PRT582 – Software Engineering Process and Tools
Software Unit Testing Report

MD AL AMIN SARKER
Student ID: S361035

Introduction:

TDD is a test-first development process where we write our test first before writing functional code to fulfill that test so that we can think through our requirements or design before writing our functional code. By using TDD we can write clean code that works. However, I have developed/implemented a Scissor Paper Rock game using Test Driven Development in Python based on the below requirements, and going to explain the process that I have followed and the test cases that I have performed.

The basic game requirements are:

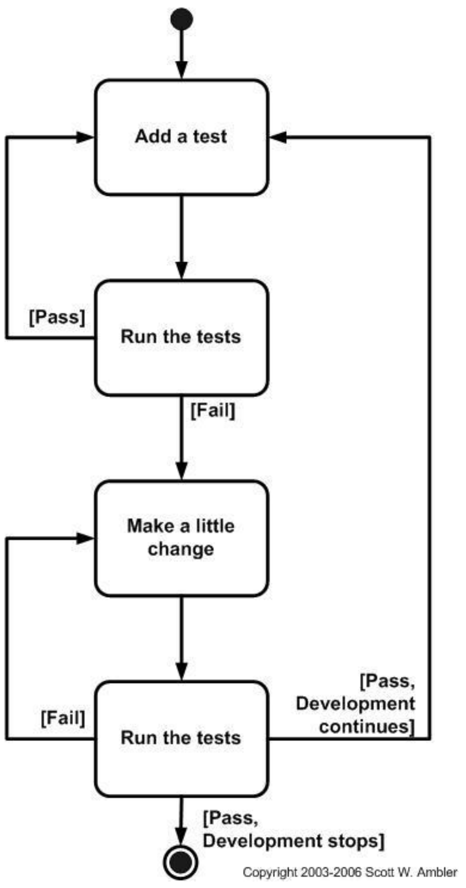
I. The computer randomly picks one of the options from Scissor, Paper and Rock.
II. The player is then given the option to type one of the options of Scissor, Paper and Rock.
III. One point is given to the winner.
IV. The first to get five points wins the game.
V. The total number of rounds played in total will also be displayed.
VI. Once the winner is determined, the player is asked to quit or restart the game
VII. The player can also quit the game at any time.

For programming language, I have used Python and for unit testing, I have used the Python Unit testing framework. As a static code analyzer for Python I have utilized the Pylint library and to check the code base against coding style and programming errors used the Python Flake8 library. Finally, for the source-code editor, I have set up my programming environment in Visual Studio Code.

Process:

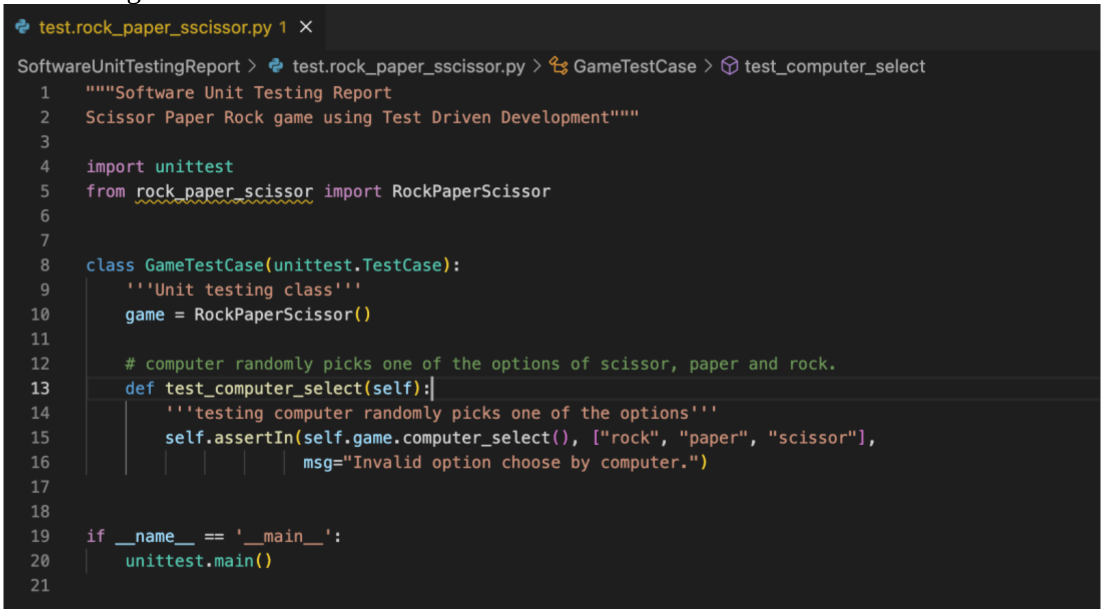
The steps involved in the test first development that I followed can be seen in below diagram and the steps are:

- At first, we have to add a test quickly, basically just enough code to fail.
- Next, we run our tests to ensure that the new test does in fact fail.
- Then we update our functional code to make it pass the new tests.
- After that, we run our tests again.
- If they fail we need to update our functional code and retest.
- Once the tests pass the next step is to start over.


Add a test
Run the tests
[Pass]
[Fail]
Make a little change
Run the tests
[Fail]
[Pass, Development continues]
[Pass, Development stops]
Copyright 2003-2006 Scott W. Ambler

Step 1:
At first, I created a python file named ‘test_rock_paper_scissor.py’ by following the snake case naming convention and going to follow this standard convention for the rest of my development process to make it understandable. However, then imported the unit testing framework to write my first test case. I have created the first test case based on the first requirement which is “computer randomly picks one of the options from Scissor, Paper and Rock”.

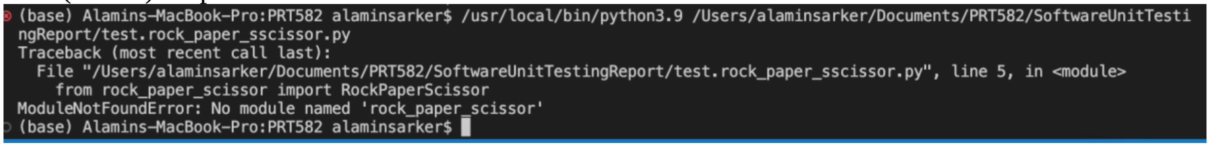
Unit testing code:


test.rock_paper_sscissor.py 1 ×
SoftwareUnitTestingReport > test.rock_paper_sscissor.py > GameTestCase > test_computer_select

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import unittest
from rock_paper_scissor import RockPaperScissor


class GameTestCase(unittest.TestCase):
    '''Unit testing class'''
    game = RockPaperScissor()

    # computer randomly picks one of the options of scissor, paper and rock.
    def test_computer_select(self):
        '''testing computer randomly picks one of the options'''
        self.assertIn(self.game.computer_select(), ["rock", "paper", "scissor"],
                      msg="Invalid option choose by computer.")


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

```

It is going to check if the computer is picking one of the options or not and I know that the test will fail as I haven't written my functional code for that method yet and what's more I haven't even created my gaming code file yet either.

Test (Failed) output:

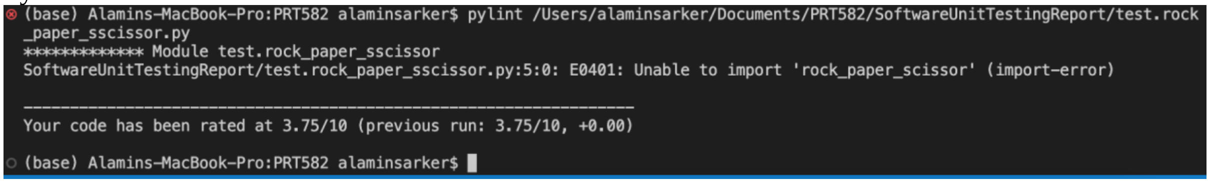


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 5, in <module>
    from rock_paper_scissor import RockPaperScissor
ModuleNotFoundError: No module named 'rock_paper_scissor'
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

As expected, it got a "ModuleNotFoundError" exception in the output console. "ModuleNotFoundError: No module named 'rock_paper_scissor!".

I am also going to check and analyze my coding style and programming errors often with the Pylint and flake8 libraries.

Pylint:

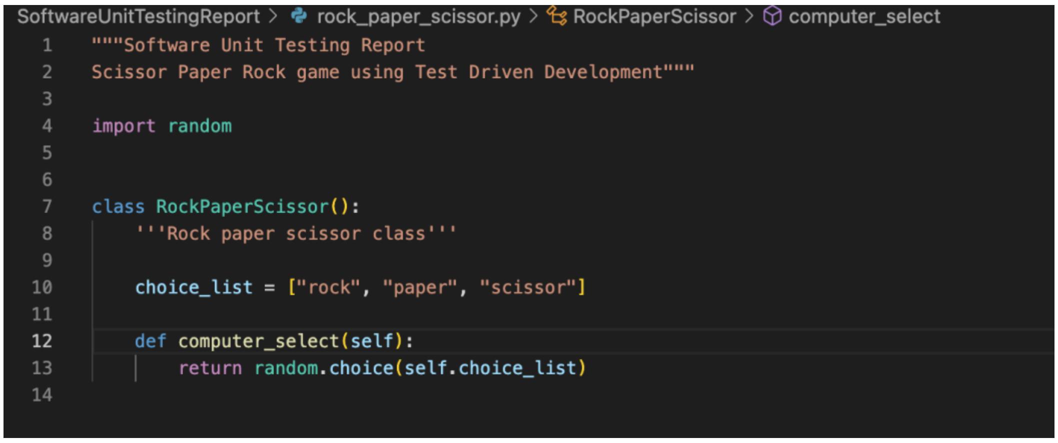


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py
************* Module test.rock_paper_sscissor
SoftwareUnitTestingReport/test.rock_paper_sscissor.py:5:0: E0401: Unable to import 'rock_paper_scissor' (import-error)

------------------------------------------------------------------
Your code has been rated at 3.75/10 (previous run: 3.75/10, +0.00)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After running Pylint for my test file I can see that score is quite low due to the error of not finding the functional coding file but other than that coding convention seems right.

Now it's time for me to create my game python file and write the first method to test the first requirement using the test case that I wrote before. So I wrote the code for the computer selection of Rock, Paper and Scissor like below:

Functional code for method:

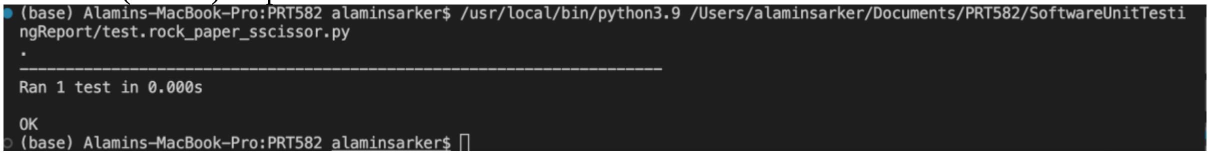


SoftwareUnitTestingReport > rock_paper_scissor.py > RockPaperScissor > computer_select

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import random


class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]

    def computer_select(self):
        return random.choice(self.choice_list)

```

Now test our test case again and see what happens, I am expecting to get it to pass this time.

