Software Unit Testing Report

PRT582

Thien Phuc Tran - S383410

Introduction

This is an assignment report in which my task is to write a program: Scrabble Score using TDD and automated unit testing tool in Python.

These are the requirements for my program:

1. The numbers are added up correctly for a given word.
2. Upper- and lower-case letters should have the same value.
3. Your program should prompt the user with the right feedback if the user does not enter an alphabet.
4. A 15-second timer is shown. User is asked to input a word of a certain length. The number of alphabets required in the word is randomly generated. The program will check to ensure that the right length of word is entered before generating the score. Score will be higher if less time is used to enter the right length of word.
5. Ensure that the user enters a valid word from a dictionary. The program will not tabulate the score if the word is not a proper word from a dictionary. Prompts will be given asking the user to enter a valid word if the user does not enter a valid word.
6. The game will keep going:
    6.1. Until the player quits the game and display the total score of the player.
    6.2. After 10 rounds and compute the total score of the player

In the assignment, I will use *unittest* in Python to write test cases that meet the requirements. Next, based on that, I will write Scrabble game code and make sure the code runs properly. Then I will use automated unit testing tool to check for errors.

For automated unit testing tool, I use flake8 and Pylint.

- Flake8 “is a command-line utility for enforcing style consistency across Python projects” (Cordasco, 2016).

- Pylint is a static code analyzer for Python 2 or 3. I do not have to run the code for it to analyze. According to Pylint community (n.d.), “It checks for errors, enforces a coding standard, looks for code smells, and can make suggestions about how the code could be refactored.”

Process

TDD

I write test cases as follows:

Requirement 1: The numbers are added up correctly for a given word.

```python
import unittest
from Scrabble_Game import scrabble_score # type: ignore

class TestScrabbleScore(unittest.TestCase):
    # Requirement 1: The numbers are added up correctly for a given word
    def test_single_letter(self):
        self.assertEqual(scrabble_score('A'), 1)
        self.assertEqual(scrabble_score('Z'), 10)

    def test_word(self):
        self.assertEqual(scrabble_score('cabbage'), 14)
        self.assertEqual(scrabble_score('Scrabble'), 14)

    def test_mixed_case(self):
        self.assertEqual(scrabble_score('CaBbAgE'), 14)
```

- test_single_letter():
    - Purpose: To verify that the Scrabble score is correctly calculated for single letters with different values.
- test_word():
    - Purpose: To ensure that the Scrabble score is correctly calculated for a given word.
- test_mixed_case():
    - Purpose: To confirm that the Scrabble score is calculated correctly for words with mixed case, ensuring case insensitivity.

Requirement 2: Upper- and lower-case letters should have the same value

```python
    # Requirement 2: Upper- and lower-case letters should have the same value
    def test_uppercase(self):
        self.assertEqual(scrabble_score('CABBAGE'), 14)

    def test_lowercase(self):
        self.assertEqual(scrabble_score('cabbage'), 14)
```

- test_uppercase():
    - Purpose: To verify that the Scrabble score is correctly calculated for words in uppercase.
- test_lowercase():

- Purpose: To verify that the Scrabble score is correctly calculated for words in lowercase.

- test_mixed_case():
  - Purpose: To confirm that the Scrabble score is calculated correctly for words with mixed case, ensuring case insensitivity.

Requirement 3: Prompt user with the right feedback if user does not enter an alphabet

```
# Requirement 3: Prompt user with the right feedback if user does not enter an alphabet
def test_invalid_input(self):
    self.assertEqual(scrabble_score('123'), 0)
    self.assertEqual(scrabble_score('<>?"'), 0)
    self.assertEqual(scrabble_score(''), 0)
    self.assertEqual(scrabble_score(' '), 0)
```

- test_invalid_input():
  - Purpose: To ensure that invalid inputs (non-alphabetic characters, empty strings, whitespace) are handled correctly and do not contribute to the score.

Requirement 4: Timer functionality and word length check

```
# Requirement 4: Timer functionality
def test_timer_functionality(self):

    import time
    start_time = time.time()
    time.sleep(1)  # Simulate delay
    end_time = time.time()
    self.assertTrue(end_time - start_time >= 1)

def test_word_length_check(self):
    word = 'cabbage'
    word_length = 7
    self.assertEqual(len(word), word_length)

def test_score_based_on_time(self):
    word = 'cabbage'
    score = scrabble_score(word)
    time_taken = 10  # Simulate time taken in seconds
    bonus = max(0, 30 - time_taken)  # Bonus points for faster input
    total_score = score + bonus
    self.assertEqual(total_score, 14 + 20)
```

- test_timer_functionality():
  - Purpose: To verify that the timer functionality works correctly by simulating a delay and checking the elapsed time.

