Software Unit Testing

This task involved developing a Scrabble Score using Test Driven Development and an automated unit testing tool. I have used Python programming language to develop the unit tests and also work on the Scrabble Score program. The automated unit testing tool used in this task is PyUnit. The purpose of the unit testing is to identify bugs early in the development stage of an application where the bugs are less recurrent and less expensive to fix. PyUnit is a python unit testing framework well known for creating unit testing programs using Python. This framework uses the name “unittest” which is also the module name when importing the module to a python testing file.

Test-Driven Development (TDD) refers to a software development methodology or a programming technique that involves evaluating coding with tests to validate what code will do [1]. In this process, test code for each function in the scrabble score file is written to validate the functionality. The output of the function is validated with an expected value and if the function fails to give the appropriate response, new code is written to make sure that the particular function passes the test.

There are several test cases for the functions in the main file. The testing process begins by importing the scrabble module that has the code for the scrabble score. In addition to that, import also the automated testing module named “unittest”. This framework requires a class to be created and initialized with the methods and parameters of the TestCase class in the unittest module. Afterward, create functions that will be used to validate each test case in the main file functions.

There are 8 tests in total in the test python file. We begin by testing if the program allows the user 15 seconds to enter a word. This is done by expecting the value returned by the main function to be true if the user managed to type in a word within 15 seconds. The first time the function fails because there is no value returned by the main function. To ensure that the test passes, the Timer class is imported from the threading module to assist in developing this functionality of closing the input prompt after 15 seconds. The main function returns True if a word was typed within 15 seconds after the prompt and False if no word was typed after 15 seconds elapsed.

The next test case involved checking whether the word is an actual English word. As a result, the test function expected a True assertion if the word is an actual English word and False if it is not. At first, the function fails because there is no value returned by the check\_word function in the scrabble module. This test passes after importing the enchant module that assists in validating strings as correct English words. A user's input is passed in the check word function and returns True if the word exists and False if the input is not an actual English word.

The main function test also validates that the input from the user contains the right number of letters as prompted. This is through an if-else statement in the scrabble module main function that asserts that the length of the input matches the length required in the prompt to continue chasing if the word exists in the English dictionary or not. In the case where the user decided to enter a number instead of a word made up of actual letters, a score of zero was given because scores have not been allocated characters that are not in the alphabet. This is achieved by adding an else statement in the calculate letter score function that adds a score zero to the total score.

To ensure that the code adds up the values properly, test cases of known words have been written. A word is passed in the scrabble module’s calculate\_word\_score that returns a score for the particular word. The returned score is asserted to ensure that it is equal to the manually calculated score. This test is repeated where an uppercase word and also mixed upper and lowercase letters that make up a correct English word are passed to the calculate\_word\_score function to ensure that the correct score is returned. A word in uppercase and lowercase should have the same score as asserted in this test.

For this task, I have learned how to correctly execute unit tests for functions in the main file and use Test Driven Development approaches to ensure that a program works as expected. I have also understood that TDD saves a lot of time and resources because errors are captured before the development phase allowing them to be mitigated easily.

Github link: https://github.com/nazzy243/PRT582-assignment-1.git

References

|  |  |
| --- | --- |
| [1] | J. Unadkat, "BDD vs TDD vs ATDD : Key Differences," *BrowserStack ,* 4 May 2021. |
| [2] | O. Mikhalchuk, "The importance of unit testing, or how bugs found in time will save you money," *Forté Group,* 10 December 2020. |
| [3] | T. Hamilton, "What is Test Driven Development (TDD)? Tutorial with Example," *Guru99,* 8 October 2021. |