CHARLES DARWIN UNIVERSITY AUSTRALIA

Software Engineering: Process and Tools

PRT 582 Assignment 1:

Software Unit Testing Report

Name: Yunzhu Wang

Student ID: S346832

Email: S346832@students.cdu.edu.au

Table of Contents

Introduction

Objectives

The program I wrote is a simple scrabble game. Players will be asked to input some words of a certain length within 15 seconds. Then the program will generate a total score according to their performance.

The required length I set is 3-10 length, which will be generated automatically and randomly.

The total score depends on the words they input, how many times they play within 15 seconds, and how long they use to input the word:

- The letters that make up different words have different scores.
- Players can get more scores if they can play more times under same condition.
- The score will be higher if less time is used to enter the same right length of the word.

When users input wrong content, they will get clear feedback for their wrong input:

- If input includes numbers or other symbols, "Your input is not alphabet!" will be shown.
- If the length of input does not meet the required length, the program will show "Length of your Word is not same with the requirement!".
- If the input is not in the dictionary, it will show "Your input is not an English Word!".
- If the input correctly meets the requirement, the program will generate a score and show how much time being used and left. Then players can play another round until 15 seconds run out.

Requirements

This report will focus on unit testing, so I will only address the main functional requirements here.

ID	Requirements
FR-001	The program should assign values to different letters correctly.
FR-002	The program should be able to add up the values correctly for a given word.
FR-003	The program should assign same value to upper-or lower-case input letters.
FR-004	The program should ask users to input a word of 3-10 length randomly.
FR-005	The program should be able to check whether users enter the right length of input and give clear feedback if not.
FR-006	The program should be able to check whether users enter alphabet and give clear feedback if not.
FR-007	The program should be able to check whether users enter a valid word from a dictionary and give clear feedback if not.
FR-008	The program should show a 15-seconds timer to countdown when users play the scrabble game.
FR-009	The program should give higher score if less time is used to enter the word as required and generate an overall score.

Requirements Form

Programming Language

The programming language I chose is Python. As a new learner of software engineering and programming, Python is easier to read, learn and write compared to other programming languages. It has English-like syntax and vast libraries support. And there is another unit (HIT 137) in this semester, which can teach us some fundamental knowledge of Python. Besides, I can also find lots of courses, materials, and videos to learn online.

Automated Unit Testing Tool

According to the programming language I chose, I used the PyUnit as the automated unit testing tool for this program, which is an easy way to create unit testing programs and Unit Tests with Python. I will explain how I used it in detail and show the screenshot below.

Process

Explanations

During the Agile software development process, I used the Test-driven development (TDD) technique:

- Firstly, I subdivided the general requirements provided and refined each specific requirement for the scrabble game, as is shown in the Requirements Form above.
- After that, I wrote testing code according to each specific requirement and used PyUnit to do testing. At this stage, the unit tests ran and failed because no coding was implemented yet.
- Then, I tried to write the minimum amount of code to pass the test.
- At last, I refactored the new code to acceptable standards.
- And repeated these steps for all the requirement I wrote.

Unit Test Case & PyUnit Screenshot

To better show my workflow by using TDD to write this program, examples of the unit test cases and PyUnit screenshots will be shown below.

Note:

- Some requirements are closely related, so I put them together for same module to write the test cases.
- The blue Test Case ID will show testing screenshot.

- Usually the initial code (before refactoring) and testing code are in two different files, but I put them together for easier screenshot.

Unit Test cases for requirements FR-001, FR-002, FR-003:

Project Name: Scrabble game Functionality Module Name: Values Calculating Functionality

