M05 Unit Test Assignment

Parrish, Scott

SDEV 220

11/26/23

In working through the Unit Testing assignment, we used 3 different methods of testing. In the first, assertions were used to manually check results from the built in sum() function. These tests, which tested the function with a list and tuple (formatted intentionally to be incorrect) produced the appropriate results, but this method could quickly become tedious the more complex the objects to be tested became.

$ python3 test\_sum\_2.py

Traceback (most recent call last):

File "/home/scott/Documents/School/sdev\_220/playground/sdev\_220HW/ParrishScott\_M05\_UnitTests/test\_sum\_2.py", line 11, in <module>

test\_sum\_tuple()

File "/home/scott/Documents/School/sdev\_220/playground/sdev\_220HW/ParrishScott\_M05\_UnitTests/test\_sum\_2.py", line 6, in test\_sum\_tuple

assert sum((1, 2, 2)) == 6, "Should be 6"

AssertionError: Should be 6

The assignment then switched to using the built-in unittest library which requires a test class to be built to do the testing. The class structure allowed the same tests to be conducted using class methods for the assertions instead of the assert statement. Both of these tests can be run using alternative modules, pytest and nose2. Unlike simple assertions these libraries can be used to write tests for larger projects without becoming unwieldy.

$ python3 -m unittest\_sum

.F

======================================================================

FAIL: test\_sum\_tuple (\_\_main\_\_.TestSum.test\_sum\_tuple)

----------------------------------------------------------------------

Traceback (most recent call last):

File "/home/scott/Documents/School/sdev\_220/playground/sdev\_220HW/ParrishScott\_M05\_UnitTests/unittest\_sum.py", line 10, in test\_sum\_tuple

self.assertEqual(sum((1, 2, 2)), 6, "Should be 6")

AssertionError: 5 != 6 : Should be 6

----------------------------------------------------------------------

Ran 2 tests in 0.000s

FAILED (failures=1)

The assignment then had me switch to creating a custom sum() function which was imported and then tested using the unittest library. This simple sum() function was tested for list input as with the previous examples, and then it was tested by summing a list of fractions which, like some of the previous tests, was intentionally incorrect to produce a failure. By using directory structure and more detailed commands to conduct the test, tests could be initiated on multiple files with multiple tests using commands like the “discover” option which allows python unittest to search the current directory (or a specified directory) for any test files.

(.venv) ± python3 -m unittest discover

F.

======================================================================

FAIL: test\_list\_fraction (test.TestSum.test\_list\_fraction)

Test that it can sum a list of fractions

----------------------------------------------------------------------

Traceback (most recent call last):

File "/home/scott/Documents/School/sdev\_220HW/ParrishScott\_M05\_UnitTests/UnitTestEx/test.py", line 23, in test\_list\_fraction

self.assertEqual(result, 1)

AssertionError: Fraction(9, 10) != 1

----------------------------------------------------------------------

Ran 2 tests in 0.001s

FAILED (failures=1)