<https://docs.python.org/3/tutorial/stdlib.html>

10.11. Quality Control

One approach for developing high quality software is to write tests for each function as it is developed and to run those tests frequently during the development process.

The [doctest](https://docs.python.org/3/library/doctest.html" \l "module-doctest" \o "doctest: Test pieces of code within docstrings.) module provides a tool for scanning a module and validating tests embedded in a program’s docstrings. Test construction is as simple as cutting-and-pasting a typical call along with its results into the docstring. This improves the documentation by providing the user with an example and it allows the doctest module to make sure the code remains true to the documentation:

**def** average(values):

*"""Computes the arithmetic mean of a list of numbers.*

*>>> print(average([20, 30, 70]))*

*40.0*

*"""*

**return** sum(values) / len(values)

**import** **doctest**

doctest.testmod() *# automatically validate the embedded tests*

The [unittest](https://docs.python.org/3/library/unittest.html" \l "module-unittest" \o "unittest: Unit testing framework for Python.) module is not as effortless as the [doctest](https://docs.python.org/3/library/doctest.html" \l "module-doctest" \o "doctest: Test pieces of code within docstrings.) module, but it allows a more comprehensive set of tests to be maintained in a separate file:

**import** **unittest**

**class** **TestStatisticalFunctions**(unittest.TestCase):

**def** test\_average(self):

self.assertEqual(average([20, 30, 70]), 40.0)

self.assertEqual(round(average([1, 5, 7]), 1), 4.3)

**with** self.assertRaises(**ZeroDivisionError**):

average([])

**with** self.assertRaises(**TypeError**):

average(20, 30, 70)

unittest.main() *# Calling from the command line invokes all tests*