Created By: Yunzhu Wang Created Date: 24-09-2021

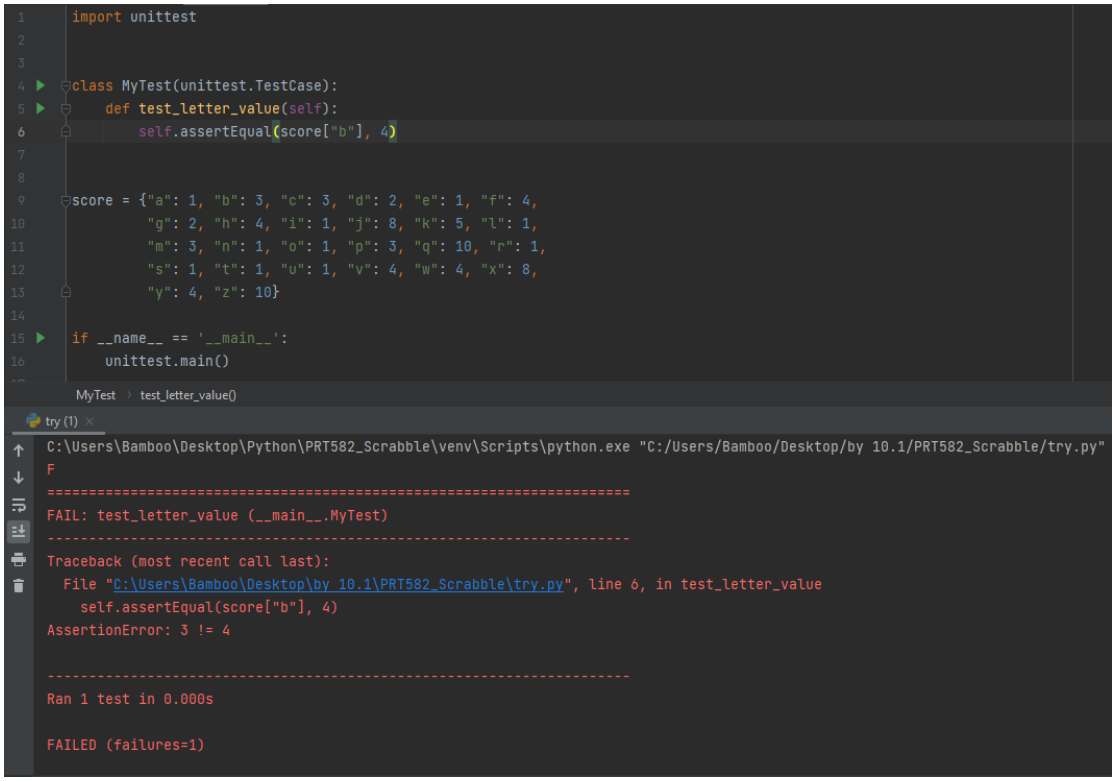
Executed by: Yunzhu Wang Executed Date: 24-09-2021

Requirements	Test Case ID	Description	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status
FR-001	TC_FR-001_R02	Use **assertEqual(a,b)** Method to verify the functionality of assigning correct value for each letter.	1.Define a method called "test_letter_value"; 2. Use assertEqual(a,b) to check whether the value of letters equal with the value as required.		Letter: b Assert Value: 3	OK	As expected	Pass
	TC_FR-001_R26				Letter: z Assert Value:10	OK	As expected	Pass
	TC_FR-001_F02			TC_FR-001_R01 to TC_FR-001_R26 tests all pass	Letter: b Assert Value: 4	F. Assertion Error	As expected	Pass
FR-002	TC_FR-002_R01	Assert values to verify the functionality of adding up the values correctly for a given word.	1. Define a method called "test_word_value"; 2. assert values to check specific words.	TC_FR-001 tests all pass	Word: cabbage Assert Value: 14	OK	As expected	Pass
	TC_FR-002_F02				Word: cabbage Assert Value: 13	F.	As expected	Pass
FR-003	TR_FR-003_R01	Verify the functionality of assigning same value to upper-or lower-case input letters.	1. Under the "test_word_value"method, inputting words mixed with upper-case and lower-case or all upper-case, check the value.	TC_FR-002 tests all pass	Word: Cabbage Assert Value: 14	OK	Not as expected	Fail
	TR_FR-003_R02				Word: Cabbage Assert Value: 14	OK	As expected	Pass
	TR_FR-003_R03				Word: CABBAGE Assert Value: 14	OK	As expected	Pass

```
import unittest


class MyTest(unittest.TestCase):
    def test_letter_value(self):
        self.assertEqual(score["z"], 10)


score = {"a": 1, "b": 3, "c": 3, "d": 2, "e": 1, "f": 4,
         "g": 2, "h": 4, "i": 1, "j": 8, "k": 5, "l": 1,
         "m": 3, "n": 1, "o": 1, "p": 3, "q": 10, "r": 1,
         "s": 1, "t": 1, "u": 1, "v": 4, "w": 4, "x": 8,
         "y": 4, "z": 10}

if __name__ == '__main__':
    unittest.main()
```

```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-001_R26



Screenshot for TC_FR-001_F02

Screenshot for TC_FR-002_R01

