Integration Testing

Integration testing exercises two or more parts of an application at once, including the interactions between the parts, to determine if they function as intended. This type of testing identifies defects in the interfaces between disparate parts of a codebase as they invoke each other and pass data between themselves.

How is integration testing different from unit testing?

While unit testing is used to find bugs in individual functions, integration testing tests the system as a whole. These two approaches should be used together, instead of doing just one approach over the other. When a system is comprehensively unit tested, it makes integration testing far easier because many of the bugs in the individual components will have already been found and fixed.

As a codebase scales up, both unit and integration testing allow developers to quickly identify breaking changes in their code. Many times these breaking changes are unintended and wouldn't be known about until later in the development cycle, potentially when an end user discovers the issue while using the software. Automated unit and integration tests greatly increase the likelihood that bugs will be found as soon as possible during development so they can be addressed immediately.

Integration testing resources

- Integration testing with Context Managers gives an example of a system that needs integration tests and shows how context managers can be used to address the problem.
- Pytest has a page on integration good practices that you'll likely want to follow when testing your application.
- Integration testing, or how to sleep well at night explains what integration tests are and gives an example. The example is coded in Java but still relevant when you're learning about integration testing.
- What is an integration test exactly? is an awesome Stack Exchange thread that defines the differences in testing approaches like unit tests versus integration and other tests. There is also some practical advice like "It's not important what you call it, but what it does" which as a pragmatic programmer I am keen to agree on.

- Consistent Selenium Testing in Python gives a spectacular code-driven walkthrough for setting up Selenium along with SauceLabs for continuous browser-based testing.
- Where do our flaky tests come from? presents Google's data on where their integration tests fail and how the tools you use can sometimes lead to higher incidents of failed tests than other testing tools.
- Unleash the test army covers the author's first impressions of using Hypothesis for testing the properties of a system under test.

What else do you want to learn about testing?

How do I create unit tests?

Can I automate testing and deployments for my app?

What code metrics should I be aware of?

Full Stack Python

Full Stack Python is an open book that explains concepts in plain language and provides helpful resources for those topics.

Updates via Twitter & Facebook.

Chapters

Page Statuses

...or view the full table of contents.