- test_word_length_check():
    - Purpose: To ensure that the word length is checked before generating the score.
- test_score_based_on_time():
    - Purpose: To simulate scoring based on the time taken to input the word, with a bonus for faster input.

Requirement 5: Ensure word is valid and not tabulate score for invalid words

```
# Requirement 5: Ensure word is valid
def test_dictionary_word_validation(self):
    valid_words = ['cabbage', 'scraBBle', 'APPLE']
    invalid_words = ['123', 'azxc1213', '?><+#}']
    for word in valid_words:
        self.assertTrue(word.isalpha())
    for word in invalid_words:
        self.assertFalse(word.isalpha())

def test_no_score_for_invalid_words(self):
    invalid_words = ['890', 'abmnd12@d', '!@#']
    for word in invalid_words:
        self.assertEqual(scrabble_score(word), 0)
```

- test_dictionary_word_validation():
    - Purpose: To ensure that the words entered are valid dictionary words by checking if they contain only alphabetic characters.
- test_no_score_for_invalid_words():
    - Purpose: To ensure that no score is tabulated for invalid words (non-alphabetic characters).

Requirement 6: Game keeps going until player quits or after 10 rounds

```python
# Requirement 6: Game keeps going until player quits or after 10 rounds
def test_game_continues(self):
    rounds = 10
    total_score = 0
    for _ in range(rounds):
        word = 'red'
        score = scrabble_score(word)
        total_score += score
    self.assertEqual(total_score, 4 * rounds)

def test_game_quit(self):
    rounds = 5 # Simulate the player quitting the game
    total_score = 0
    for _ in range(rounds):
        word = 'cabbage'
        score = scrabble_score(word)
        total_score += score
    self.assertEqual(total_score, 14 * rounds)
```

- test_game_continues():
    - Purpose: To ensure that the game can run for multiple rounds and correctly accumulates the total score.
- test_game_quit():
    - Purpose: To simulate the player quitting the game after a few rounds and ensure the total score is correctly calculated up to that point. It is similar to test_game_continues() but I put it there to demonstrate the case when player quit and still show final score

After unit testing, I will write my Scrabble code based on that. Then I will use the automated unit testing tools.

Automated unit testing tools

Flake8: I use this tool to ensure that I follow coding standards, and my code is free of common errors. It mostly focuses on how I present the code for better viewing.

```
PS C:\Users\Phuc> flake8 c:/Users/Phuc/Desktop/Scrabble_Game.py
c:/Users/Phuc/Desktop/Scrabble_Game.py:6:80: E501 line too long (83 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:16:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:26:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:30:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:37:80: E501 line too long (83 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:45:80: E501 line too long (98 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:47:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:52:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:60:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:64:80: E501 line too long (82 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:75:80: E501 line too long (146 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:77:80: E501 line too long (130 > 79 characters)
c:/Users/Phuc/Desktop/Scrabble_Game.py:83:1: E305 expected 2 blank lines after class or function definition, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:84:11: W292 no newline at end of file
PS C:\Users\Phuc>
```

The message in the terminal from flake 8 is formatted as:

file path : **line number** : **column number** : **error code** : **short description**

Now I will describe the errors and how to solve them:

Error E501 indicates that my lines have more than 79 characters since flake8 suggests a maximum of mentioned characters for a line of code (Zenva, 2013). Because there is no requirement for character limitation, we do not need to change the code. To solve this problem, we will define a maximum line length for flake8.

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
```

Here I define the maximum line length for my code as 300 characters. After this the terminal update, deleting error E501 messages.

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
c:/Users/Phuc/Desktop/Scrabble_Game.py:16:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:26:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:30:1: E302 expected 2 blank lines, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:47:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:52:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:60:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:83:1: E305 expected 2 blank lines after class or function definition, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:84:11: W292 no newline at end of file
```

ERROR E302 indicates that two blank lines are expected between functions and classes. Here some part of my code has only 1 bank line instead of 2.

```
c:/Users/Phuc/Desktop/Scrabble_Game.py:16:1: E302 expected 2 blank lines, found 1
    'Q': 10, 'Z': 10
}

# Function to calculate the Scrabble score for a given word
def scrabble_score(word):
    if not word.isalpha():
```