```
import unittest


class MyTest(unittest.TestCase):
    def test_word_value(self):
        assert 13 == scrabble_score("cabbage")


score = {...}


def scrabble_score(word):
    points = 0
    for letter in word:
        points += score[letter]
    return points


if __name__ == '__main__':
    unittest.main()
```

scrabble_score()

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
F
======================================================================
FAIL: test_word_value (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 6, in test_word_value
    assert 13 == scrabble_score("cabbage")
AssertionError

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

Screenshot for TC_FR-002_F02

```
import unittest


class MyTest(unittest.TestCase):
    def test_word_value(self):
        assert 14 == scrabble_score("Cabbage")


score = {...}


def scrabble_score(word):
    points = 0
    for letter in word:
        points += score[letter]
    return points


if __name__ == '__main__':
    unittest.main()
```

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
E
======================================================================
ERROR: test_word_value (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 6, in test_word_value
    assert 14 == scrabble_score("Cabbage")
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 19, in scrabble_score
    points += score[letter]
KeyError: 'C'

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (errors=1)
```

Screenshot for TC_FR-003_F01

```
import unittest


class MyTest(unittest.TestCase):
    def test_word_value(self):
        assert 14 == scrabble_score("Cabbage")


score = {...}


def scrabble_score(word):
    points = 0
    for letter in word.lower():
        points += score[letter]
    return points


if __name__ == '__main__':
    unittest.main()
```

```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-003_R02

Note:

- The screenshot of TC_FR-003_F01 and TC_FR-003_R02 showing different outcome by using same test data is because I revised the coding by adding “lower-case” function after the test of TC_FR-003_R01 got failed.

Unit Test cases for requirements FR-004, FR-005, FR-006, FR-007:

Project Name: Scrabble game Functionality Module Name: Input Checking Functionality

Created By: Yunzhu Wang Created Date: 26-09-2021

Executed by: Yunzhu Wang Executed Date: 26-09-2021

Requirements	Test Case ID	Description	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status
FR-004	TC_FR-004_R01	Use **assertIn(a,b)** Method to verify the functionality of generating a word of 3-10 length randomly.	1.Define a method called “test_generate_random_letter”; 2. Use assertIn(a,b) to check the random.randint method.		List_length = [3,4,5,6,7,8,9,10]	OK	As expected	Pass
FR-005	TC_FR-005_R01	Use **assertTrue** Method to verify the functionality of checking correct length of input.	1.Define a method called “test_word_length”; 2. Use assertTrue to check whether the length of input is same with the required length.		Word: university Require length: 10	OK	As expected	Pass
	TC_FR-005_F02				Word: university Require length: 9	F.	As expected	Pass
	TC_FR-005_R02				Word: study Require length: 5	OK	As expected	Pass
FR-006	TC_FR-006_R01	Verify the functionality of checking whether users enter alphabet.	1. Define a method called “test_word_alphabet”; 2. Assert different input mixed with numbers and other symbols		Word: university	OK	As expected	Pass
	TC_FR-006_F02				input: uni45@#oty	F.	As expected	Pass
FR-007	TR_FR-007_R01	Verify the functionality of checking whether users enter a valid word from a dictionary	1. Define a method called “test_dictionary”; 2. Assert word from dictionary and fake word.	TR_FR-006 test all pass	Word: cabbage	OK	As expected	Pass
	TR_FR-007_F02				Word: cabagge	F.	As expected	Pass

```python
import unittest
from random import randint


class MyTest(unittest.TestCase):
    def test_generate_random_length(self):
        self.assertIn(char_length, list_length)


# the length of required input word is 3-10, which is generated randomly.
char_length = randint(3, 10)
list_length = [3, 4, 5, 6, 7, 8, 9, 10]

score = {...}


def scrabble_score(word):...


