So, now you know about the different kinds of automated tests. Unit tests, integration tests, and end-to-end tests. But what kind of tests should you write in your application? Well, all of them. This is what we call the test pyramid. This pyramid argues that most of your tests should be in the category of unit tests, because these tests are easy to write, and they execute quickly. But since they don't give you much confidence about the health of your application, you should have a bunch of integration tests that test the integration of your application code with its external dependencies. These tests provide many advantages of end-to-end tests, but without the complexities of dealing with the user interface.

And finally, you should write very few end-to-end tests for the key functions of the application, but you should not test the edge cases with

these end-to-end tests. You only test the happy path, and leave the edge cases to unit tests. Now this pyramid, it's just a guideline. It's not a hard and fast rule you need to follow in every application. The actual ratio between your unit integration and end-to-end tests, really depends on your project.

Unit tests are great for quickly testing the logic of conditional statements and loops. If you have methods with complex logic and calculation, you should test them with your unit tests.

Earlier in the section, I showed you an example of a calculate

function. Unit tests are ideal for testing these functions, because you can quickly test all the execution paths of these functions in less than a second. Manually testing these functions through the user interface takes a significant amount of time and is prone to errors. However, not every application has a complex logic, and functions like our calculate function.

You might have an application that simply reads some data from or

writes it to a database. In that case you may need more integration

tests than unit tests. So in summary, this test pyramid gives you three recommendations. First, is to favor unit tests over UI or end-to-end

tests. Because these unit tests are the fastest to run, and cheapest to write, and they're very precise. So we can pinpoint exactly where something fails. They give you rapid feedback. Second is to cover the unit test gaps, with integration tests. And finally, use end-to-end tests sparingly, only for the key functions of your application. The right balance is different for each project and each team. At the end of the day you need to use your own judgement to determine what kind of tests you need to write for different parts of your applications.