**Example of a simple Unit Test**:

Suppose, we have the Calculator class containing basic functions like **add**, **subtract**, **divide**, **multiply**, and **clear**.

|  |
| --- |
| class Calculator {  int add (int value1, int value2) {  return value1 + value2;  }  int subtract (int value1, int value2) {  return value1 – value2;  }  int multiply (int value1, int value2) {  return value1 \* value2;  }  float divide (int value1, int value2) {  return value1 / value2;  }  int clear () {  return 0;  }  } |

We will create the following *Unit Test* to test these methods of Calculator class.

|  |
| --- |
| class CalculatorUnitTest {  @Test  void testCalculator () {  Calculator myCalculator = new Calculator ();  int value1 = 10; int value2 = 5;  assertEquals (myCalculator.add(value1, value2), 15);  assertEquals (myCalculator.subtract(value1, value2), 5);  assertEquals (myCalculator.multiply(value1, value2), 50);  assertEquals (myCalculator.divide(value1, value2), 2.0);  assertEquals (myCalculator.clear(), 0);  }  } |
|  |

AssertEquals is important in Unit Test. It takes two arguments: 1) The actual implementation of the method (will return actuals result) that we want to test, and 2) The expected result of the method under test. Assert assumes the actual and expected results are equal, and any flaws will throw an exception for AssertionError.

Note that this is demo code. The syntax of the Unit Test may vary for different languages. Some useful links for Unit Test is given below for your references.

**Some useful links of Unit Test**:

PHP:

* PHPUnit: <https://phpunit.readthedocs.io/en/9.2/writing-tests-for-phpunit.html>
  + Code Coverage Analysis: <https://phpunit.readthedocs.io/en/9.2/code-coverage-analysis.html>
* SimpleTest: <http://simpletest.sourceforge.net/en/first_test_tutorial.html>

JS:

* JEST: <https://jestjs.io/docs/en/tutorial-react>
  + Code coverage in Jest: <https://www.valentinog.com/blog/jest/>
* ReactJS: <https://reactjs.org/docs/testing-recipes.html>
* Node.js: <https://developer.ibm.com/technologies/node-js/tutorials/learn-nodejs-unit-testing-in-nodejs/>
* AngularJS: <https://docs.angularjs.org/guide/unit-testing>

Django:

* <https://docs.djangoproject.com/en/3.0/topics/testing/overview/>
* <https://django-testing-docs.readthedocs.io/en/latest/basic_unittests.html>
* Measuring Coverage: <https://django-testing-docs.readthedocs.io/en/latest/coverage.html>

Python:

* unittest: <https://docs.python-guide.org/writing/tests/>
* Python3: <https://docs.python.org/3/library/unittest.html>
* Python2: <https://docs.python.org/2/library/unittest.html>
* Coverage.py: <https://coverage.readthedocs.io/en/coverage-5.1/>

Java:

* <https://github.com/junit-team/junit4/wiki/Getting-started>
* <http://tutorials.jenkov.com/java-unit-testing/simple-test.html>

C#:

* <https://docs.microsoft.com/en-us/visualstudio/test/getting-started-with-unit-testing?view=vs-2019>
* <https://docs.microsoft.com/en-us/visualstudio/test/unit-test-basics?view=vs-2019>

Some IDE/Tool/Plugin:

* IntelliJ IDEA: <https://www.jetbrains.com/help/idea/configuring-code-coverage-measurement.html?keymap=secondary_intellij_idea_classic>
* PyCharm: <https://www.jetbrains.com/help/pycharm/configuring-code-coverage-measurement.html>
* Visual Studio: <https://docs.microsoft.com/en-us/visualstudio/test/using-code-coverage-to-determine-how-much-code-is-being-tested?view=vs-2019>
* JaCoCo Plugin: <https://docs.gradle.org/current/userguide/jacoco_plugin.html>
* More Tools: <https://www.softwaretestinghelp.com/code-coverage-tools/>