25.3. unittest — Unit testing framework

New in version 2.1.

(If you are already familiar with the basic concepts of testing, you might want to skip to the list of assert methods.)

The Python unit testing framework, sometimes referred to as "PyUnit," is a Python language version of JUnit, by Kent Beck and Erich Gamma. JUnit is, in turn, a Java version of Kent's Smalltalk testing framework. Each is the de facto standard unit testing framework for its respective language.

`unittest` supports test automation, sharing of setup and shutdown code for tests, aggregation of tests into collections, and independence of the tests from the reporting framework. The `unittest` module provides classes that make it easy to support these qualities for a set of tests.

To achieve this, `unittest` supports some important concepts:

test fixture
: A *test fixture* represents the preparation needed to perform one or more tests, and any associate cleanup actions. This may involve, for example, creating temporary or proxy databases, directories, or starting a server process.

test case
: A *test case* is the smallest unit of testing. It checks for a specific response to a particular set of inputs. `unittest` provides a base class, `TestCase`, which may be used to create new test cases.

test suite
: A *test suite* is a collection of test cases, test suites, or both. It is used to aggregate tests that should be executed together.

test runner
: A *test runner* is a component which orchestrates the execution of tests and provides the outcome to the user. The runner may use a graphical interface, a textual interface, or return a special value to indicate the results of executing the tests.

The test case and test fixture concepts are supported through the `TestCase` and `FunctionTestCase` classes; the former should be used when creating new tests, and the latter can be used when integrating existing test code with a `unittest`-driven framework. When building test fixtures using `TestCase`, the `setUp()` and `tearDown()` methods can be overridden to provide initialization and cleanup for the fixture. With `FunctionTestCase`, existing functions can be passed to the constructor for these purposes. When the test is run, the fixture initialization is run first; if it succeeds, the cleanup method is run after the test has been executed, regardless of the outcome of the test. Each instance of the `TestCase` will only be used to run a single test method, so a new fixture is created for each test.

Test suites are implemented by the `TestSuite` class. This class allows individual tests and test suites to be aggregated; when the suite is executed, all tests added directly to the suite and in "child" test suites are run.