if __name__ == '__main__':
    unittest.main()
```

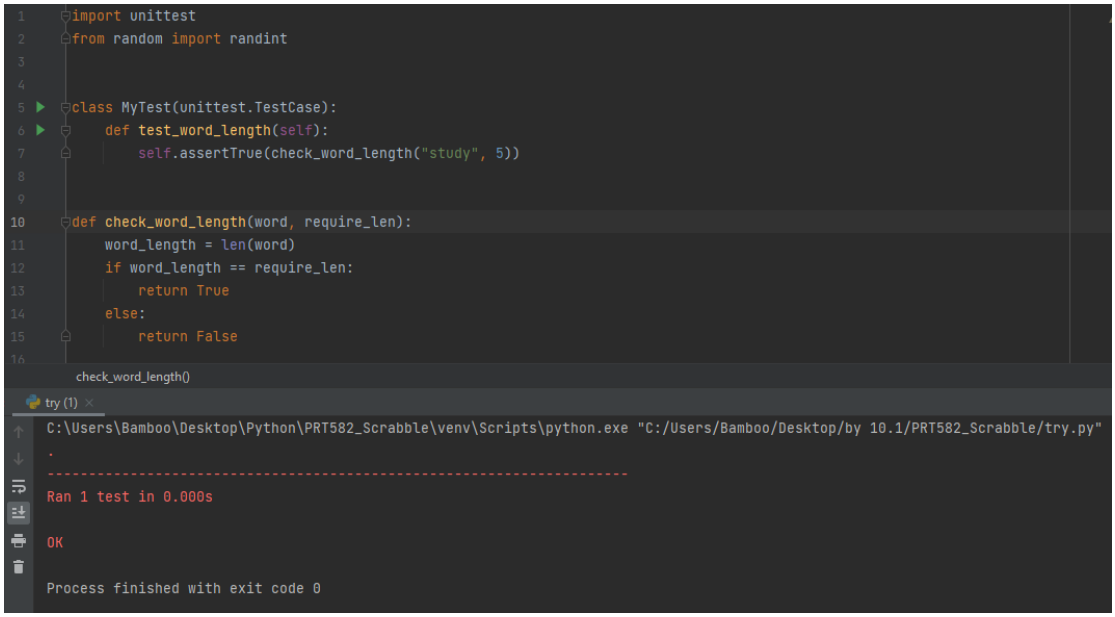
```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-004_R01

```python
import unittest
from random import randint


class MyTest(unittest.TestCase):
    def test_word_length(self):
        self.assertTrue(check_word_length("university", 9))


def check_word_length(word, require_len):
    word_length = len(word)
    if word_length == require_len:
        return True
    else:
        return False
```

```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
F
======================================================================
FAIL: test_word_length (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 7, in test_word_length
    self.assertTrue(check_word_length("university", 9))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)

Process finished with exit code 1
```

Screenshot for TC_FR-005_F02



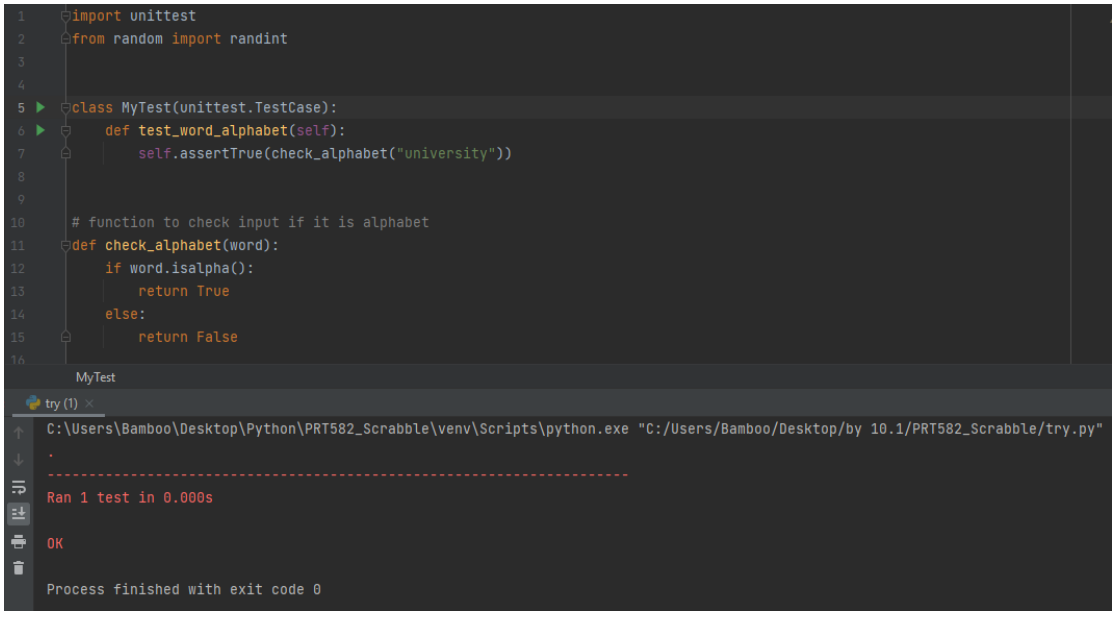
```
import unittest
from random import randint


class MyTest(unittest.TestCase):
    def test_word_length(self):
        self.assertTrue(check_word_length("study", 5))


def check_word_length(word, require_len):
    word_length = len(word)
    if word_length == require_len:
        return True
    else:
        return False
```

