10/14/2022

Software Unit Testing Report

Scissor Paper Rock game using TDD in Python Programming Language

Manan Amitkumar Patel
STUDENT ID: S361298
CAMPUS: CASURINA (INTERNAL STUDENT)

Table of Contents

1. Introduction:

The main objective of the task is to develop ROCK-PAPER-SCISSORS game in python using automate unit testing environment. There are several options and rounds in the game until game finished. Firstly, user will enter their name and choose any one option among rock, paper, or scissors then computer will choose any option randomly. Secondly, the process of the game will match the options choose by human player and the computer, if options matched then the round will draw otherwise 1 point will be given to any winner either computer or human. Moreover, after each round the round and score board of the game will be updated by the process. Furthermore, the one who will secure 5 points first, will be the winner of the game. Player can quite the game too any time during the game by choosing the quite option. Finally, after the end of the game there will be option for the user to restart or quite the game.

Demonstrating the phrase automated testing, which covers the automatic testing of the application, in which the developer will construct a unique and different script for the testing file using various automated testing tools. Following the creation of the testing file, the programmed file will automatically test the application and display the results. This is the basic principle of an automated testing tool.

In this project, I am using unit testing, which will automatically test the functioning of the game's key functionalities by inserting pass/fail situations into the test cases. Consider the game to be the unit of code. The core capability of unit testing is to test that unit of code using various test cases, analyzing their behavior and results.

We need to import the Unit test package module in testing file, which is already in the standard library, to do unit testing in Python. We don't need to install any other modules . Each of the test cases has been expanded further with appropriate screenshots.

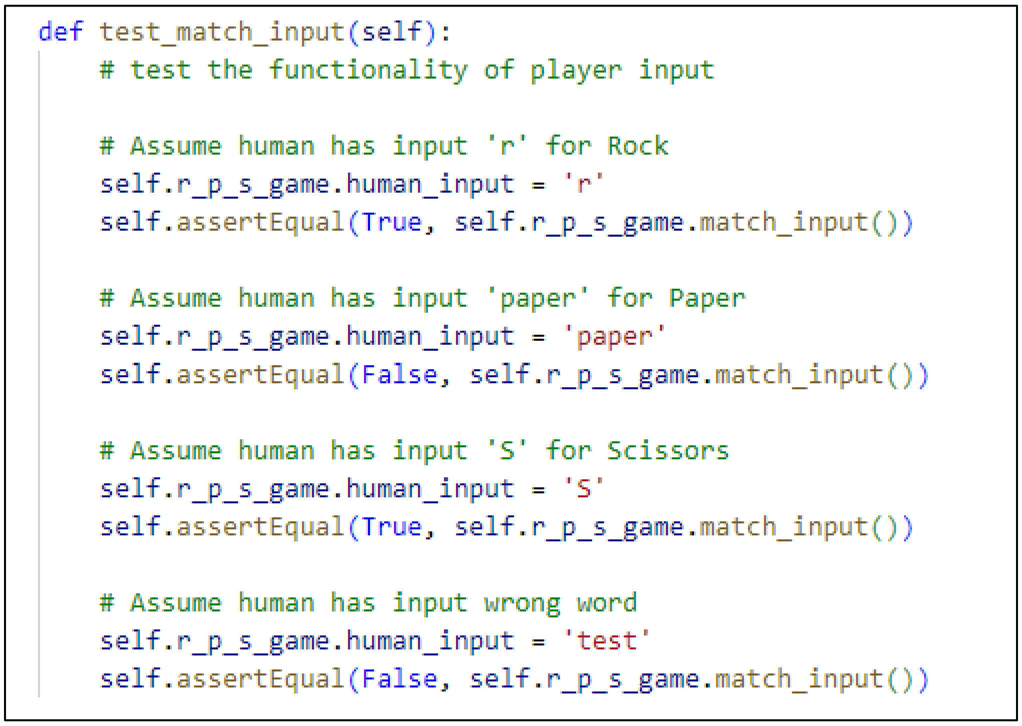
2. Process:

Rock-Paper-Scissors game has been developed using the TDD (test driven development) in python programming language by using the following steps and conditions.

I've written different functions for the major functionality of the game in the R_S_P_GAME.py file which is the major game coding file and I've also written the test cases of these major functionalities into the other unit testing file named as Testing.py. Testing.py file will automatically test the behavior and functionality of the game as I've written different test cases with different inputs and also provided the pass and failure scenario.