Unit Test (Passed) output:

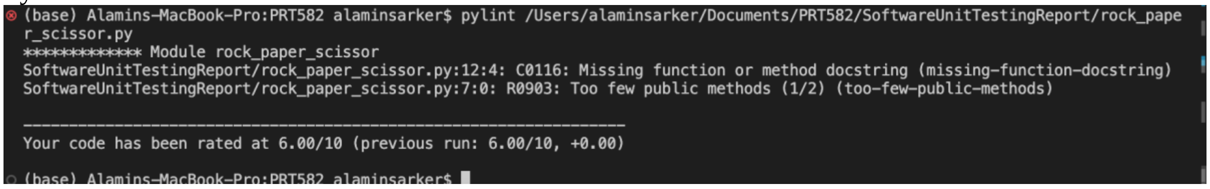


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

So after running the unit test again it passed and got the output mentioned above.

Now I want to see if my coding style is correct or not, lets run Pylint on my game python file "'rock_paper_scissor.py".

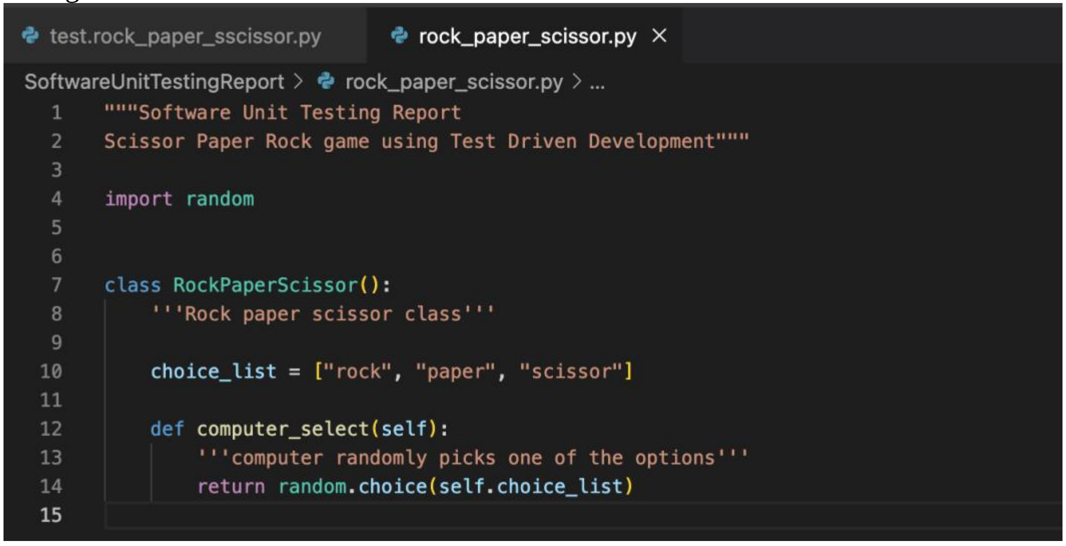
Pylint for functional code:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py
************* Module rock_paper_scissor
SoftwareUnitTestingReport/rock_paper_scissor.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
SoftwareUnitTestingReport/rock_paper_scissor.py:7:0: R0903: Too few public methods (1/2) (too-few-public-methods)

------------------------------------------------------------------
Your code has been rated at 6.00/10 (previous run: 6.00/10, +0.00)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After running pylint, I can see that it is showing that my method doesn't have any docstring, so I am going to add that to my game python file.

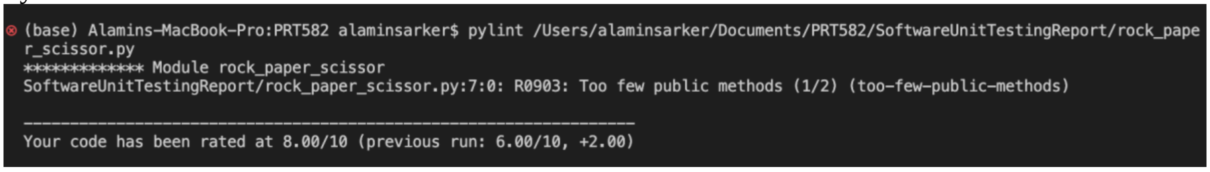
Fixing functional code:



test.rock_paper_sscissor.py
rock_paper_scissor.py
SoftwareUnitTestingReport > rock_paper_scissor.py > ...

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import random


class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]

    def computer_select(self):
        '''computer randomly picks one of the options'''
        return random.choice(self.choice_list)

```

Pylint:

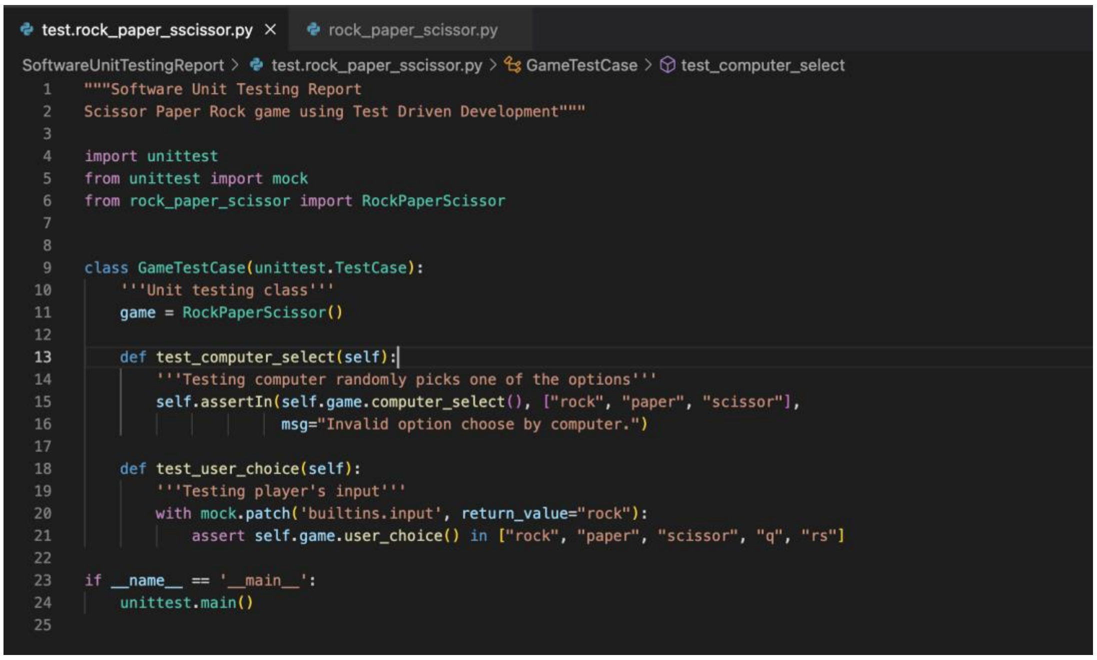


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py
************* Module rock_paper_scissor
SoftwareUnitTestingReport/rock_paper_scissor.py:7:0: R0903: Too few public methods (1/2) (too-few-public-methods)

------------------------------------------------------------------
Your code has been rated at 8.00/10 (previous run: 6.00/10, +2.00)
```

After fixing the issue of the docstring, scores improved. However, now I know my coding style is in the right format and will follow that style as I progress and will check for any issues at the end with Pylint and flake8.

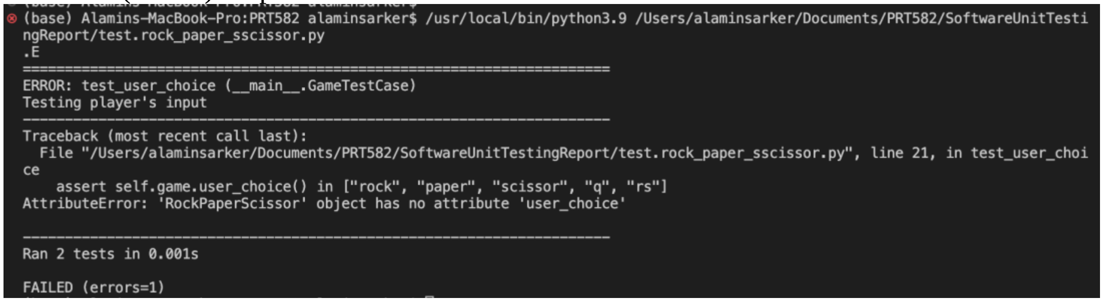
Step 2:
Now, I am going to move to the next requirement which is "Player is then given the option to type one of the options of scissor, paper and rock." and create my test case based on that and also add the functionality of quitting and restarting the game which will be an input from the player. So let's do that.