We can see from the message that in line 16, column 1 (based on the format), there is only

1 blank line, which causes the error. (The number on the left column determines the line number).

```
12      Q : 10,  Z': 10
13  }
14
15
16  # Function to calculate the Scrabble score for a given word
17  def scrabble_score(word):
```

By adding another blank line, we solve the error. We also apply the same method to other lines of code stated on the message.

```
c:/Users/Phuc/Desktop/Scrabble_Game.py:30:1: E302 expected 2 blank lines, found 1
```

```
26
27  # Function to generate a random word length between 3 and 10
28  def get_word_length():
29      return random.randint(3, 10)
30
31  # Main function to run the Scrabble game
32  def main():
33      total_score = 0
```

```
26
27  # Function to generate a random word length between 3 and 10
28  def get_word_length():
29      return random.randint(3, 10)
30
31
32  # Main function to run the Scrabble game
33  def main():
34      total_score = 0
```

After solving:

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
c:/Users/Phuc/Desktop/Scrabble_Game.py:50:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:55:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:63:1: W293 blank line contains whitespace
c:/Users/Phuc/Desktop/Scrabble_Game.py:86:1: E305 expected 2 blank lines after class or function definition, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:87:11: W292 no newline at end of file
PS C:\Users\Phuc>
```

ERROR W293 indicates that blank lines should not contain any tabs or spaces.

```
c:/Users/Phuc/Desktop/Scrabble_Game.py:55:1: W293 blank line contains whitespace
```

```
53              print("Invalid input. Please enter a valid word.")
54              continue
55          ..........
56          # Calculate the time taken to enter the word
57          time_taken = end_time - start_time
58
59          # Check if the word was entered within the time limit
```

We can see here that there are ... at where I highlight. They are usually hidden so you must be careful as they are tricky to spot. It depends on your visual code editor. Here I'm using Visual Sudio Code. To solve this problem, delete the ... and error messages will disappear.

```
52          if not word.isalpha():
53              print("Invalid input. Please enter a valid word.")
54              continue
55
56          # Calculate the time taken to enter the word
57          time_taken = end_time - start_time
```

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
c:/Users/Phuc/Desktop/Scrabble_Game.py:86:1: E305 expected 2 blank lines after class or function definition, found 1
c:/Users/Phuc/Desktop/Scrabble_Game.py:87:11: W292 no newline at end of file
```

ERROR E305 indicates that functions and classes should have two blank lines to separate them from other functions and classes. Here I have only 1 blank line instead of 2.

```
c:/Users/Phuc/Desktop/Scrabble_Game.py:86:1: E305 expected 2 blank lines after class or function definition, found 1
83      # Display the total score at the end of the game
84      print(f"Game over! Your total score is {total_score}")
85
86  if __name__ == '__main__':
87      main()
```

We apply the same method from **ERROR E302** by adding another blank line, solving the problem.

```
    # Display the total score at the end of the game
    print(f"Game over! Your total score is {total_score}")


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

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
c:/Users/Phuc/Desktop/Scrabble_Game.py:88:11: W292 no newline at end of file
PS C:\Users\Phuc>
```

ERROR W292 indicates that the code file should end with a newline.

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

Here you can see that there is no newline. We solve this by adding a comment showing that there is a newline ending the code.

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

# This is a new line that ends the file.

```

In this case, if I put the newline on line 89, it will still cause the **ERROR W292** since the editor thinks that it belongs to the function.

```
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
PS C:\Users\Phuc> flake8 --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
PS C:\Users\Phuc>
```

When you fix every error, the terminal will not show anything. The duplication above is the demonstration of that.

Pylint:

```
PS C:\Users\Phuc> pylint c:/Users/Phuc/Desktop/Scrabble_Game.py
************* Module Scrabble_Game
c:\Users\Phuc\Desktop\Scrabble_Game.py:78:0: C0301: Line too long (146/100) (line-too-long)
c:\Users\Phuc\Desktop\Scrabble_Game.py:80:0: C0301: Line too long (130/100) (line-too-long)
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0114: Missing module docstring (missing-module-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0103: Module name "Scrabble_Game" doesn't conform to snake_case naming style (invalid-name)
c:\Users\Phuc\Desktop\Scrabble_Game.py:17:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:28:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:33:0: C0116: Missing function or method docstring (missing-function-docstring)

-----------------------------------------------------------------
Your code has been rated at 8.54/10 (previous run: 8.54/10, +0.00)

PS C:\Users\Phuc>
```

The message in the terminal from Pylint is formatted as:

file path : **line number** : **column number** : **error code** : **short description**

Which is similar to flake8. The difference is that it has scoring. The higher the score, the better the code quality.

Now I will describe the errors and how to solve them:

ERROR C0303 indicates a line is longer than the limit, which is 100 according to the error message. Since there is no requirement for character limitation, we will define a maximum line length, which is similar to **ERROR E501** in flake8.