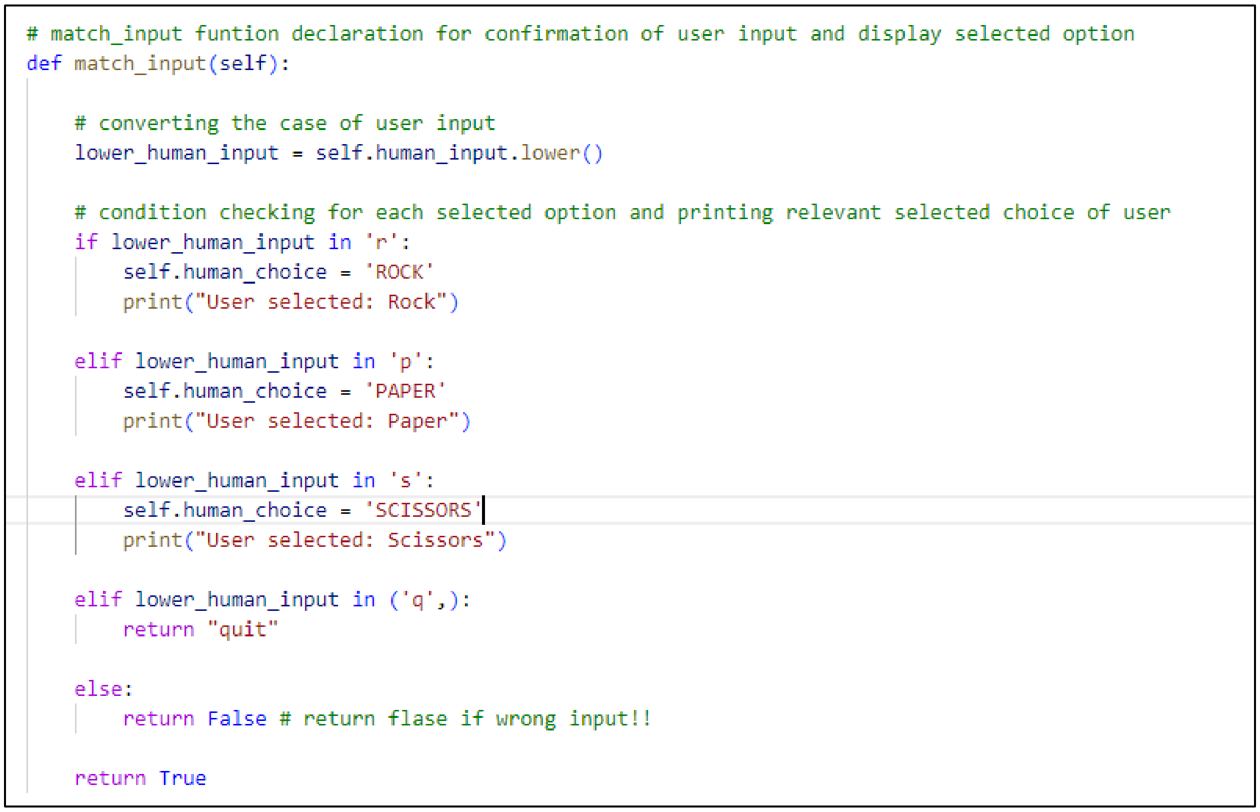
2.1 Functionality 1:

User will enter their name and then choose the option from the given three options Rock, Paper, or Scissors by enter the R, P or S shortcuts or user will select the Q/q to quite the game. Then, to verify the input of the user the function match input is used. This function will assign the proper full form of the matched input for example is user has chosen R then this function will assign the Rock option to the player option variable and same scenario for all other options. One interesting thing is that the icon of the user choose option will display at the same time for better results and understating of the user. If user entered q/Q, then the option will be assigned by the quiet and the game will stop. Simultaneously, the testing file will also verify the functionality of the match input function by providing the different input options through the testing file and check the response of the function automatically thorough coding. Here are the screen shots of both game and testing files



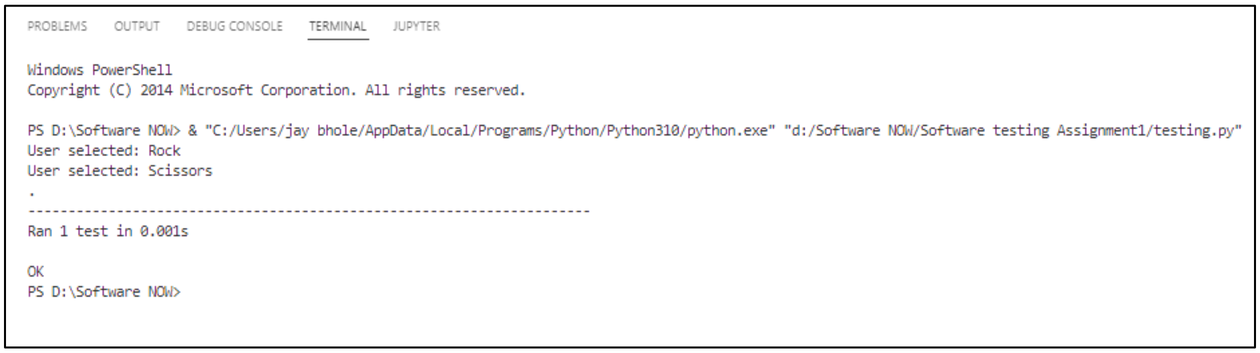
```python
def test_match_input(self):
    # test the functionality of player input

    # Assume human has input 'r' for Rock
    self.r_p_s_game.human_input = 'r'
    self.assertEqual(True, self.r_p_s_game.match_input())

    # Assume human has input 'paper' for Paper
    self.r_p_s_game.human_input = 'paper'
    self.assertEqual(False, self.r_p_s_game.match_input())

    # Assume human has input 'S' for Scissors
    self.r_p_s_game.human_input = 'S'
    self.assertEqual(True, self.r_p_s_game.match_input())

    # Assume human has input wrong word
    self.r_p_s_game.human_input = 'test'
    self.assertEqual(False, self.r_p_s_game.match_input())
```

Figure 1: test_match_input test case from testing.py file



```python
# match_input funtion declaration for confirmation of user input and display selected option
def match_input(self):

    # converting the case of user input
    lower_human_input = self.human_input.lower()

    # condition checking for each selected option and printing relevant selected choice of user
    if lower_human_input in 'r':
        self.human_choice = 'ROCK'
        print("User selected: Rock")

    elif lower_human_input in 'p':
        self.human_choice = 'PAPER'
        print("User selected: Paper")

    elif lower_human_input in 's':
        self.human_choice = 'SCISSORS'
        print("User selected: Scissors")

    elif lower_human_input in ('q',):
        return "quit"

    else:
        return False # return flase if wrong input!!

    return True
```

