**Software Unit Testing Report**

**Jinyi Teng s342221**

**Git link:** **https://github.com/5202tt/TDD-Scrabble-Score.git**

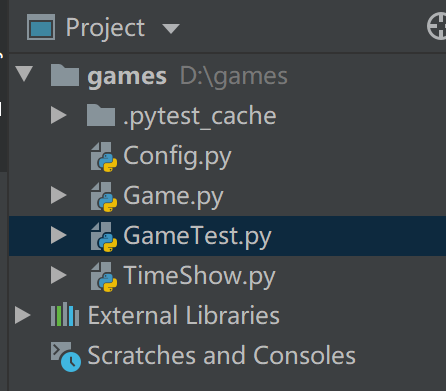
Introduction

Trying to do Scrabble Score game source code and unit test code, using TDD and automated unit testing tools to create the program.

The programming language used is Python and the unit test tool is Pytest.

The game source code file is: Game.py, Timeshow.py config.py

The unit test file is: GameTest.py



Process (step1 to step 6)

Step 1: Write test case code

The test case method is:

@Pytest. mark.parametrize("words, expect,length",

[

("a", 1, 1),

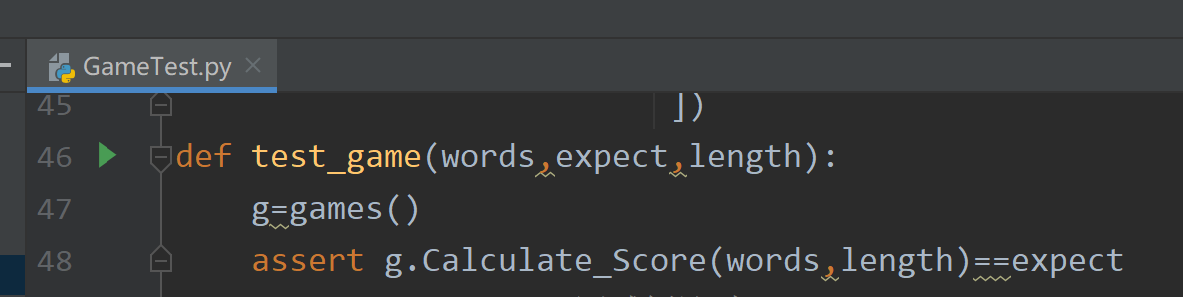
("cabbage", 14, 7)

def test\_game(words,expect,length):

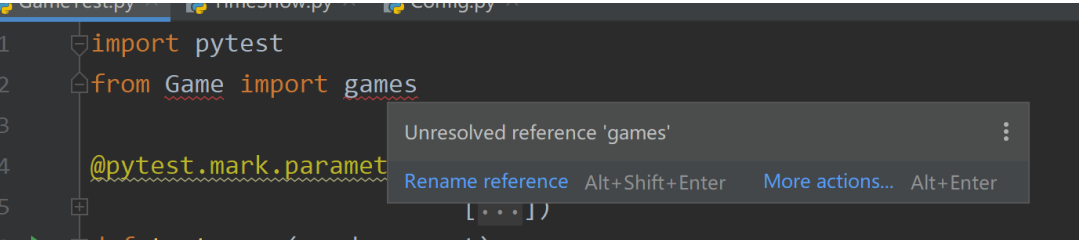
g=games ()

assert g.Calculate\_Score(words,length)==expect

Where ("cabbage", 14, 7) is the parameter of the test case, "cabbage" represents the input test of the game source code, 14 is the expected output result of the game source program, and 7 is the length of the input string. The function of assert is to assert and automatically compare the actual test results with the expected test results. As shown below:



Step 2: Run the test case. It cannot run at this time because the game source code has not been implemented.