Unit testing code:


test.rock_paper_sscissor.py
rock_paper_scissor.py
SoftwareUnitTestingReport > test.rock_paper_sscissor.py > GameTestCase > test_computer_select

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import unittest
from unittest import mock
from rock_paper_scissor import RockPaperScissor


class GameTestCase(unittest.TestCase):
    '''Unit testing class'''
    game = RockPaperScissor()

    def test_computer_select(self):
        '''Testing computer randomly picks one of the options'''
        self.assertIn(self.game.computer_select(), ["rock", "paper", "scissor"],
                      msg="Invalid option choose by computer.")

    def test_user_choice(self):
        '''Testing player's input'''
        with mock.patch('builtins.input', return_value="rock"):
            assert self.game.user_choice() in ["rock", "paper", "scissor", "q", "rs"]

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

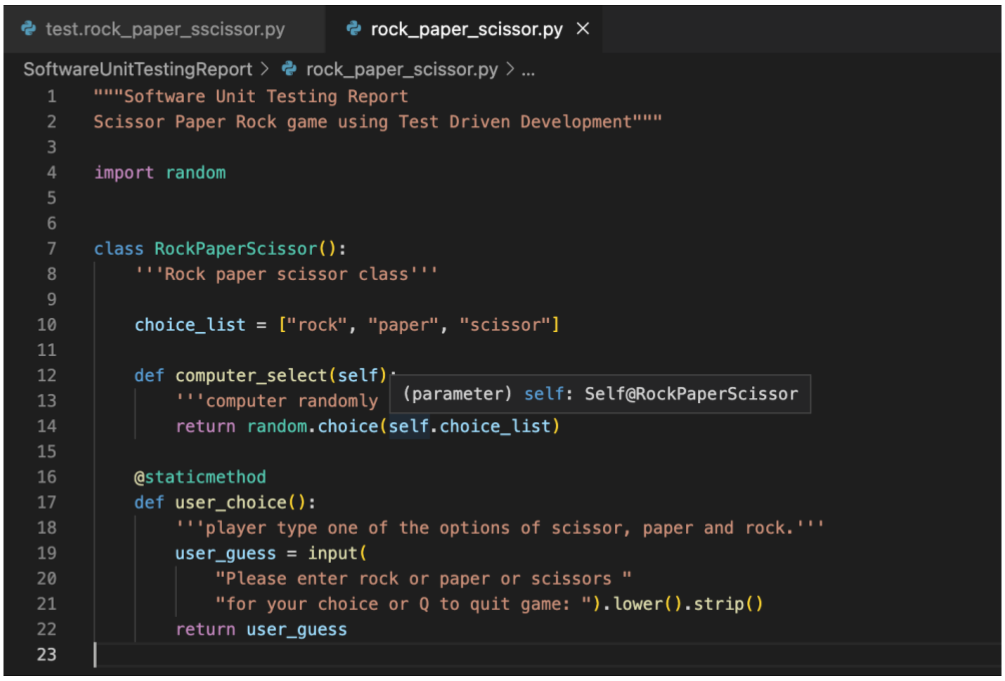
Unit Test (Failed) output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
.E
======================================================================
ERROR: test_user_choice (__main__.GameTestCase)
Testing player's input
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 21, in test_user_choi
ce
    assert self.game.user_choice() in ["rock", "paper", "scissor", "q", "rs"]
AttributeError: 'RockPaperScissor' object has no attribute 'user_choice'

----------------------------------------------------------------------
Ran 2 tests in 0.001s

FAILED (errors=1)
```

After running the unit testing, I can see that it ran 2 tests and failed because of one error as I haven't written the functional code yet. Let's write the functional code for this test case.

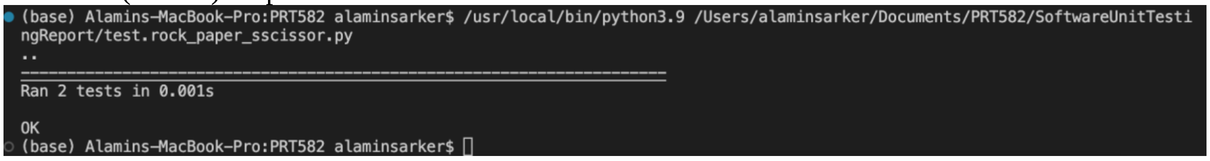
Functional Code:


test.rock_paper_sscissor.py
rock_paper_scissor.py
SoftwareUnitTestingReport > rock_paper_scissor.py > ...
(parameter) self: Self@RockPaperScissor

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import random


class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]

    def computer_select(self):
        '''computer randomly
        return random.choice(self.choice_list)

    @staticmethod
    def user_choice():
        '''player type one of the options of scissor, paper and rock.'''
        user_guess = input(
            "Please enter rock or paper or scissors "
            "for your choice or Q to quit game: ").lower().strip()
        return user_guess

```

Now, I am going to perform the testing again to check if my unit test passes or fails.

Unit Test (Passed) output:

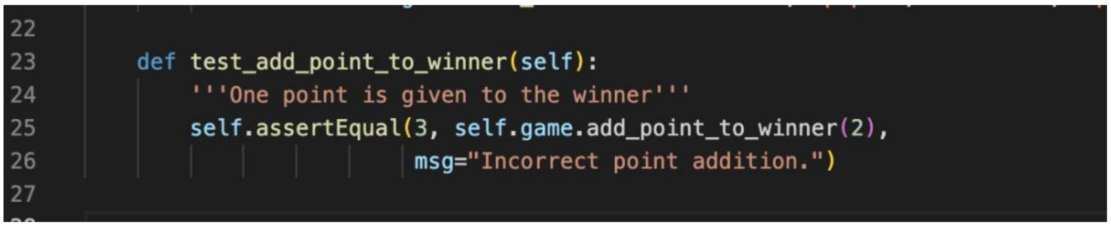


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

After running the test file I can see that both test cases passed. So, I am moving to the next requirement for the unit test.

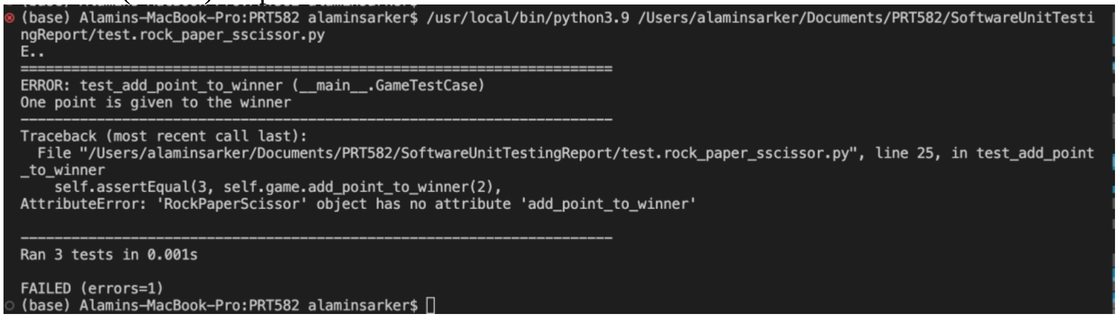
Step 3:
I wrote the unit test code for the requirement "One point is given to the winner" and after running this test I am expecting it to fail. Let's do that now.

Unit test code:



```
22
23      def test_add_point_to_winner(self):
24          '''One point is given to the winner'''
25          self.assertEqual(3, self.game.add_point_to_winner(2),
26                           msg="Incorrect point addition.")
27
```

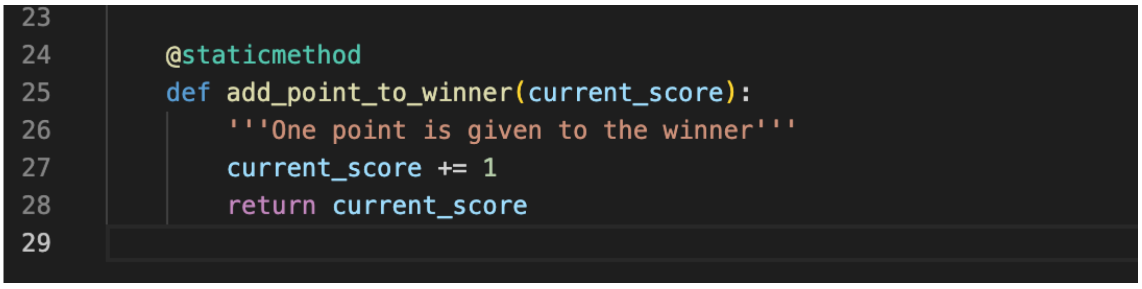
Unit Test (Failed) output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
E..
======================================================================
ERROR: test_add_point_to_winner (__main__.GameTestCase)
One point is given to the winner
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 25, in test_add_point
_to_winner
    self.assertEqual(3, self.game.add_point_to_winner(2),
AttributeError: 'RockPaperScissor' object has no attribute 'add_point_to_winner'

----------------------------------------------------------------------
Ran 3 tests in 0.001s

FAILED (errors=1)
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

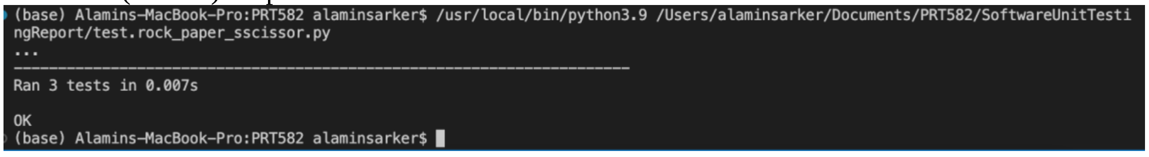
After running the test code as expected it failed with an error that "add_point_to_winner()" method is not found because it is not created yet.