Figure 2: match_input functionality from the coding game



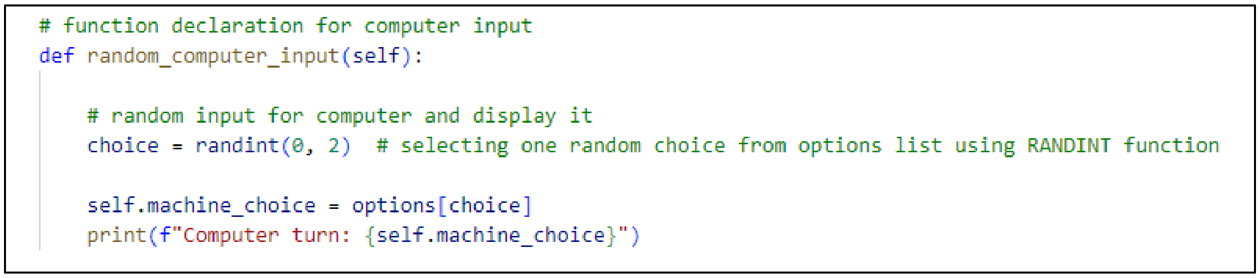
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
User selected: Rock
User selected: Scissors
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
PS D:\Software NOW>
```

Figure 3: Testcase scenario of user entering proper input of choice

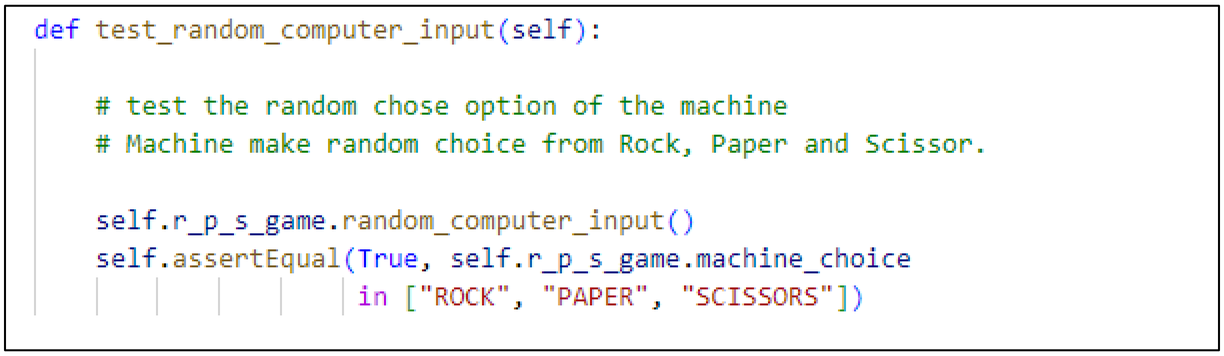
2.2 Functionality 2:

The second functionality is for computer to choose the option randomly from the given option range. For this purpose, random library is used. For this purpose, I've created a list of options for computer [Rock, Paper, and Scissors]. In the random computer input function from 0 to 2 any random number is chosen and stored into separate variable and according to that random choose number the value is assigned to the computer option variable from the list and also displayed the icon of the option choose by the computer to the user. Now, user have both the icons in front of him/her. User will better understand the reason behind their win or loss. On the other hand, in the testing file the test case of the random computer input will verify if the computer chooses their input from Rock, Paper and Scissors with the help of python unit test cases. Here are the screen shots from both the files.



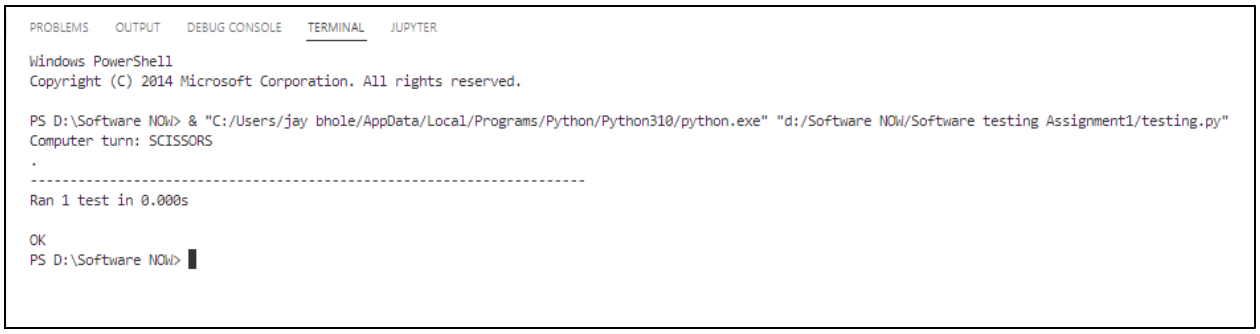
```
# function declaration for computer input
def random_computer_input(self):

    # random input for computer and display it
    choice = randint(0, 2)  # selecting one random choice from options list using RANDINT function

    self.machine_choice = options[choice]
    print(f"Computer turn: {self.machine_choice}")
```

Figure 4: game function of random input from computer



```python
def test_random_computer_input(self):

    # test the random chose option of the machine
    # Machine make random choice from Rock, Paper and Scissor.

    self.r_p_s_game.random_computer_input()
    self.assertEqual(True, self.r_p_s_game.machine_choice
                        in ["ROCK", "PAPER", "SCISSORS"])
```

Figure 5: test case to verify the random input of the computer