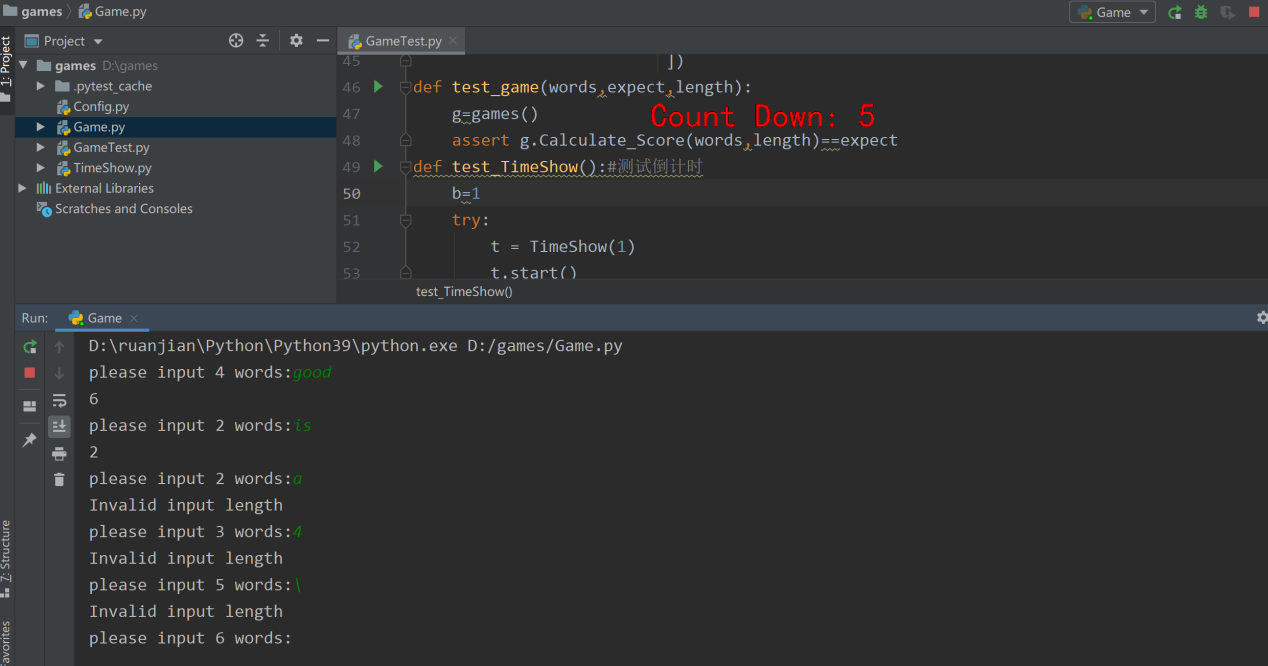
Step 3: write the implementation code of the game.