Method code:



```
    @staticmethod
    def add_point_to_winner(current_score):
        '''One point is given to the winner'''
        current_score += 1
        return current_score
```

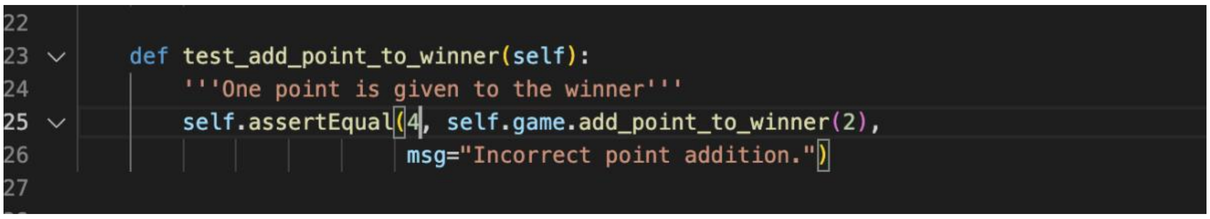
Unit Test (Passed) output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.007s

OK
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

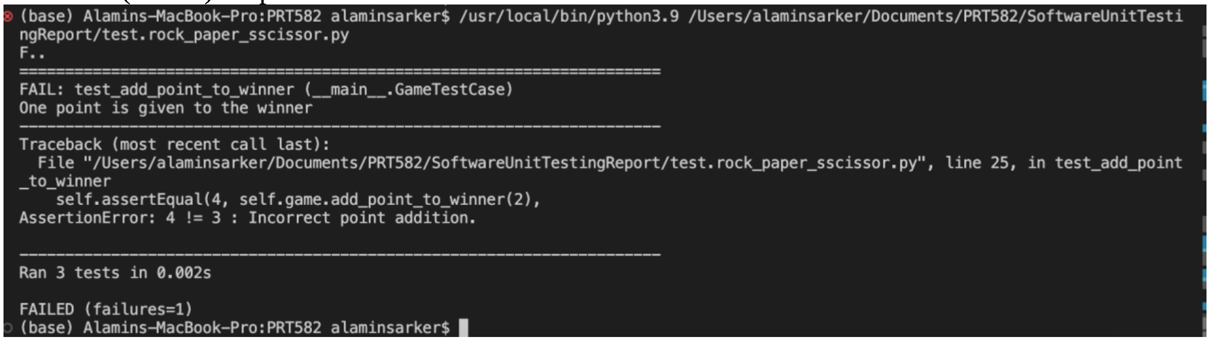
Now, I am going to check if my "add_point_to_winner()" is returning the correct result or not. Therefore, I have changed the value from 3 to 4, but after adding 1 with 2 it should be 3. So, my test case is going to fail for sure.

Failed test case code:



```
    def test_add_point_to_winner(self):
        '''One point is given to the winner'''
        self.assertEqual(4, self.game.add_point_to_winner(2),
                         msg="Incorrect point addition.")
```

Unit Test (Failed) output:

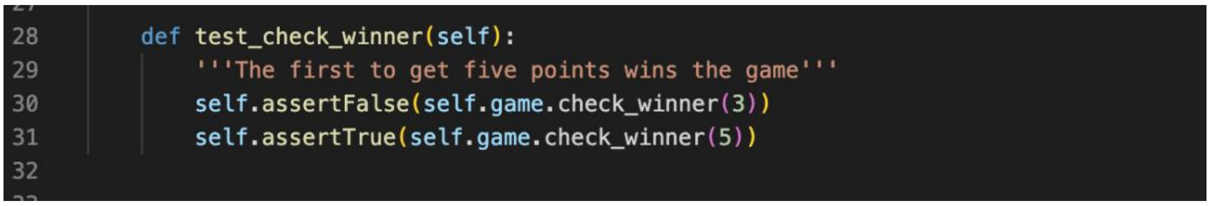


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
F..
======================================================================
FAIL: test_add_point_to_winner (__main__.GameTestCase)
One point is given to the winner
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 25, in test_add_point
_to_winner
    self.assertEqual(4, self.game.add_point_to_winner(2),
AssertionError: 4 != 3 : Incorrect point addition.

----------------------------------------------------------------------
Ran 3 tests in 0.002s

FAILED (failures=1)
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After running the test python file, I can see it failed as the returned value did not match with the expected value.

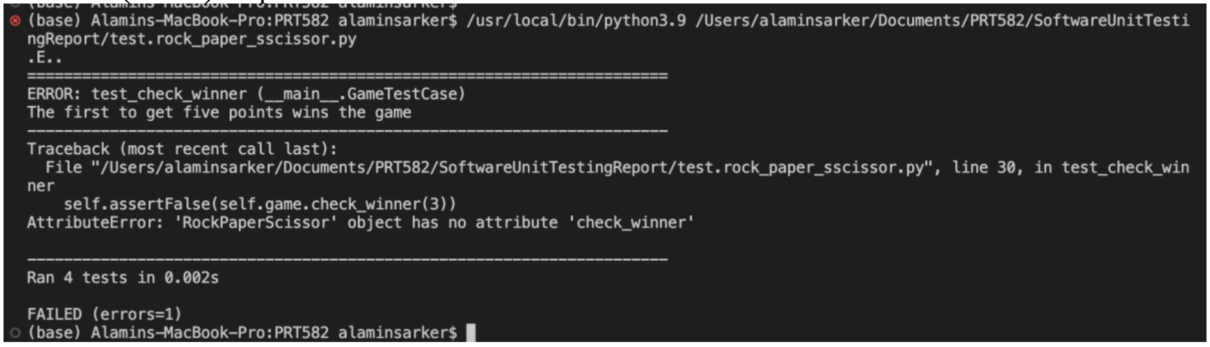
Step 4:
Now I am going to create my next test case for the requirement "The first to get five points wins the game".

Unit test code:



```
    def test_check_winner(self):
        '''The first to get five points wins the game'''
        self.assertFalse(self.game.check_winner(3))
        self.assertTrue(self.game.check_winner(5))
```

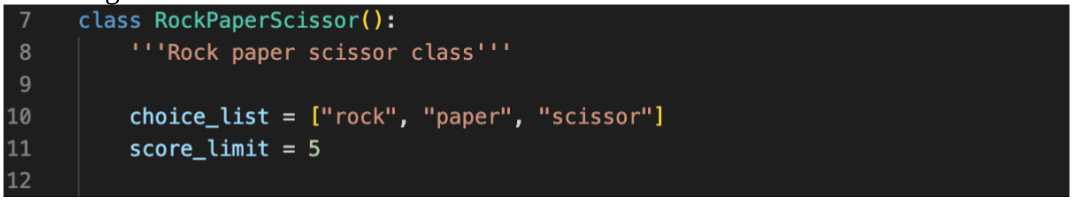
Unit Test (Failed) output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
.E..
======================================================================
ERROR: test_check_winner (__main__.GameTestCase)
The first to get five points wins the game
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 30, in test_check_win
ner
    self.assertFalse(self.game.check_winner(3))
AttributeError: 'RockPaperScissor' object has no attribute 'check_winner'

----------------------------------------------------------------------
Ran 4 tests in 0.002s

FAILED (errors=1)
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

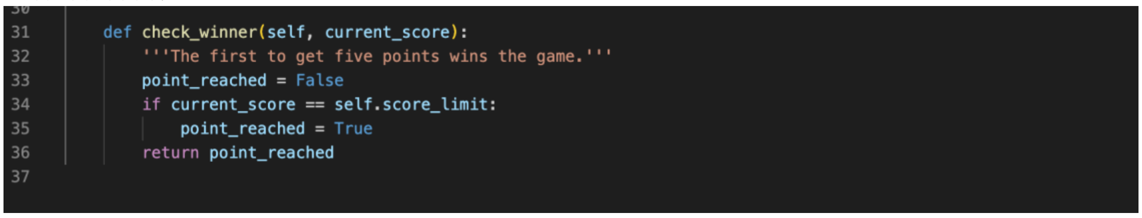
As expected got an error after running my test python file. Let’s write the functional code for my above test case.

Declaring a class variable for score limit.



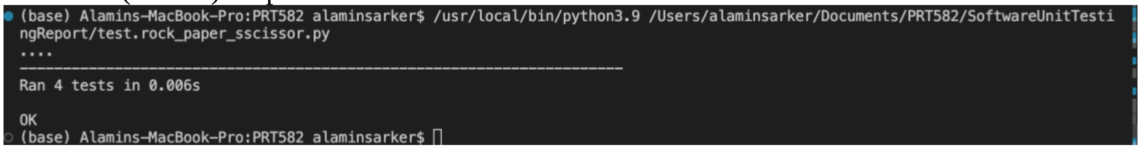
```python
class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]
    score_limit = 5

```

Method code:



```python
    def check_winner(self, current_score):
        '''The first to get five points wins the game.'''
        point_reached = False
        if current_score == self.score_limit:
            point_reached = True
        return point_reached

```

Unit Test (Passed) output:



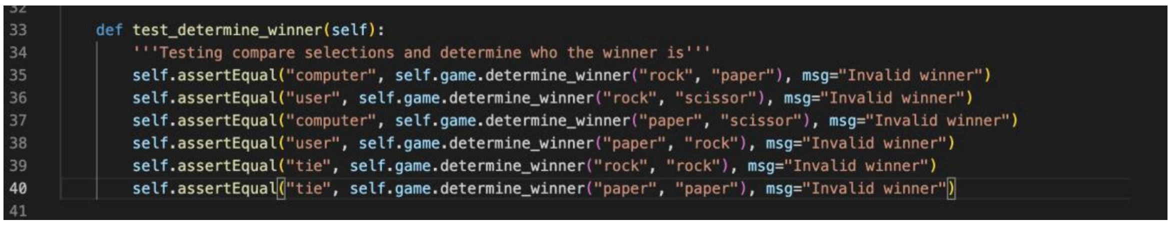
```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
....
----------------------------------------------------------------------
Ran 4 tests in 0.006s

OK
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

All the test cases are passed and so I am moving to the next one.

Step 5:
Now I have written a test case to determine the winner according to the following rules:

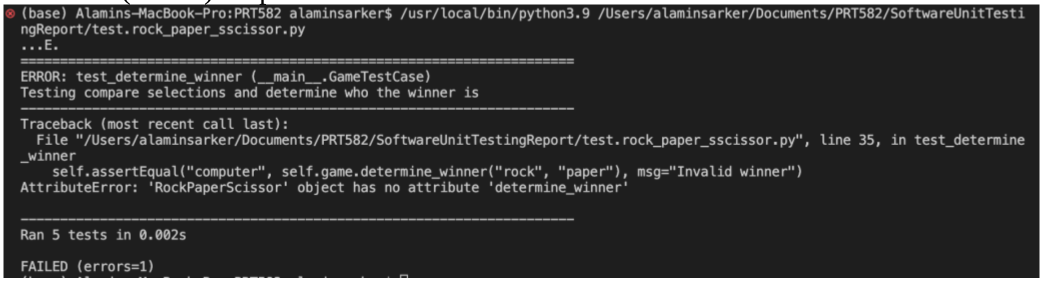
- rock vs paper -> paper wins
- rock vs scissor -> rock wins
- paper vs scissor -> scissor wins
- Tie the computer and player choose same option

Unit test code:



```python
    def test_determine_winner(self):
        '''Testing compare selections and determine who the winner is'''
        self.assertEqual("computer", self.game.determine_winner("rock", "paper"), msg="Invalid winner")
        self.assertEqual("user", self.game.determine_winner("rock", "scissor"), msg="Invalid winner")
        self.assertEqual("computer", self.game.determine_winner("paper", "scissor"), msg="Invalid winner")
        self.assertEqual("user", self.game.determine_winner("paper", "rock"), msg="Invalid winner")
        self.assertEqual("tie", self.game.determine_winner("rock", "rock"), msg="Invalid winner")
        self.assertEqual("tie", self.game.determine_winner("paper", "paper"), msg="Invalid winner")

```

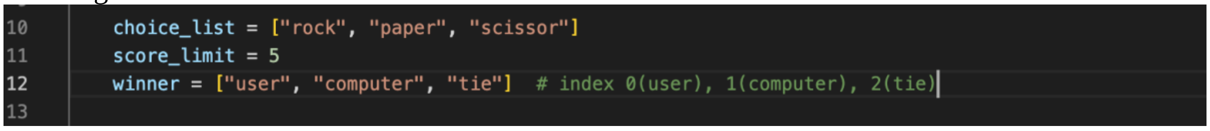
Unit Test (Failed) output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
...E.
======================================================================
ERROR: test_determine_winner (__main__.GameTestCase)
Testing compare selections and determine who the winner is
----------------------------------------------------------------------
Traceback (most recent call last):
  File "/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py", line 35, in test_determine
_winner
    self.assertEqual("computer", self.game.determine_winner("rock", "paper"), msg="Invalid winner")
AttributeError: 'RockPaperScissor' object has no attribute 'determine_winner'

----------------------------------------------------------------------
Ran 5 tests in 0.002s

FAILED (errors=1)
```

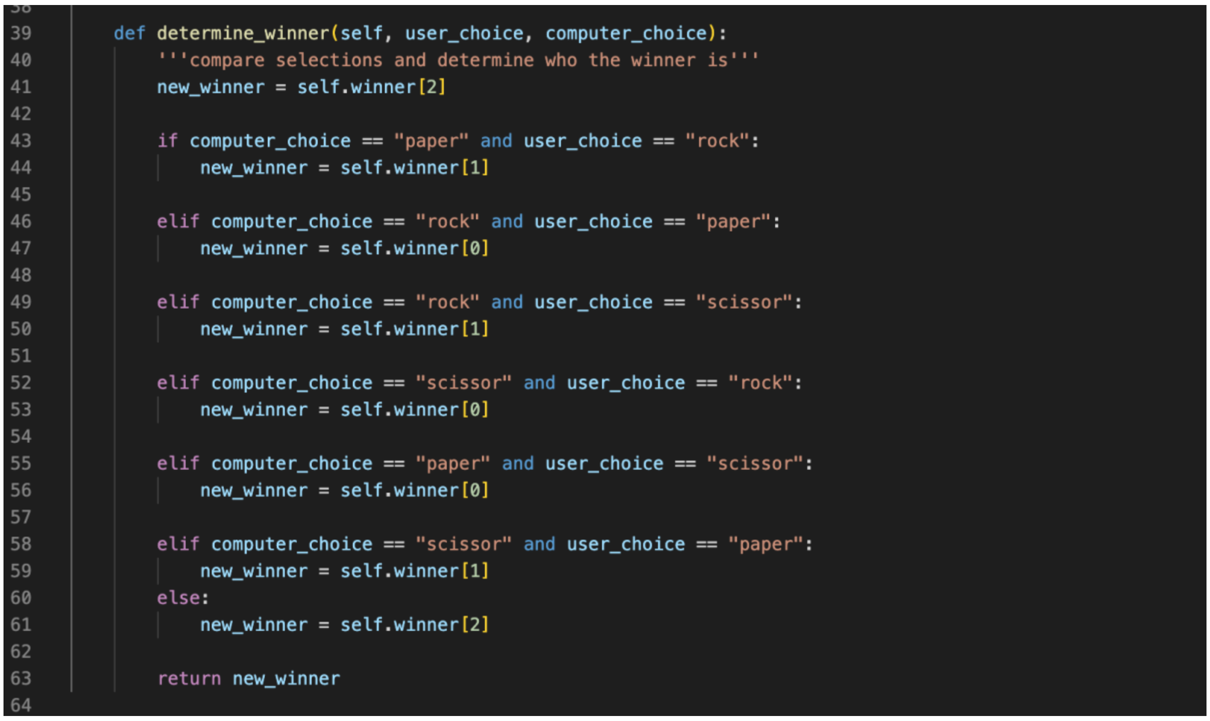
The unit test failed due to errors of not finding the functional code method. Let's write the functional code for that test case.

Defining the variable of a list of winners.



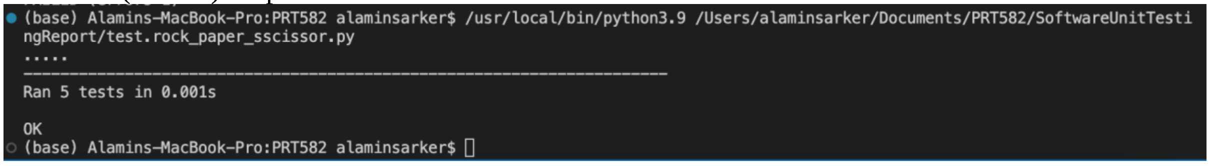
```python
    choice_list = ["rock", "paper", "scissor"]
    score_limit = 5
    winner = ["user", "computer", "tie"]  # index 0(user), 1(computer), 2(tie)
```

Method code:



```python
    def determine_winner(self, user_choice, computer_choice):
        '''compare selections and determine who the winner is'''
        new_winner = self.winner[2]

        if computer_choice == "paper" and user_choice == "rock":
            new_winner = self.winner[1]

        elif computer_choice == "rock" and user_choice == "paper":
            new_winner = self.winner[0]

        elif computer_choice == "rock" and user_choice == "scissor":
            new_winner = self.winner[1]

        elif computer_choice == "scissor" and user_choice == "rock":
            new_winner = self.winner[0]

        elif computer_choice == "paper" and user_choice == "scissor":
            new_winner = self.winner[0]

        elif computer_choice == "scissor" and user_choice == "paper":
            new_winner = self.winner[1]
        else:
            new_winner = self.winner[2]

        return new_winner
```

Unit Test (Passed) output:

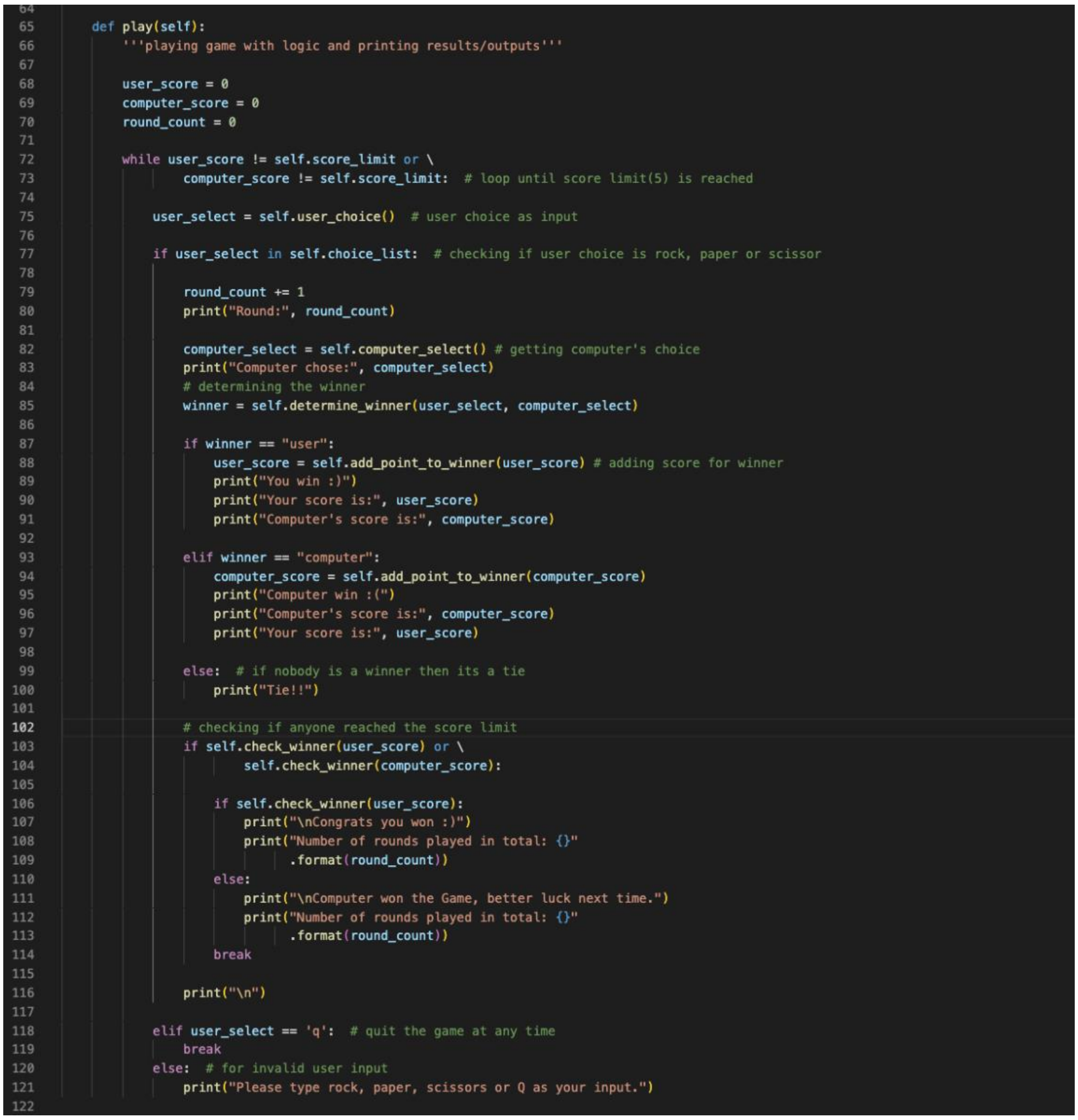


```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTesti
ngReport/test.rock_paper_sscissor.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

After running the test file, I can see that all the unit test cases passed for game requirement for winner condition.

Step 6:
Now I am going to write a function for the game to combine all the methods to play the game.

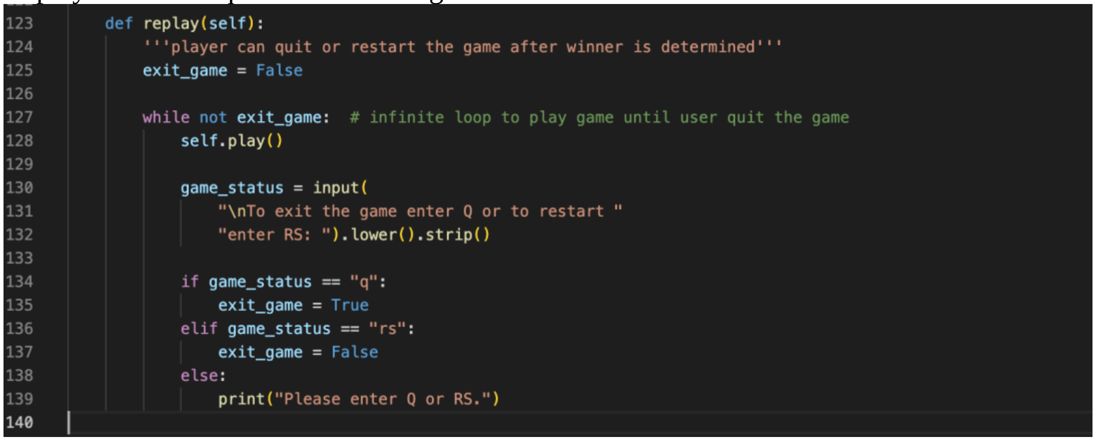
Play() method to play the game until one player reached score 5:



```python
    def play(self):
        '''playing game with logic and printing results/outputs'''

        user_score = 0
        computer_score = 0
        round_count = 0

        while user_score != self.score_limit or \
                computer_score != self.score_limit:  # loop until score limit(5) is reached

            user_select = self.user_choice()  # user choice as input

            if user_select in self.choice_list:  # checking if user choice is rock, paper or scissor

                round_count += 1
                print("Round:", round_count)

                computer_select = self.computer_select() # getting computer's choice
                print("Computer chose:", computer_select)
                # determining the winner
                winner = self.determine_winner(user_select, computer_select)

                if winner == "user":
                    user_score = self.add_point_to_winner(user_score) # adding score for winner
                    print("You win :)")
                    print("Your score is:", user_score)
                    print("Computer's score is:", computer_score)

                elif winner == "computer":
                    computer_score = self.add_point_to_winner(computer_score)
                    print("Computer win :(")
                    print("Computer's score is:", computer_score)
                    print("Your score is:", user_score)

                else:  # if nobody is a winner then its a tie
                    print("Tie!!")

                # checking if anyone reached the score limit
                if self.check_winner(user_score) or \
                        self.check_winner(computer_score):

                    if self.check_winner(user_score):
                        print("\nCongrats you won :)")
                        print("Number of rounds played in total: {}"
                              .format(round_count))
                    else:
                        print("\nComputer won the Game, better luck next time.")
                        print("Number of rounds played in total: {}"
                              .format(round_count))
                    break

                print("\n")

            elif user_select == 'q':  # quit the game at any time
                break
            else:  # for invalid user input
                print("Please type rock, paper, scissors or Q as your input.")
```

Step 7:
Now, I need one last method that is going to enable the player to quit or restart the game.

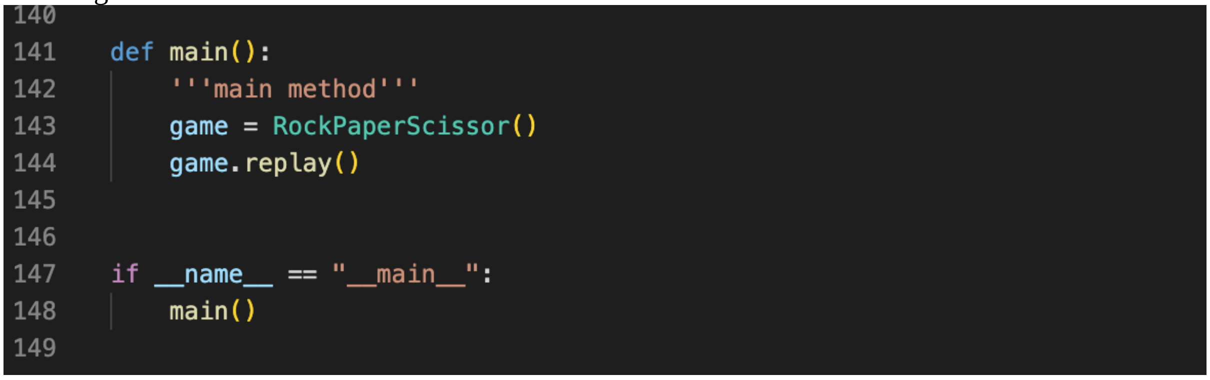
Replay method to quit or restart the game:



```python
    def replay(self):
        '''player can quit or restart the game after winner is determined'''
        exit_game = False

        while not exit_game:  # infinite loop to play game until user quit the game
            self.play()

            game_status = input(
                "\nTo exit the game enter Q or to restart "
                "enter RS: ").lower().strip()

            if game_status == "q":
                exit_game = True
            elif game_status == "rs":
                exit_game = False
            else:
                print("Please enter Q or RS.")
```

Step 8:

Defining the Main method:



```python
def main():
    '''main method'''
    game = RockPaperScissor()
    game.replay()


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

My Unit test code file looks like this after all the steps that I followed.

```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import unittest
from unittest import mock
from rock_paper_scissor import RockPaperScissor

class GameTestCase(unittest.TestCase):
    '''Unit testing class'''
    game = RockPaperScissor()

    def test_computer_select(self):
        '''Testing computer randomly picks one of the options'''
        self.assertIn(self.game.computer_select(), ["rock", "paper", "scissor"],
                      msg="Invalid option choose by computer.")

    def test_user_choice(self):
        '''Testing player's input'''
        with mock.patch('builtins.input', return_value="rock"):
            assert self.game.user_choice() in ["rock", "paper", "scissor", "q", "rs"]

    def test_add_point_to_winner(self):
        '''Testing tne point is given to the winner'''
        self.assertEqual(3, self.game.add_point_to_winner(2),
                         msg="Incorrect point addition.")

    def test_check_winner(self):
        '''Testing first to get five points wins the game'''
        self.assertFalse(self.game.check_winner(3))
        self.assertTrue(self.game.check_winner(5))

    def test_determine_winner(self):
        '''Testing compare selections and determine who the winner is'''
        self.assertEqual("computer", self.game.determine_winner("rock", "paper"), msg="Invalid winner")
        self.assertEqual("user", self.game.determine_winner("rock", "scissor"), msg="Invalid winner")
        self.assertEqual("computer", self.game.determine_winner("paper", "scissor"), msg="Invalid winner")
        self.assertEqual("user", self.game.determine_winner("paper", "rock"), msg="Invalid winner")
        self.assertEqual("tie", self.game.determine_winner("rock", "rock"), msg="Invalid winner")
        self.assertEqual("tie", self.game.determine_winner("paper", "paper"), msg="Invalid winner")

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