```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
Computer turn: SCISSORS
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS D:\Software NOW>
```

Figure 6: Output of the testing case

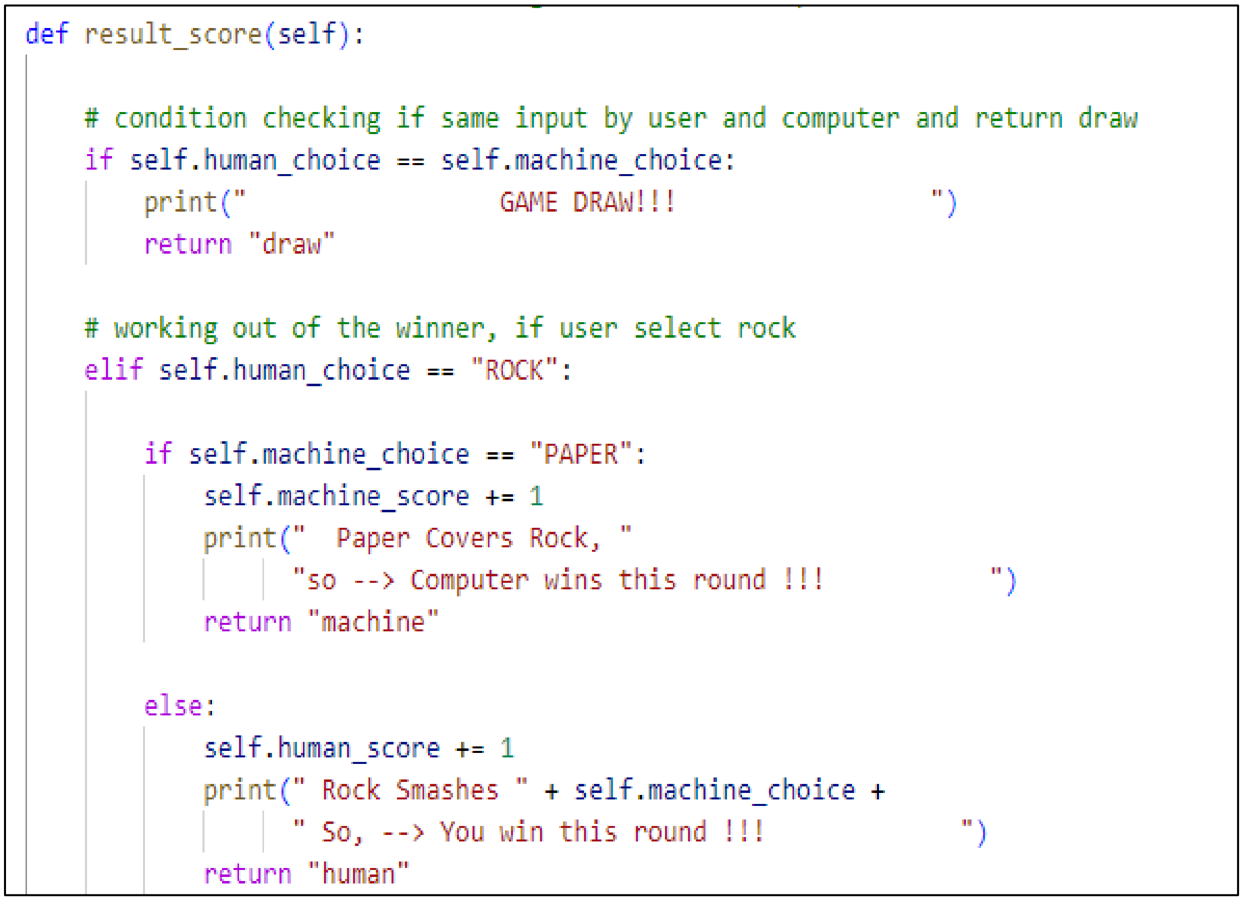
2.3 Functionality 3:

After input process now the game will decide the winner of the first round of the game there will be unlimited rounds of the game until anyone will win the game user or computer. For this purpose, result score function is written by me. This function will match the input of the user and computer if both the input matched then the round will be drawn, and no one earned the point just next round will start. The logic behind the loose and win of the user or computer

- If user select rock and computer, select scissor then Rock wins.
- If user select paper and computer, select rock, then paper wins.
