Testing Report

Zhejun Xiang
S349548

1. Introduction:

According to the TDD requirements. The report is aiming to test the software in five requirements which mentioned on Assignment1. After player input the choice from Rock/ Paper/ Scissors, computer choose a random choice and then compare the result to judge who win who lose. Also, use flake8 and pylint to show the error not only in coding program but also in test stage.

i. The computer randomly selects one of scissors, paper and rock.
ii. The player can then choose to select/key one of the scissors, paper and rock options.
iii. The winner scores a point.
iv. The first person to score five points wins. The total number of rounds is also displayed.
v. Once a winner has been determined, the player is asked to quit or restart the game
vi. Players can also quit the game at any time.

2. Process:

At first, test Driven Development requires us to write a test cases before implementing the actual function.
Therefore, we write the tests before we write the coding.

Test Case

Project Name: Rock-paper-scissors game **Module Name:** Game test case

Created By: Zhejun Xiang **Created Date:** 2022/10/05

Executed by: Zhejun Xiang **Executed Date:** 2022/10/05

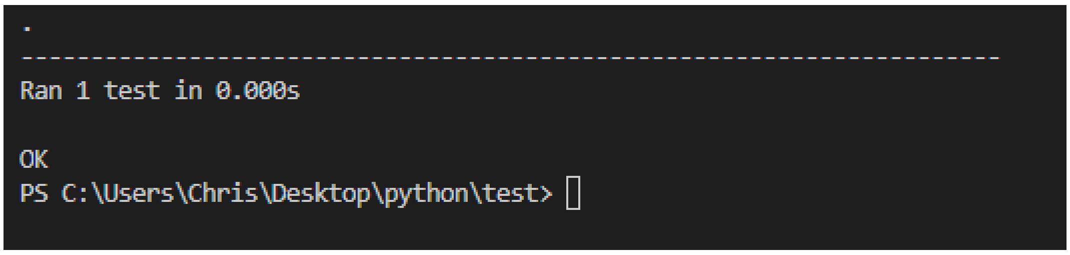
Test Case ID	Description	Test Step	Preconditions	Test Data	Expected Result	Actual Result	Status
1	Test random pick a choice from [rock,paper, scissors]	1. Call random_pick() for 20 times. 2. Check results are in [rock,paper,scissors]	Module imported	None	All return values are in [rock,paper,scissors]	All return values are in [rock,paper,scissors]	OK
2	Test validate user choice is of 'rock', 'paper', 'scissors'	1. Prepare input list 2. Prepare results 3. Call player = input('rock, paper or scissors: ').lower() 4. Check result	Module imported	input_list = ['', 'a', 'ABC', 'rock', 'paper', 'scissors',]	[False, False, False, True, True, True]	[False, False, False, True, True, True]	OK
3	Test compare computer choice and user choice	1. Prepare choice list 2. Prepare result list 3. Call who_win	Module imported	choice_list = [(game.ROCK, game.SCISSOR), (game.ROCK, game.ROCK), (game.ROCK, game.PAPER), (game.SCISSOR, game.SCISSOR), (game.SCISSOR, game.ROCK), (game.SCISSOR, game.PAPER), (game.PAPER, game.SCISSOR), (game.PAPER, game.ROCK), (game.PAPER, game.PAPER)]	‘player’, ’computer’, ‘tie’ ‘tie’, ‘player’, ‘computer’ ‘computer’, ‘tie’, ‘player’	‘player’, ’computer’, ‘tie’ ‘tie’, ‘player’, ‘computer’ ‘computer’, ‘tie’, ‘player’	OK

4	To test when should end the game	1. prepare scores for both computer and player 2. call is_end	Module imported	5:4 4:5 4:4	Player Computer None	Player Computer None	OK
5	To test quit after a single game	1. prepare user input 2. call is_quit	Program imported	Rock,yes Rock,no	Result—continue Result, real end	Result—continue Result, real end	OK

Choice test:
1,2

```
import unittest

from choice import *


class MyTestCase(unittest.TestCase):
    def test_choice(self):
        key='rock' or 'paper' or 'scissors'
        wronginput='glass'
        choice=['rock','paper','scissors']
        self.assertIn(key,choice)
        self.assertNotIn(wronginput,choice)

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

result:



```
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS C:\Users\Chris\Desktop\python\test> 
```

Rule test:
3

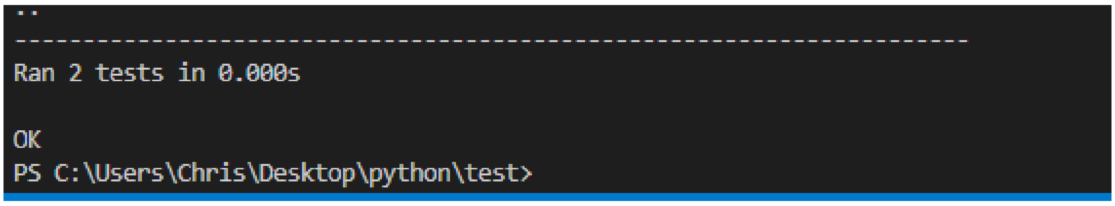
```
def game_once(player, computer):
    if player == computer:
        return 'tie'
    else:
        if judge(player,computer):
            return 'player'
        else:
            return 'computer'
```

```
def judge(player,computer):
    if player=='rock':
        if computer=='scissors':
            return True
        if computer=='paper':
            return False
    elif player=='scissors':
        if computer=='paper':
            return True
        if computer=='rock':
            return False
    elif player=='paper':
        if computer=='rock':
            return True
        if computer=='scissors':
            return False
```

result:

```
import unittest

import rule


class MyTestCase(unittest.TestCase):
#all posible situation
    def test_valid_game_once(self):
        self.assertEqual(rule.game_once(player='rock',
computer='scissors'), 'player')
        self.assertEqual(rule.game_once(player='paper',computer='scissor
s'),'computer')
        self.assertEqual(rule.game_once(player='scissors',computer='scis
sors'),'tie')

        self.assertEqual(rule.game_once(player='rock', computer='rock'),
'tie')
```

```
        self.assertEqual(rule.game_once(player='paper',computer='rock'),'player')
        self.assertEqual(rule.game_once(player='scissors',computer='rock'),'computer')

        self.assertEqual(rule.game_once(player='rock', computer='paper'), 'computer')
        self.assertEqual(rule.game_once(player='paper',computer='paper'),'tie')
        self.assertEqual(rule.game_once(player='scissors',computer='paper'),'player')
#wrong result test
        self.assertNotEqual(rule.game_once(player='rock', computer='paper'), 'player' or 'tie')
```



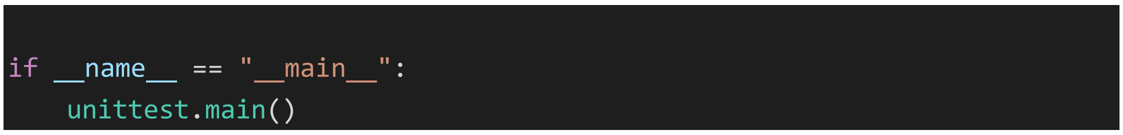
```
..
----------------------------------------------------------------------
Ran 2 tests in 0.000s

OK
PS C:\Users\Chris\Desktop\python\test>
```

Ending test:
4

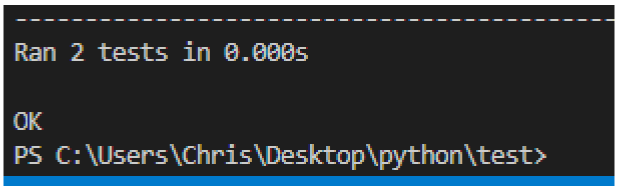
```
def is_end(player_score, computer_score):
    """
        return : if game is over(bool), winner (string)
    """
    if player_score >= 5:
        return True, "player"
    elif computer_score >= 5:
        return True, "computer"
    else:
        return False, None
```

result:

```
import unittest

import rule


class MyTestCase(unittest.TestCase):
#end rule test
    def test_valid_is_end(self):
        self.assertEqual(rule.is_end(5, 4), (True, "player"))
        self.assertEqual(rule.is_end(4, 5), (True, "computer"))
```

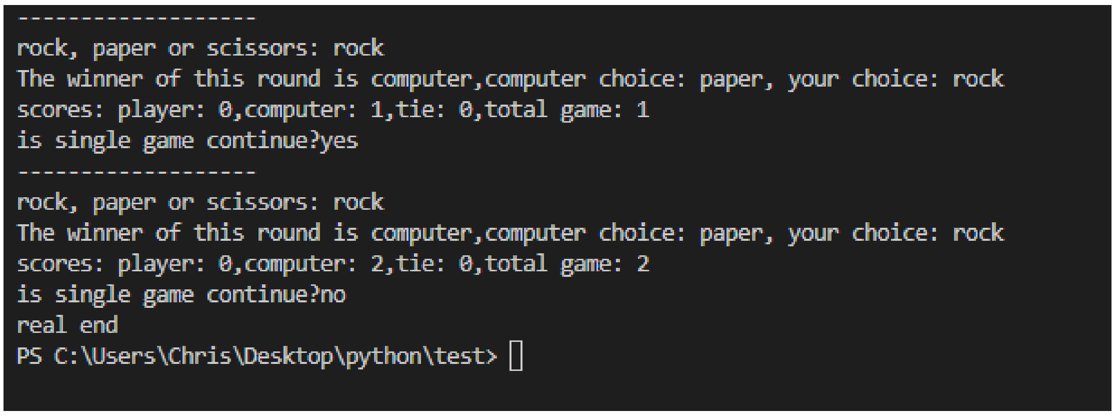


```
if __name__ == "__main__":
    unittest.main()
```



```
----------------------------------------------
Ran 2 tests in 0.000s

OK
PS C:\Users\Chris\Desktop\python\test>
```

Single game quit test:
5



```
-------------------
rock, paper or scissors: rock
The winner of this round is computer,computer choice: paper, your choice: rock
scores: player: 0,computer: 1,tie: 0,total game: 1
is single game continue?yes
-------------------
rock, paper or scissors: rock
The winner of this round is computer,computer choice: paper, your choice: rock
scores: player: 0,computer: 2,tie: 0,total game: 2
is single game continue?no
real end
PS C:\Users\Chris\Desktop\python\test> 
```

3. **Conclusion:**

Through this assignment on software testing, I gradually understood the essential meaning of testing and how to substitute the thinking of testing in the coding process, so that the entire software development team can develop in an orderly and efficient manner. Test-driven development provides the greatest results when the code is constantly being improved. It allows me to continually verify correctness and to driver the program design. By dividing the software as a whole into functional functions that run independently, in the design process of a single test, I realized the integrity and smoothness of the software process from declaring variables to final output results. Because of writing TDD reports, I have a better understanding of the thinking of testing, especially the input items corresponding to each function of the software, the function itself, and the output items. The role of individual modules in the software, and when adjustments should be made to the overall structure of the software. . . both helped me a lot.

Github Link:
https://github.com/CDUChris/ROCK-SCISSORS-PAPER-GAME