```
pylint --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
```

```
PS C:\Users\Phuc> pylint --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
************* Module Scrabble_Game
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0114: Missing module docstring (missing-module-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0103: Module name "Scrabble_Game" doesn't conform to snake_case naming style (invalid-name)
c:\Users\Phuc\Desktop\Scrabble_Game.py:17:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:28:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:33:0: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 8.96/10 (previous run: 8.54/10, +0.42)
```

Here we can see that the score raises by 0,42 points.

ERROR C0114 indicates that the module has no docstring.

```
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0114: Missing module docstring (missing-module-docstring)
```

To fix this, we need to add a docstring at the top of the module.

```
C: > Users > Phuc > Desktop > Scrabble_Game.py > ...
1   """Module providing a function printing python version."""
2
3   import random
4   import time
5
```

```
PS C:\Users\Phuc> pylint --max-line-length=300 c:/Users/Phuc/Desktop/Scrabble_Game.py
************* Module Scrabble_Game
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0103: Module name "Scrabble_Game" doesn't conform to snake_case naming style (invalid-name)
c:\Users\Phuc\Desktop\Scrabble_Game.py:19:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\Scrabble_Game.py:35:0: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 9.17/10 (previous run: 8.96/10, +0.21)
```

After fixing the error, the score raised by 0,21 points.

ERROR C0103 indicates when the name doesn't conform to naming rules associated to its type (constant, variable, class...).

```
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0103: Module name "Scrabble_Game" doesn't conform to snake_case naming style (invalid-name)
```

Based on snake_case naming style, my module name Scrabble_Game has uppercase letters. By changing all the uppercase letters to lowercase, we solve the problem.

```
************* Module Scrabble_Game
c:\Users\Phuc\Desktop\Scrabble_Game.py:1:0: C0103: Module name "Scrabble_Game" doesn't conform to snake_case naming style (invalid-name)
```

```
PS C:\Users\Phuc> pylint --max-line-length=300 c:/Users/Phuc/Desktop/scrabble_game.py
************* Module scrabble_game
c:\Users\Phuc\Desktop\scrabble_game.py:19:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\scrabble_game.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
c:\Users\Phuc\Desktop\scrabble_game.py:35:0: C0116: Missing function or method docstring (missing-function-docstring)

-------------------------------------------------------------------
Your code has been rated at 9.38/10 (previous run: 9.17/10, +0.21)

PS C:\Users\Phuc>
⚠ 0   0
```

After fixing the error, the score raised by 0,21 points

ERROR C0116 indicates when the name doesn't conform to naming rules associated to its type (constant, variable, class...).

```
c:\Users\Phuc\Desktop\Scrabble_Game.py:30:0: C0116: Missing function or method docstring (missing-function-docstring)
```

```
28
29  # Function to generate a random word length between 3 and 10
30  def get_word_length():
31      return random.randint(3, 10)
32
```

You can see in line there is no docstring. We can fix this problem by adding the docstring in the line that Pylint messages show.

```
29  # Function to generate a random word length between 3 and 10
30  def get_word_length():
31      """Function get word length version."""
32      return random.randint(3, 10)
33
```

```
PS C:\Users\Phuc> pylint --max-line-length=300 c:/Users/Phuc/Desktop/scrabble_game.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.79/10, +0.21)
```

When fixing all the errors, we have 10/10 in the score.

We can see that the errors from flake8 and Pylint are very minor, and we might rarely see it. However, in the long run, fixing those errors will help other programmers read our code.

Conclusion

The unit testing report shows I am trying to do using TDD and automated unit testing tools flake and Pylint8, and my process fixing the errors.

Reference:

Cordasco, I. (2016). *Flake8*. https://flake8.pycqa.org/en/3.1.1/manpage.html

Pylint community. (n.d.). *What is Pylint?*. Pylint community. https://pypi.org/project/pylint/

Zenva. (2013, December 12). Flake8 Tutorial – Complete Guide. Zenva Pty Ltd. https://gamedevacademy.org/flake8-tutorial-complete-guide/