check_word_length()

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-005_R02



```
import unittest
from random import randint


class MyTest(unittest.TestCase):
    def test_word_alphabet(self):
        self.assertTrue(check_alphabet("university"))


# function to check input if it is alphabet
def check_alphabet(word):
    if word.isalpha():
        return True
    else:
        return False
```

MyTest

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-006_R01

```python
import unittest
from random import randint


class MyTest(unittest.TestCase):
    def test_word_alphabet(self):
        self.assertTrue(check_alphabet("uni45@#oty"))


# function to check input if it is alphabet
def check_alphabet(word):
    if word.isalpha():
        return True
    else:
        return False
```

check_alphabet() › if word.isalpha()

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
F
======================================================================
FAIL: test_word_alphabet (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 7, in test_word_alphabet
    self.assertTrue(check_alphabet("uni45@#oty"))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 1 test in 0.001s

FAILED (failures=1)
```

Screenshot for TC_FR-006_F02

```python
import unittest
from random import randint
import enchant


class MyTest(unittest.TestCase):
    def test_dictionary(self):
        self.assertTrue(check_dictionary("cabbage"))


# function to check input if it is valid word from dictionary
def check_dictionary(word):
    dictionary = enchant.Dict("en_US")
    if dictionary.check(word):
        return True
    else:
        return False
```

MyTest › test_dictionary()

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.092s

OK

Process finished with exit code 0
```

Screenshot for TR_FR-007_R01

```python
import unittest
from random import randint
import enchant


class MyTest(unittest.TestCase):
    def test_dictionary(self):
        self.assertTrue(check_dictionary("cabagge"))


# function to check input if it is valid word from dictionary
def check_dictionary(word):
    dictionary = enchant.Dict("en_US")
    if dictionary.check(word):
        return True
    else:
        return False
```

MyTest › test_dictionary()

try (1)

```
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
F
======================================================================
FAIL: test_dictionary (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 8, in test_dictionary
    self.assertTrue(check_dictionary("cabagge"))
AssertionError: False is not true

----------------------------------------------------------------------
Ran 1 test in 0.095s

FAILED (failures=1)

Process finished with exit code 1
```

Screenshot for TC_FR-007_F02

Unit Test cases for requirements FR-008, FR-009:

Project Name: Scrabble game Functionality — Module Name: Timer Functionality

Created By: Yunzhu Wang — Created Date: 28-09-2021

Executed by: Yunzhu Wang — Executed Date: 28-09-2021

Requirements	Test Case ID	Description	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status
FR-008	TC_FR-008_R01	Verify the functionality of 15-seconds timer.	1.Define a method called “test_timer”; 2. Use threading to check countdown function.		Timer: 15 seconds	OK	As expected	Pass
FR-009	TC_FR-009_F01	Verify the functionality of giving higher score if less time is used.	1. Define a method called “test_time_score”; 2. Use “assertGreater” to verify that higher score will get if less time is used by putting same word.	TC_FR-008 test pass.	Word: cabbage My_timer:5 My_timer:8	F.	As expected	Pass
	TC_FR-009_R01				Word: cabbage My_timer:5 My_timer:3	OK	As expected	Pass

```python
import unittest
from random import randint
import enchant
from time import *
import threading


class MyTest(unittest.TestCase):
    def test_timer(self):
        countdown_thread = threading.Thread(target=countdown)
        countdown_thread.start()
        self.assertTrue(countdown)


# function to make 15 sec timer
def countdown():
    global my_timer
    my_timer = 15
    for x in range(15):
        if my_timer == 0:
            return True
            break
        my_timer = my_timer - 1
        sleep(1)
```

```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-008_R01

```python
import unittest
from random import randint
import enchant
from time import *
import threading


class MyTest(unittest.TestCase):
    def test_time_score(self):
        self.assertGreater((total_score("cabbage", 5)), (total_score("cabbage", 3)))


# function to give higher score if less time is used.
def total_score(word, my_timer):
    overall_score = 0
    dictionary = enchant.Dict("en_US")
    if word.isalpha() and my_timer > 0:
        if dictionary.check(word):
            overall_score = overall_score + scrabble_score(word)*my_timer
            return overall_score
```

```
MyTest › test_time_score()
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.197s

OK

Process finished with exit code 0
```

Screenshot for TC_FR-009_R01

```python
import unittest
from random import randint
import enchant
from time import *
import threading


class MyTest(unittest.TestCase):
    def test_time_score(self):
        self.assertGreater((total_score("cabbage", 5)), (total_score("cabbage", 8)))


# function to give higher score if less time is used.
def total_score(word, my_timer):
    overall_score = 0
    dictionary = enchant.Dict("en_US")
    if word.isalpha() and my_timer > 0:
        if dictionary.check(word):
            overall_score = overall_score + scrabble_score(word)*my_timer
            return overall_score
```

```
try (1)
C:\Users\Bamboo\Desktop\Python\PRT582_Scrabble\venv\Scripts\python.exe "C:/Users/Bamboo/Desktop/by 10.1/PRT582_Scrabble/try.py"
F
======================================================================
FAIL: test_time_score (__main__.MyTest)
----------------------------------------------------------------------
Traceback (most recent call last):
  File "C:\Users\Bamboo\Desktop\by 10.1\PRT582_Scrabble\try.py", line 10, in test_time_score
    self.assertGreater((total_score("cabbage", 5)), (total_score("cabbage", 8)))
AssertionError: 70 not greater than 112

----------------------------------------------------------------------
Ran 1 test in 0.222s

FAILED (failures=1)

Process finished with exit code 1
```

Screenshot for TC_FR-009_F01

Conclusion

What I have learnt

The most important thing I have learnt is the coding knowledge and TDD technique. I cannot list how many mistakes I have been made during this process of making this program. As a new learner of programming, understanding the theory of TDD is not difficult, but it is hard to really practice it when making my own program. Due to lack of sufficient coding knowledge, it took me a lot of time and huge efforts to search materials and watch videos. Sometimes after searching a lot, I still cannot write proper initial code to meet the requirements and pass the tests, which make me feel terribly upset and desperate.

What went well

However, the good thing is after the early desperate testing stage, it went well when I refactored the real code. I just realized it is the benefit that TDD technique has. In the beginning, as a programmer we could get stuck for writing some code to meet one specific requirement. We need search a lot to solve the problems and pass the test. After that, it will be easier to refactor the real code, sometimes only need to make it tidy up and adjust the sequence as a more logical way. It is also easy to check and revise by using TDD if there is something wrong.

What can be improved

For this program, I think what can be improved is the timer countdown functionality. As the requirement wrote “A 15-seconds timer is shown”, it is better to show all the time when users are playing the game, which can make users feel more nervous as well as excited. The countdown timer I made now can only show when users input something and get an outcome. Even like this, it took me long time to think about the keywords and search online. When I get more coding knowledge, hope I can make this function better.

This is my first coding program. The process is hard, but finally I make it. Although it is a quite simple scrabble game, I feel extremely excited when I play it.

The GitHub link: https://github.com/s346832/Scrabble-game_PRT582