I'm going to comment out this line, make a small change, and return 1. Now we run our tests again.

So, back to this test, I'm going to run it again, the test still passing, I created a bug in the production code, I made a change, so instead of returning 0, I return 1, and the test is still passing. That means the test is not testing the right thing. I can return anything here, like int.Max value, and this test will still pass.

Not a trustworthy test so the test has a bug. Have a look here, okay, the assertion is wrong, so I'm going to delete this, and bring back this original assertion.

Now we run this, our test failed. So this is a trustworthy test, it's testing the right thing.

In this case, this testing that results is equal to three. So, we go back to our production code, and fix the issue.

Return A + B. So, here is what I want you to take away. When you write your tests after the production code, run your test if it passes, then go in the production code, and make a small change in the line that is supposed to make the test pass. Create a bug, return a different value, maybe comment out that line, if you change the line that is supposed to make the test pass, and then your test still passes that means that test is not testing the right thing. Because if you modify that line, and create a bug, the test should fail, right?

So, make sure to write trustworthy tests that give you value. So when

they pass you know your code is working, and when they fail you know that there is probably something wrong with the code.