The file to realize the game score calculation function is: game.py

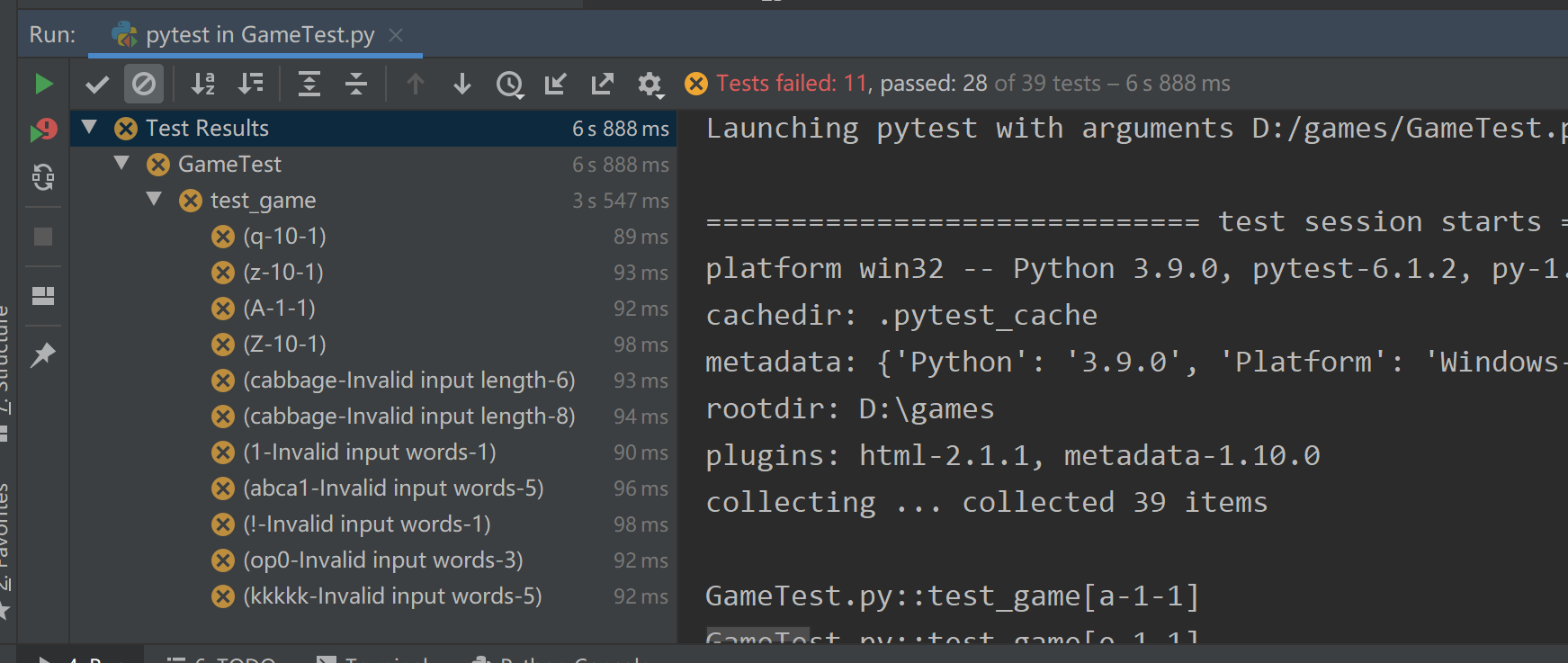
The file to realize the countdown display function is timeshow.py

