When you get started with automated testing a word that you hear often is test-driven development or TDD.

Test-driven development, or TDD, also called test first, is an approach to build software. With TDD, you write your test before writing the application or production code. The first time I heard this I asked

how on earth is that even possible, how can we test a code you haven't written, chances are you have the same question. So, let's see how TDD works. You start by writing a failing test. This test should fail because you don't have any application code that would make it pass, right?

Then you will write the simplest application code that will make this test pass. The absolute simplest, you don't want to over engineer here, you don't want to design a class diagram with a zillion classes and

methods. Use the simplest implementation that would make the test pass. Then, refactor your code if necessary.

These three steps are the foundation of TDD.

You repeat these three steps over and over until you build a complete feature. Now what is so special about TDD? Well, first, is that

source code will be testable right from the get go. You don't have to make any changes to your code to make it testable. Second, is that every line of your production code is fully covered by tests. Which means you can refactor and deploy with confidence, and third, is that it often results in a simpler implementation.

When you start with a big class diagram. Chances are you are over engineering and making the solution more complex. If you write enough code to make all the tests pass, and that solution works, there is no reason to write more code. The fact that all your tests passed, means you have fulfilled all the business requirements. So, unless there is a new

requirement, you don't need to write new code. And if there is a new requirement, you start with failing test. So this is TDD.

In TDD, we write our tests first, and that's why we call this approach test driven development. So our development is driven by our tests. In contrast to TDD, or test first, we have code first, which is what you have been doing so far. You start with your application code, and then you write tests. Which approach is better? It depends who you ask.

In theory, TDD is more promising because of the benefits I told you. But in practice, sometimes it can get really complex and it may slow you down. If that's the case, it's better to switch to the code first approach, and write your tests after. In this course, our focus will be on the code first approach, so we can master the fundamentals of testing.

Once you master the fundamentals, then you will be ready to start

your test first journey. TDD really requires a separate course, and I'm not going to talk about it throughout this course. But if you're interested, potentially in the future, I can create a course on test driven development.