- If user select scissor and computer select paper, then scissor wins.

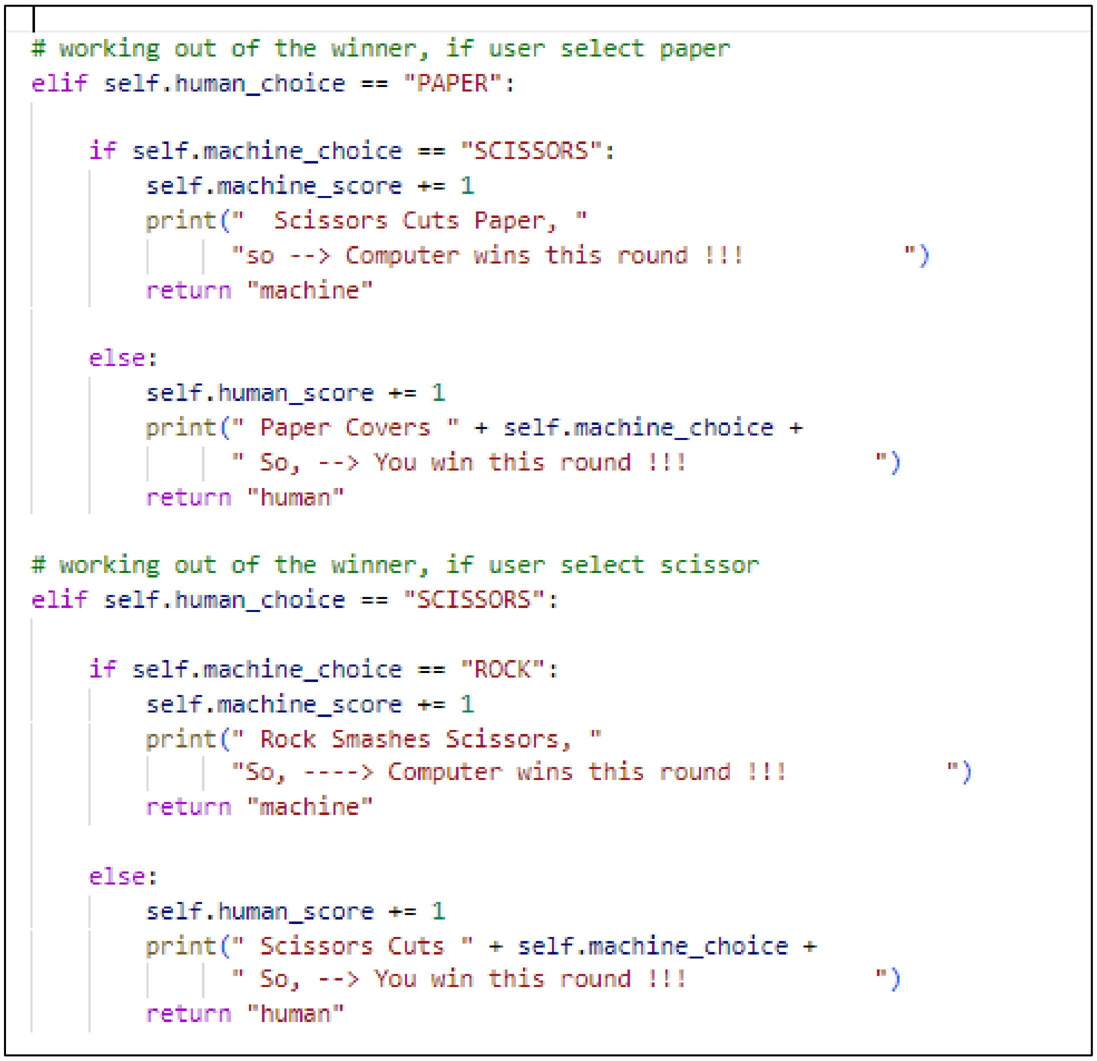
According to the given scenario the user or computer will win the round and the points will add to the winter side until anyone reached to the 5 points the rounds will be jumped too next. In the testing file the test each round winner test is written to test the automatic functionality and output of the function in which I've provided the different input options for both user and computer and test the function by providing the true and false required option into the test cases. If the output is desired, then the test case is passed, and the function is accurate otherwise there will be error in the coding.

Here is the coding from both the files:



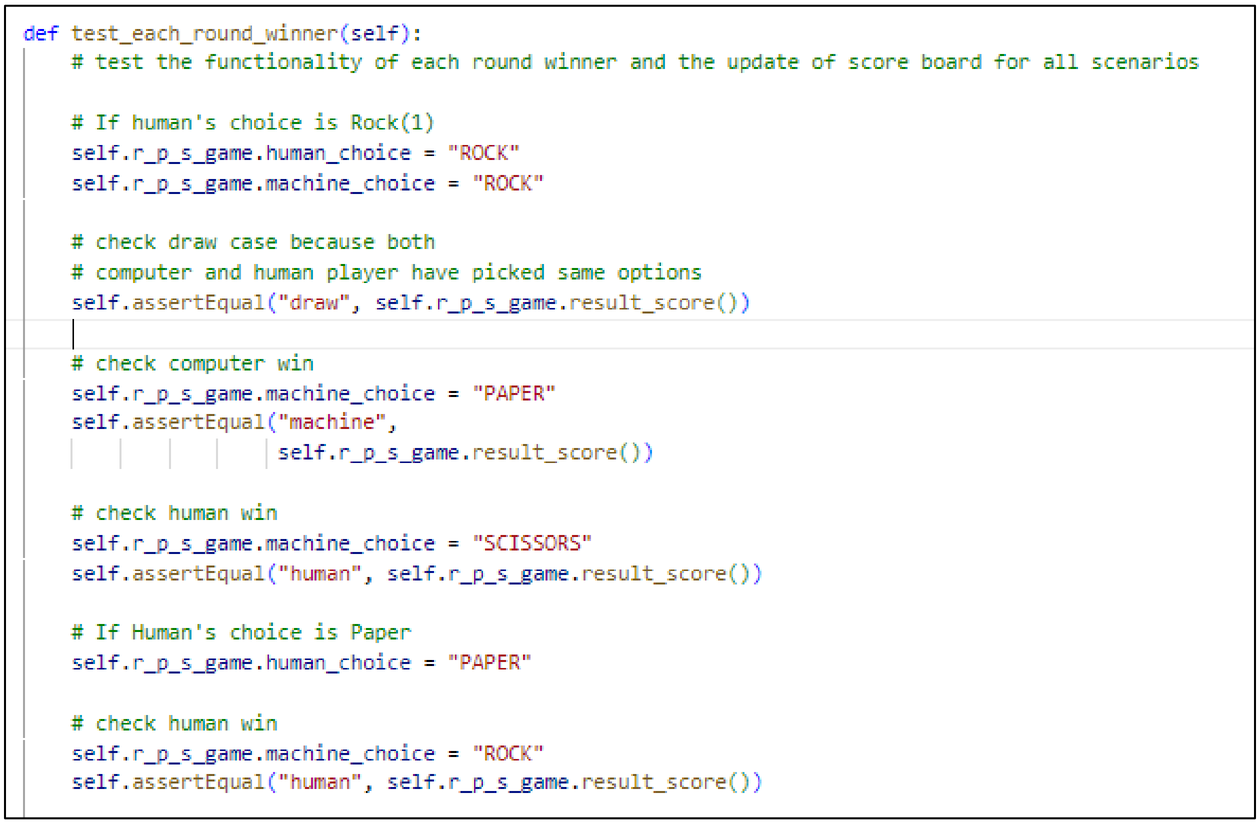
```python
def result_score(self):