Intermediate variable: config.py

Execute and run the game source code:



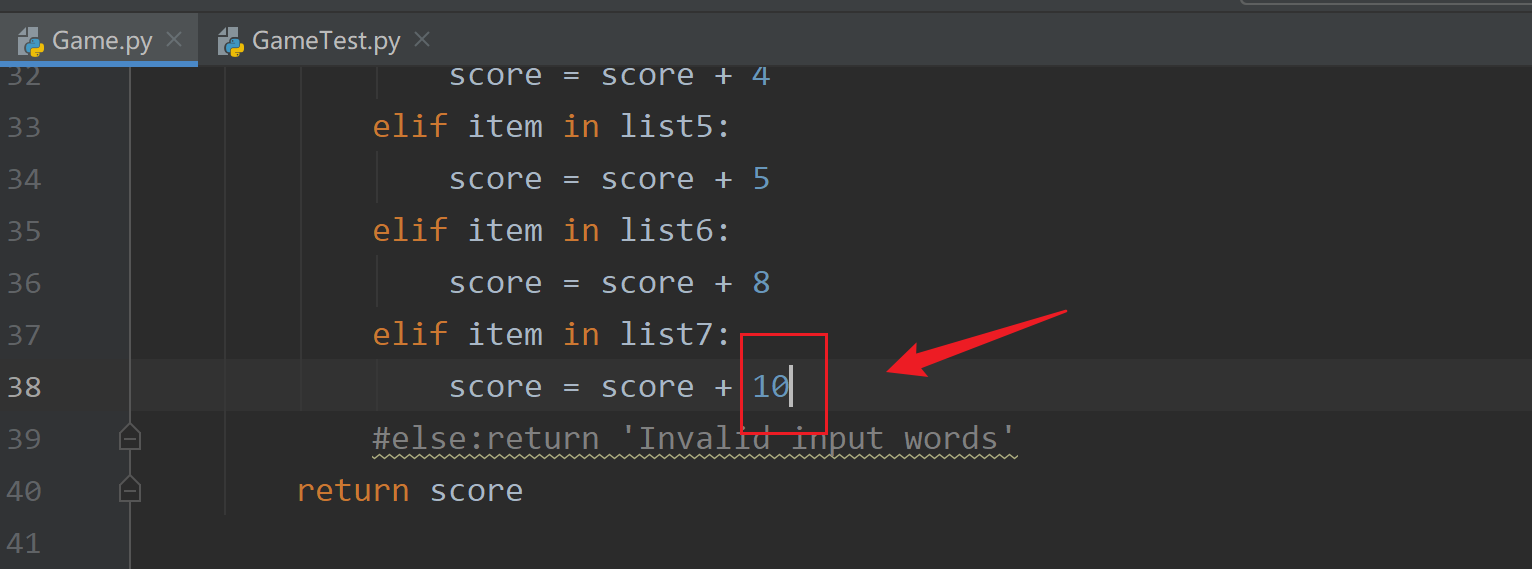
Step 4: Run the test code, and a total of 11 test cases fail.



