Now look at the implementation of these tests methods here. Each test method looks like its the only test in the world. We're not calling one test in another test. And each test is using a new instance of the math class. This is extremely important. In each test, you want to start with a fresh, clean state. So, you shouldn't create a private field here, like private math\_math.

And then reuse it across different tests. Because it is possible that one test may leave some state, in this math object, and that state will leak into another test and impact the result of that test. So each test should be executed using a clean fresh state. However, I don't like the fact that at the beginning of every test method that we are initializing this math object.

It looks a little bit redundant, but in this case it's not a big deal, because it's only a single line of code. But in the real world when you're working with more complex applications, it is possible that you're arranged part might be a few lines of code, made two, three, four lines of code. You don't want to repeat all those four lines in every test method. So let me show you a cleaner way, to rewrite these tests. In NUnit we have two special attributes. SetUp and TearDown.

You can create a method here, and decorate it with a SetUp attribute and then NUnit test runner will call that method before running each test. And this is a great opportunity for us to initialize our math object. Similarly, if we create a method and decorate it with the TearDown attribute, NUnit test runner will call that method after each test.

Now in this course we're not going to use a TearDown attribute, because this is often used with integration tests, because in your integration tests, you may create some data in your database, and then you want to do cleanup after each test. That's where we use the TearDown attribute, so in this lecture, let me show you how to use the SetUp attribute to rewrite these tests, and make them cleaner.

I'm going to create a public void method, we can call it anything, the name doesn't matter but by convention we use SetUp, now, we apply the SetUp attribute, and here we can initialize our math object to a new instance of the math class. So even though I've defined a private field here, we'll be using this in different test, but I'm reinitializing this to a new instance before each test.

Now in this, you can remove the first line of every test. And, of course we need to reference our private field here, similarly, in our second test, our third test, and the last test. And you can see each test, is only two lines of code. So, use the set up method to initialize the object you're going to test.