    # condition checking if same input by user and computer and return draw
    if self.human_choice == self.machine_choice:
        print("                  GAME DRAW!!!                  ")
        return "draw"

    # working out of the winner, if user select rock
    elif self.human_choice == "ROCK":

        if self.machine_choice == "PAPER":
            self.machine_score += 1
            print("  Paper Covers Rock, "
                  "so --> Computer wins this round !!!          ")
            return "machine"

        else:
            self.human_score += 1
            print(" Rock Smashes " + self.machine_choice +
                  " So, --> You win this round !!!             ")
            return "human"
```

Figure 7.1



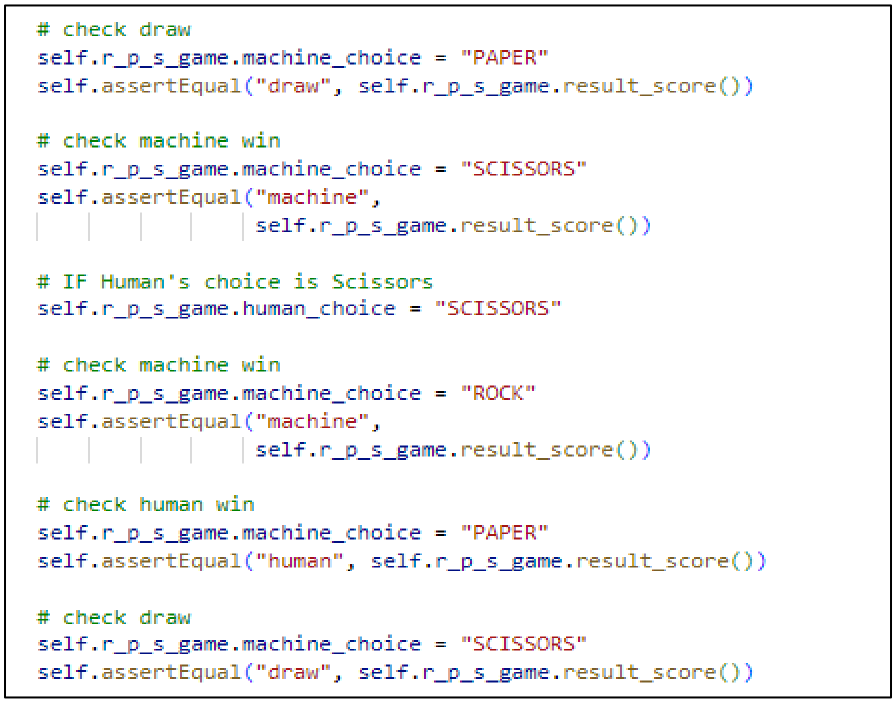
```
# working out of the winner, if user select paper
elif self.human_choice == "PAPER":

    if self.machine_choice == "SCISSORS":
        self.machine_score += 1
        print("  Scissors Cuts Paper, "
              "so --> Computer wins this round !!!           ")
        return "machine"

    else:
        self.human_score += 1
        print(" Paper Covers " + self.machine_choice +
              " So, --> You win this round !!!             ")
        return "human"

# working out of the winner, if user select scissor
elif self.human_choice == "SCISSORS":

    if self.machine_choice == "ROCK":
        self.machine_score += 1
        print(" Rock Smashes Scissors, "
              "So, ----> Computer wins this round !!!           ")
        return "machine"

    else:
        self.human_score += 1
        print(" Scissors Cuts " + self.machine_choice +
              " So, --> You win this round !!!             ")
        return "human"
```

Figure 7.2: checking functionality from the game coding



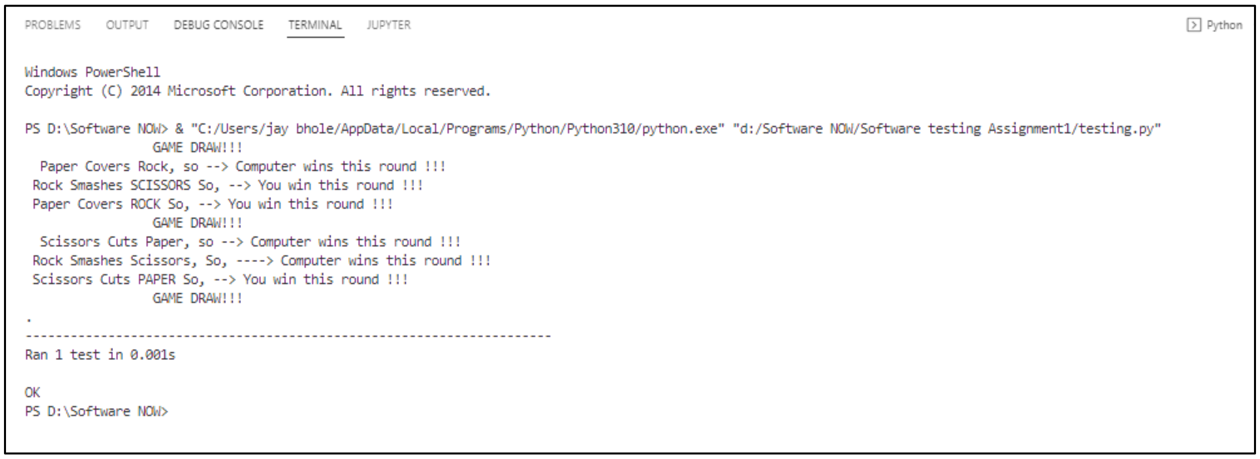
```python
def test_each_round_winner(self):
    # test the functionality of each round winner and the update of score board for all scenarios

