Part 11: Test Driven Development

Bookmark this page

Advanced Testing

Part 11: Test Driven Development

No discussion of testing is complete without Test Driven Development (TDD).

TDD is a practice that extends the idea of testing; instead of writing some cool code and then writing a test to make sure it never gets broken, first you conceive of what you want your software to accomplish, you write tests that will pass once you've written your software, and then you write the software, then you run the tests to make sure that you have satisfied their behavioral requirements.

To begin with, go back to the TDD video and watch a few more minutes of the talk. You're welcome to watch the whole thing; the guy is a good presenter!

Although proponents of TDD describe it as an essential practice, TDD is not currently "the standard" practice of test writing. For better or worse, most software engineers write their tests *after* they write their code, like we did in our videos above, not *before*. Even so, TDD is a valuable tool to explore: trying it in one of your own projects can be an enlightening experience and improve your coding of tests in general.

Conclusion

Brian Kernighan, computer science, and co-author of the famed K&R *C Programming Language* wrote that, "Controlling complexity is the essence of computer programming."

Today, you might be focused on the *syntax* of computer programming: how to turn individual bits of logic into control statements in the Python language. And you might be focused on how to build those statements up into programs of a few hundred lines of code.

With practice, you'll soon become comfortable with the line-by-line and method-by-method of constructing Python programming. You'll encounter the challenge of maintaining and contributing to large, old code bases. In this lesson, we introduced you to *some* of the tools that engineers use to manage the complexity of large software projects