Step 5 Reason for failure and code modification

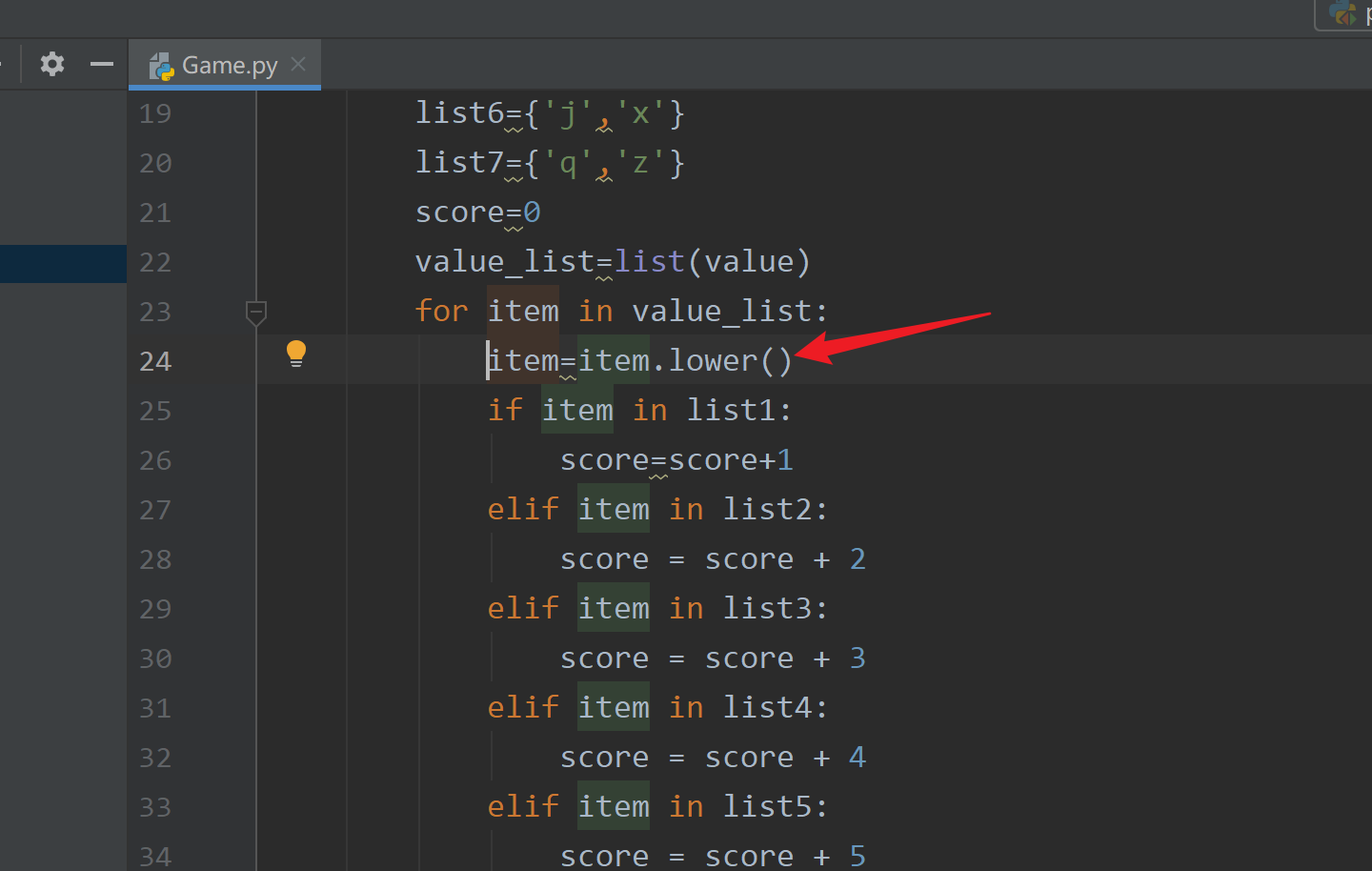
5.1

For the first and second failed cases, the reason is that the score of Q and Z letters is calculated incorrectly by the source program. The solution is to modify the score of Q and Z letter to 10, as shown in the following figure:



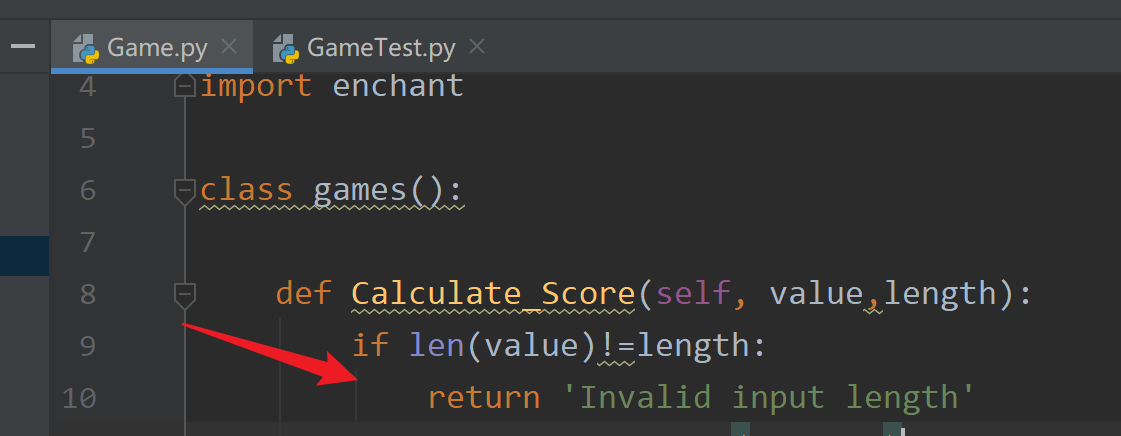
5.2

The reason for the failure of test cases (a-1-1) and (z-10-a) is that the source program is not compatible with the input of uppercase letters. The modification scheme is to add a line of code for the function of converting lowercase to uppercase (add a line of code after line 23 of the game.py file: item = item. Lower ())