    # If human's choice is Rock(1)
    self.r_p_s_game.human_choice = "ROCK"
    self.r_p_s_game.machine_choice = "ROCK"

    # check draw case because both
    # computer and human player have picked same options
    self.assertEqual("draw", self.r_p_s_game.result_score())

    # check computer win
    self.r_p_s_game.machine_choice = "PAPER"
    self.assertEqual("machine",
                     self.r_p_s_game.result_score())

    # check human win
    self.r_p_s_game.machine_choice = "SCISSORS"
    self.assertEqual("human", self.r_p_s_game.result_score())

    # If Human's choice is Paper
    self.r_p_s_game.human_choice = "PAPER"

    # check human win
    self.r_p_s_game.machine_choice = "ROCK"
    self.assertEqual("human", self.r_p_s_game.result_score())
```

Figure 8.1



```python
    # check draw
    self.r_p_s_game.machine_choice = "PAPER"
    self.assertEqual("draw", self.r_p_s_game.result_score())

    # check machine win
    self.r_p_s_game.machine_choice = "SCISSORS"
    self.assertEqual("machine",
                     self.r_p_s_game.result_score())

    # IF Human's choice is Scissors
    self.r_p_s_game.human_choice = "SCISSORS"

    # check machine win
    self.r_p_s_game.machine_choice = "ROCK"
    self.assertEqual("machine",
                     self.r_p_s_game.result_score())

    # check human win
    self.r_p_s_game.machine_choice = "PAPER"
    self.assertEqual("human", self.r_p_s_game.result_score())

    # check draw
    self.r_p_s_game.machine_choice = "SCISSORS"
    self.assertEqual("draw", self.r_p_s_game.result_score())
```

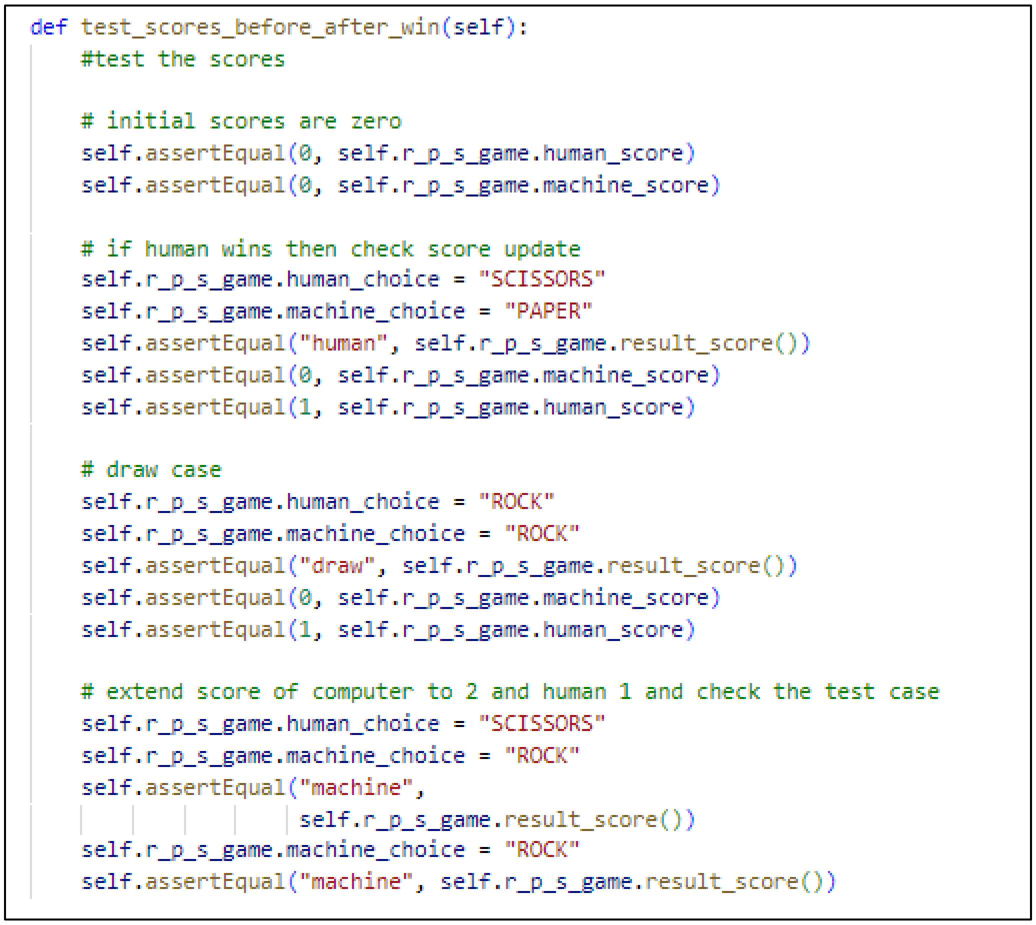
Figure 8.2: checking functionality from test case



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER Python

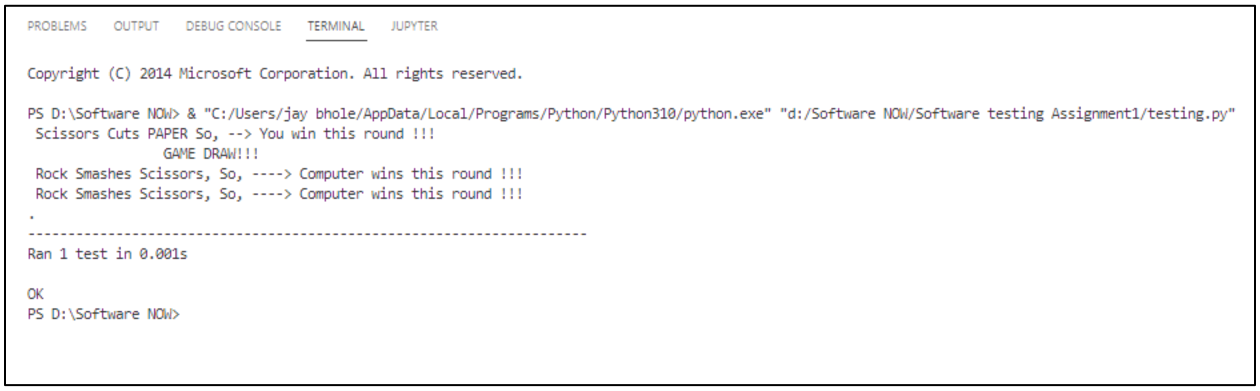
```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
                 GAME DRAW!!!
  Paper Covers Rock, so --> Computer wins this round !!!
 Rock Smashes SCISSORS So, --> You win this round !!!
 Paper Covers ROCK So, --> You win this round !!!
                 GAME DRAW!!!
  Scissors Cuts Paper, so --> Computer wins this round !!!
 Rock Smashes Scissors, So, ----> Computer wins this round !!!
 Scissors Cuts PAPER So, --> You win this round !!!
                 GAME DRAW!!!
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
PS D:\Software NOW>
```

Figure 9: test case result of each input and possibility rounds