```

And my rock paper scissor functional code is as below:

```
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""

import random



class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]
    score_limit = 5
    winner = ["user", "computer", "tie"]  # index 0(user), 1(computer), 2(tie)

    def computer_select(self):
        '''computer randomly picks one of the options'''
        return random.choice(self.choice_list)

    @staticmethod
    def user_choice():
        '''player type one of the options of scissor, paper and rock.'''
        user_guess = input(
            "Please enter rock or paper or scissors "
            "for your choice or Q to quit game: ").lower().strip()
        return user_guess

    @staticmethod
    def add_point_to_winner(current_score):
        '''One point is given to the winner'''
        current_score += 1
        return current_score

    def check_winner(self, current_score):
        '''The first to get five points wins the game.'''
        point_reached = False
        if current_score == self.score_limit:
            point_reached = True
        return point_reached

    def determine_winner(self, user_choice, computer_choice):
        '''compare selections and determine who the winner is'''
        new_winner = self.winner[2]
        if computer_choice == "paper" and user_choice == "rock":
            new_winner = self.winner[1]

        elif computer_choice == "rock" and user_choice == "paper":
            new_winner = self.winner[0]

        elif computer_choice == "rock" and user_choice == "scissor":
            new_winner = self.winner[1]

        elif computer_choice == "scissor" and user_choice == "rock":
            new_winner = self.winner[0]

        elif computer_choice == "paper" and user_choice == "scissor":
            new_winner = self.winner[0]

        elif computer_choice == "scissor" and user_choice == "paper":
            new_winner = self.winner[1]
        else:
            new_winner = self.winner[2]

        return new_winner

    def play(self):
        '''playing game with logic and printing results/outputs'''

        user_score = 0
        computer_score = 0
        round_count = 0

        while user_score != self.score_limit or \
                computer_score != self.score_limit:  # loop until score limit(5) is reached

            user_select = self.user_choice()  # user choice as input

            if user_select in self.choice_list:  # checking if user choice is rock, paper or scissor

                round_count += 1
                print("Round:", round_count)

                computer_select = self.computer_select() # getting computer's choice
                print("Computer chose:", computer_select)
                # determining the winner
                winner = self.determine_winner(user_select, computer_select)

                if winner == "user":
                    user_score = self.add_point_to_winner(user_score) # adding score for winner
                    print("You win :)")
                    print("Your score is:", user_score)
                    print("Computer's score is:", computer_score)
```

```python
                elif winner == "computer":
                    computer_score = self.add_point_to_winner(computer_score)
                    print("Computer win :(")
                    print("Computer's score is:", computer_score)
                    print("Your score is:", user_score)

                else:  # if nobody is a winner then its a tie
                    print("Tie!!")

                # checking if anyone reached the score limit
                if self.check_winner(user_score) or \
                        self.check_winner(computer_score):

                    if self.check_winner(user_score):
                        print("\nCongrats you won :)")
                        print("Number of rounds played in total: {}"
                              .format(round_count))
                    else:
                        print("\nComputer won the Game, better luck next time.")
                        print("Number of rounds played in total: {}"
                              .format(round_count))
                    break

                print("\n")

            elif user_select == 'q':  # quit the game at any time
                break
            else:  # for invalid user input
                print("Please type rock, paper, scissors or Q as your input.")

    def replay(self):
        '''player can quit or restart the game after winner is determined'''
        exit_game = False

        while not exit_game:  # infinite loop to play game until user quit the game
            self.play()

            game_status = input(
                "\nTo exit the game enter Q or to restart "
                "enter RS: ").lower().strip()

            if game_status == "q":
                exit_game = True
            elif game_status == "rs":
                exit_game = False
            else:
                print("Please enter Q or RS.")

def main():
    '''main method'''
    game = RockPaperScissor()
```

```
   game.replay()



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

As my Test Driven Development is completed. Now I can run the functional code python file to play the game and check all the conditions and requirements that need to be satisfied. So let's play our Rock Paper Scissor game to see the outputs and results.

```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ /usr/local/bin/python3.9
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTesting/assessment1/rock_paper_scissor.py
Please enter rock or paper or scissors for your choice or Q to quit game: rock
Round: 1
Computer chose: scissor
You win :)
Your score is: 1
Computer's score is: 0


Please enter rock or paper or scissors for your choice or Q to quit game: paper
Round: 2
Computer chose: paper
Tie!!


Please enter rock or paper or scissors for your choice or Q to quit game: rock
Round: 3
Computer chose: paper
Computer win :(
Computer's score is: 1
Your score is: 1


Please enter rock or paper or scissors for your choice or Q to quit game: scissor
Round: 4
Computer chose: paper
You win :)
Your score is: 2
Computer's score is: 1


Please enter rock or paper or scissors for your choice or Q to quit game: paper
Round: 5
Computer chose: rock
You win :)
Your score is: 3
Computer's score is: 1


Please enter rock or paper or scissors for your choice or Q to quit game: rock
Round: 6
Computer chose: rock
Tie!!
```

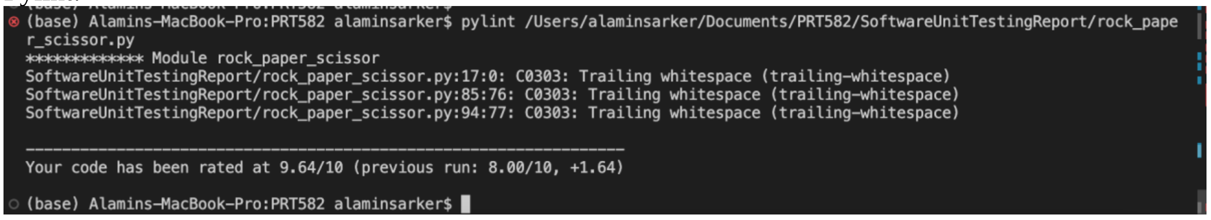
```
Please enter rock or paper or scissors for your choice or Q to quit game: paper
Round: 7
Computer chose: rock
You win :)
Your score is: 4
Computer's score is: 1


Please enter rock or paper or scissors for your choice or Q to quit game: scossor
Please type rock, paper, scissors or Q as your input.
Please enter rock or paper or scissors for your choice or Q to quit game: scissor
Round: 8
Computer chose: paper
You win :)
Your score is: 5
Computer's score is: 1

Congrats you won :)
Number of rounds played in total: 8

To exit the game enter Q or to restart enter RS: q
```

After checking and conducting user acceptance tests several times, I have found that so far all the logic is working fine without any errors or exceptions, above is a sample game results/outputs given.

Step 9: Analyzing code base for coding style and programming errors

Now that development of my rock paper scissor game is done by following TDD, I am going to check my coding style for mistakes or programming errors that I may have made during my TDD process. So I have used the code analyzer Pylint and Flake8 libraries on my code base.

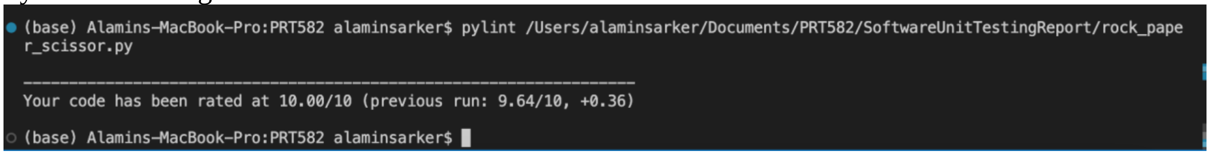
Pylint:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py
************* Module rock_paper_scissor
SoftwareUnitTestingReport/rock_paper_scissor.py:17:0: C0303: Trailing whitespace (trailing-whitespace)
SoftwareUnitTestingReport/rock_paper_scissor.py:85:76: C0303: Trailing whitespace (trailing-whitespace)
SoftwareUnitTestingReport/rock_paper_scissor.py:94:77: C0303: Trailing whitespace (trailing-whitespace)

------------------------------------------------------------------
Your code has been rated at 9.64/10 (previous run: 8.00/10, +1.64)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After running pylint against my functional code base, I can see that my coding style is fine so far except for some trailing whitespaces. Let’s fix that and run pylint again.

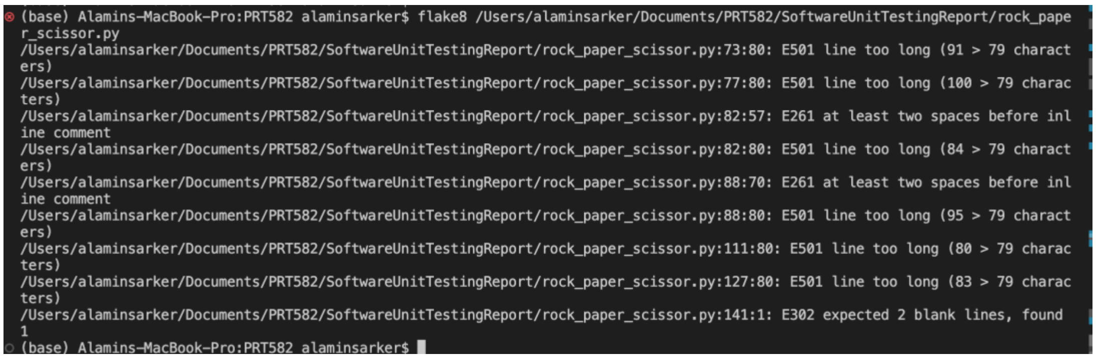
Pylint after fixing the code:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 9.64/10, +0.36)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

I can see that after fixing the whitespace issues, the code base is rated at 10/10. Next, I am going to run Flake8 to check the code base further.

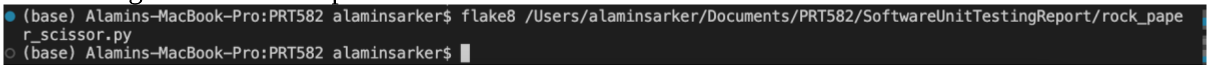
Flake8 output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ flake8 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:73:80: E501 line too long (91 > 79 charact
ers)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:77:80: E501 line too long (100 > 79 charac
ters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:82:57: E261 at least two spaces before inl
ine comment
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:82:80: E501 line too long (84 > 79 charact
ers)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:88:70: E261 at least two spaces before inl
ine comment
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:88:80: E501 line too long (95 > 79 charact
ers)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:111:80: E501 line too long (80 > 79 charac
ters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:127:80: E501 line too long (83 > 79 charac
ters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_paper_scissor.py:141:1: E302 expected 2 blank lines, found
1
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

Now, I am fixing the issues that flake8 captured.

After fixing code flake8 output:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ flake8 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/rock_pape
r_scissor.py
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After solving all the issues regarding the coding style, I ran the flake8 again and got no issues so far. So my code for the rock paper scissor game looks like below.

After fixing the code final version of rock_paper_scissor.py:

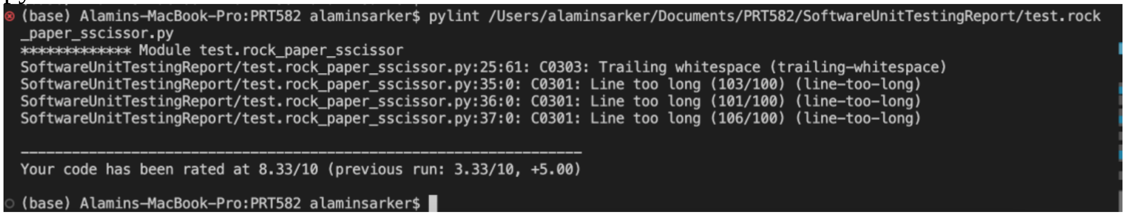
```python
"""Software Unit Testing Report
Scissor Paper Rock game using Test Driven Development"""