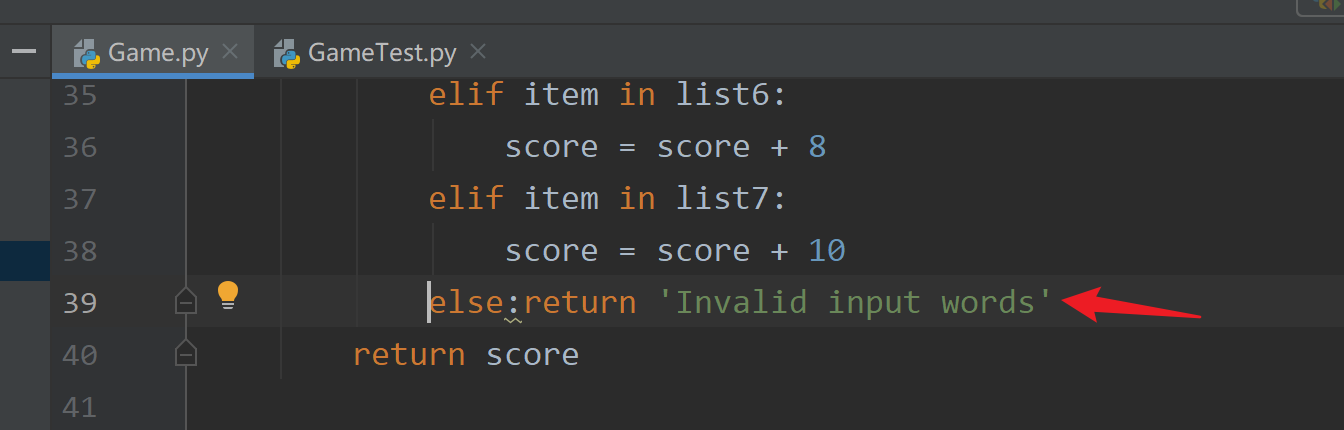
5.3

The reason why the execution of use cases ("cabbage", "invalid input length", 6) and ("cabbage", "invalid input length", 8) fails is that the source program does not judge the length of the input word. The modification scheme is adding the code to judge the word length, as shown in the figure below.



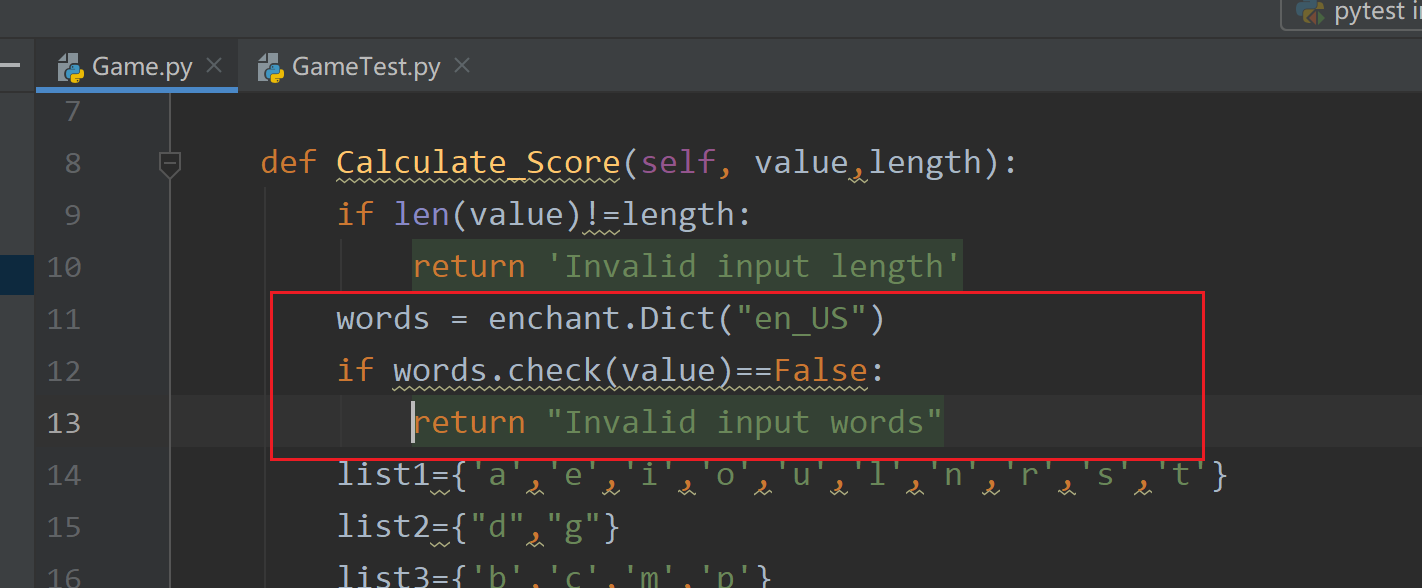
5.4

The reason for the failure of cases 7 to 10 is that the source program fails to exclude non alphabetic input. The solution is to add non alphabetic judgment, as shown in the following figure

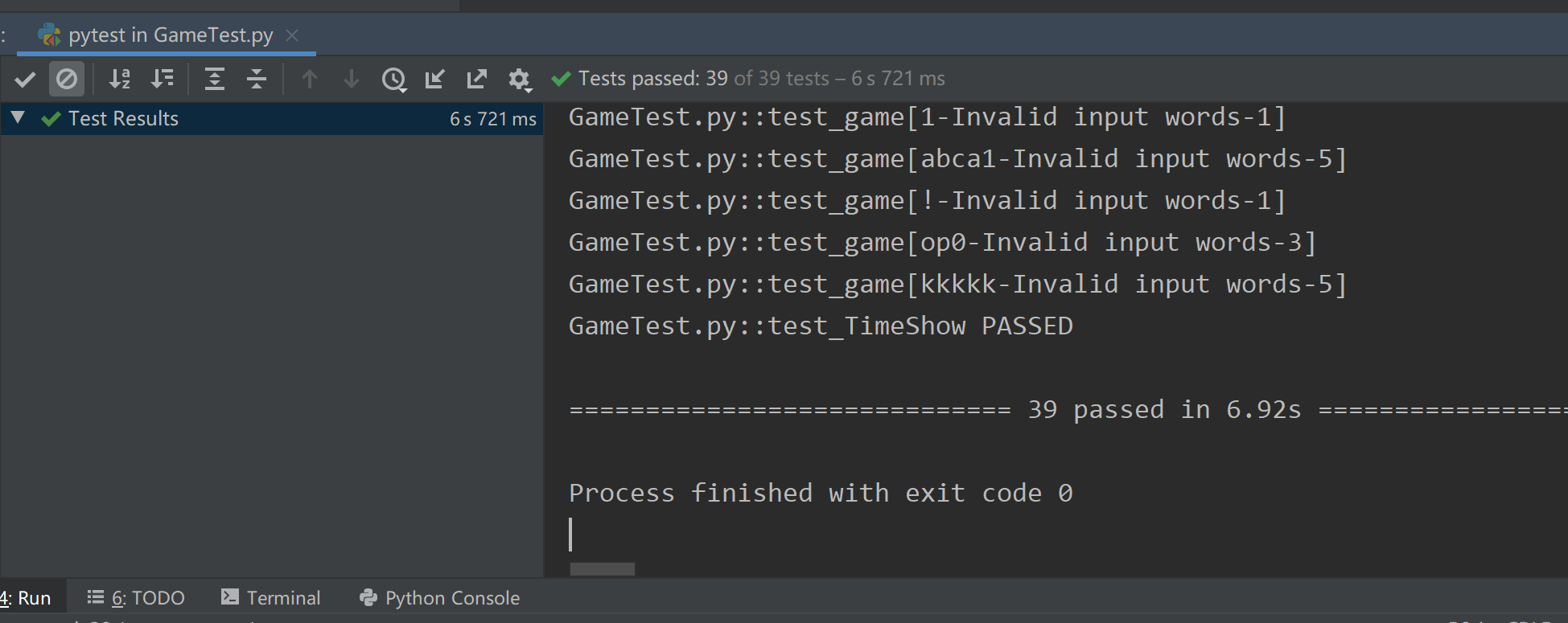


5.5

The reason for the test failure of this use case ("KKKK", "invalid input words", 5) is that the source program fails to judge whether it constitutes a word. The solution is to add the judgment of whether it constitutes a word, as shown in the following figure:



Step 6: Rerun the test cases and all tests pass.



Conclusion

Summary of experience: software testing is the most effective means to ensure software quality. At the same time, high-quality test cases and test efficiency are also effective methods to ensure software quality. In this unit test, it is found that there are many test cases in the early stage. If a use case writes a function, the amount of test code will be very large, and the later test case maintenance will be very difficult. After the parametric function (@ pytest. Mark. Parameter) is introduced, you only need to write a test case method to classify and manage test cases, which is very suitable for maintenance of use cases (such as addition, deletion, and modification of use cases). Only by continuously participating in the test practice can we better improve our test technology. git link: https://github.com/5202tt/TDD-Scrabble-Score.git