import random



class RockPaperScissor():
    '''Rock paper scissor class'''

    choice_list = ["rock", "paper", "scissor"]
    score_limit = 5
    winner = ["user", "computer", "tie"]  # index 0(user), 1(computer), 2(tie)

    def computer_select(self):
        '''computer randomly picks one of the options'''
        return random.choice(self.choice_list)

    @staticmethod
    def user_choice():
        '''player type one of the options of scissor, paper and rock.'''
        user_guess = input(
            "Please enter rock or paper or scissors "
            "for your choice or Q to quit game: ").lower().strip()
        return user_guess

    @staticmethod
    def add_point_to_winner(current_score):
        '''One point is given to the winner'''
        current_score += 1
        return current_score
```

```python
def check_winner(self, current_score):
    '''The first to get five points wins the game.'''
    point_reached = False
    if current_score == self.score_limit:
        point_reached = True
    return point_reached

def determine_winner(self, user_choice, computer_choice):
    '''compare selections and determine who the winner is'''
    new_winner = self.winner[2]

    if computer_choice == "paper" and user_choice == "rock":
        new_winner = self.winner[1]

    elif computer_choice == "rock" and user_choice == "paper":
        new_winner = self.winner[0]

    elif computer_choice == "rock" and user_choice == "scissor":
        new_winner = self.winner[1]

    elif computer_choice == "scissor" and user_choice == "rock":
        new_winner = self.winner[0]

    elif computer_choice == "paper" and user_choice == "scissor":
        new_winner = self.winner[0]

    elif computer_choice == "scissor" and user_choice == "paper":
        new_winner = self.winner[1]
    else:
        new_winner = self.winner[2]

    return new_winner

def play(self):
    '''playing game with logic and printing results/outputs'''

    user_score = 0
    computer_score = 0
    round_count = 0

    # loop until score limit(5) is reached
    while user_score != self.score_limit or \
            computer_score != self.score_limit:

        user_select = self.user_choice()  # user choice as input

        # checking if user choice is rock, paper or scissor
        if user_select in self.choice_list:

            round_count += 1
            print("Round:", round_count)
            # getting computer's choice
            computer_select = self.computer_select()
            print("Computer chose:", computer_select)
            # determining the winner
            winner = self.determine_winner(user_select, computer_select)

            if winner == "user":
                # adding score for winner #noqa: E501
                user_score = self.add_point_to_winner(user_score)
                print("You win :)")
                print("Your score is:", user_score)
                print("Computer's score is:", computer_score)

            elif winner == "computer":
                computer_score = self.add_point_to_winner(computer_score)
                print("Computer win :(")
                print("Computer's score is:", computer_score)
                print("Your score is:", user_score)

            else:  # if nobody is a winner then its a tie
                print("Tie!!")

            # checking if anyone reached the score limit
            if self.check_winner(user_score) or \
                    self.check_winner(computer_score):

                if self.check_winner(user_score):
                    print("\nCongrats you won :)")
                    print("Number of rounds played in total: {}"
                          .format(round_count))
                else:
                    print("\nComputer won the Game, better luck next time.")  # noqa: E501
                    print("Number of rounds played in total: {}"
                          .format(round_count))
                break

            print("\n")

        elif user_select == 'q':  # quit the game at any time
            break
        else:  # for invalid user input
            print("Please type rock, paper, scissors or Q as your input.")

def replay(self):
    '''player can quit or restart the game after winner is determined'''
    exit_game = False

    while not exit_game:  # paly until user quit the game
        self.play()
```

```
            game_status = input(
                "\nTo exit the game enter Q or to restart "
                "enter RS: ").lower().strip()

            if game_status == "q":
                exit_game = True
            elif game_status == "rs":
                exit_game = False
            else:
                print("Please enter Q or RS.")


def main():
    '''main method'''
    game = RockPaperScissor()
    game.replay()


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

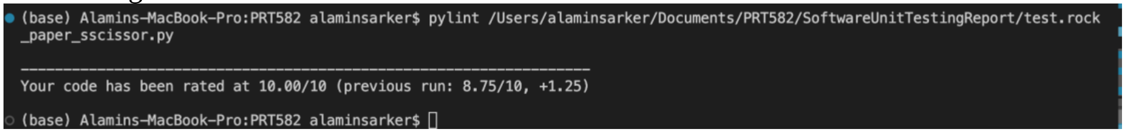
I am going to check for the standard coding style and programming errors of my unittest code as well. So I did so and below are the results of pylint and flake8.

pylint for unittest code:



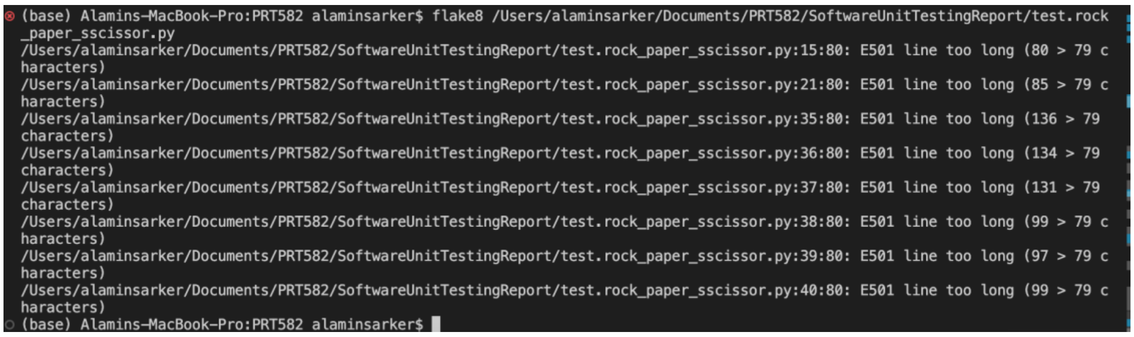
```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock
_paper_sscissor.py
************* Module test.rock_paper_sscissor
SoftwareUnitTestingReport/test.rock_paper_sscissor.py:25:61: C0303: Trailing whitespace (trailing-whitespace)
SoftwareUnitTestingReport/test.rock_paper_sscissor.py:35:0: C0301: Line too long (103/100) (line-too-long)
SoftwareUnitTestingReport/test.rock_paper_sscissor.py:36:0: C0301: Line too long (101/100) (line-too-long)
SoftwareUnitTestingReport/test.rock_paper_sscissor.py:37:0: C0301: Line too long (106/100) (line-too-long)

------------------------------------------------------------------
Your code has been rated at 8.33/10 (previous run: 3.33/10, +5.00)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After fixing:



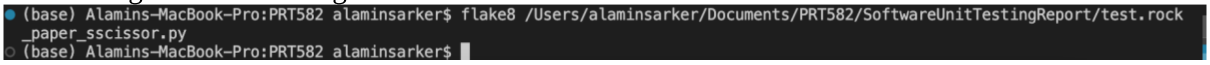
```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ pylint /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock
_paper_sscissor.py

-------------------------------------------------------------------
Your code has been rated at 10.00/10 (previous run: 8.75/10, +1.25)

(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

Flake8 errors:



```
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$ flake8 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock
_paper_sscissor.py
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:15:80: E501 line too long (80 > 79 c
haracters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:21:80: E501 line too long (85 > 79 c
haracters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:35:80: E501 line too long (136 > 79
characters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:36:80: E501 line too long (134 > 79
characters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:37:80: E501 line too long (131 > 79
characters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:38:80: E501 line too long (99 > 79 c
haracters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:39:80: E501 line too long (97 > 79 c
haracters)
/Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock_paper_sscissor.py:40:80: E501 line too long (99 > 79 c
haracters)
(base) Alamins-MacBook-Pro:PRT582 alaminsarker$
```

After fixing and commenting:



```
● (base) Alamins-MacBook-Pro:PRT582 alaminsarker$ flake8 /Users/alaminsarker/Documents/PRT582/SoftwareUnitTestingReport/test.rock
  _paper_sscissor.py
○ (base) Alamins-MacBook-Pro:PRT582 alaminsarker$ 
```

Conclusion:

The main benefit of the TDD approach that I experienced during my development process is fewer bugs, defects and errors in my code because testing first enabled me to evaluate my system and its components with the intent to find whether it satisfies the specified requirements or not. So at the end of the development of my software, my code has fewer bugs, and as a result, I spent less time fixing them than other programming methodologies that are not test-driven. Finally, TDD allows me to produce a higher overall test coverage, and therefore I was able to develop a better quality final product.

GitHub link: https://github.com/alamin-sarker/SoftwareUnitTesting/tree/main/assessment1