```python
def test_scores_before_after_win(self):
    #test the scores

    # initial scores are zero
    self.assertEqual(0, self.r_p_s_game.human_score)
    self.assertEqual(0, self.r_p_s_game.machine_score)

    # if human wins then check score update
    self.r_p_s_game.human_choice = "SCISSORS"
    self.r_p_s_game.machine_choice = "PAPER"
    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual(0, self.r_p_s_game.machine_score)
    self.assertEqual(1, self.r_p_s_game.human_score)

    # draw case
    self.r_p_s_game.human_choice = "ROCK"
    self.r_p_s_game.machine_choice = "ROCK"
    self.assertEqual("draw", self.r_p_s_game.result_score())
    self.assertEqual(0, self.r_p_s_game.machine_score)
    self.assertEqual(1, self.r_p_s_game.human_score)

    # extend score of computer to 2 and human 1 and check the test case
    self.r_p_s_game.human_choice = "SCISSORS"
    self.r_p_s_game.machine_choice = "ROCK"
    self.assertEqual("machine",
                     self.r_p_s_game.result_score())
    self.r_p_s_game.machine_choice = "ROCK"
    self.assertEqual("machine", self.r_p_s_game.result_score())
```

Figure 10: test scoring functionality before and after winning



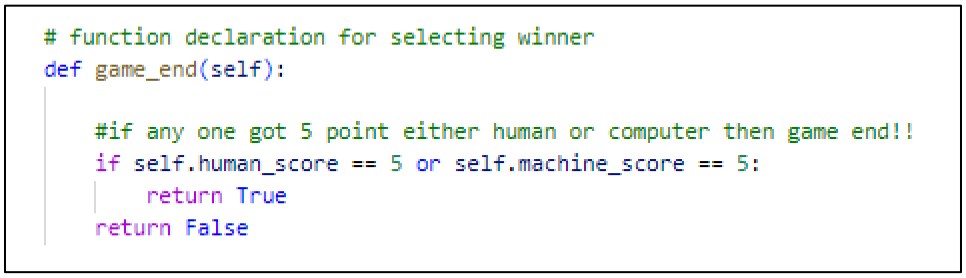
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
 Scissors Cuts PAPER So, --> You win this round !!!
                GAME DRAW!!!
 Rock Smashes Scissors, So, ----> Computer wins this round !!!
 Rock Smashes Scissors, So, ----> Computer wins this round !!!
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
PS D:\Software NOW>
```

Figure 11: checking the scoring functionality

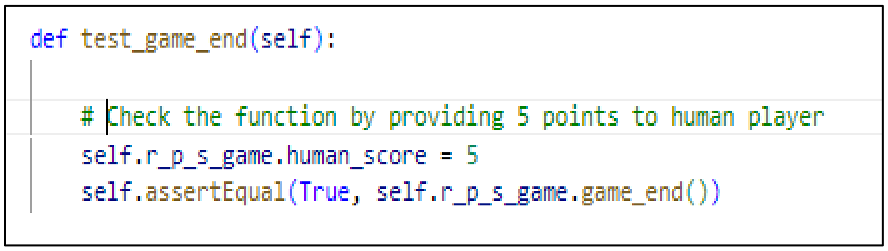
2.4 Functionality 4:

After the round and scoring logic of the game it’s time to decide the final winner of the game. Game end function will return the true if the points of the player or computer has reached the required level which is 5 points. And then the next logic and step of the game will check the response of that function if it returns true then the game will display the winner of the game by matching the points of both player and computer with 5 and display the results. However, in the testing file the response of the game end function is also tested by providing the 5 points to the player and required the output from the function true which is accurate. Also verified the winner of the game in the testing file by providing the winning inputs for any one consistently until the game response winner scenario. Here is the code from both the files



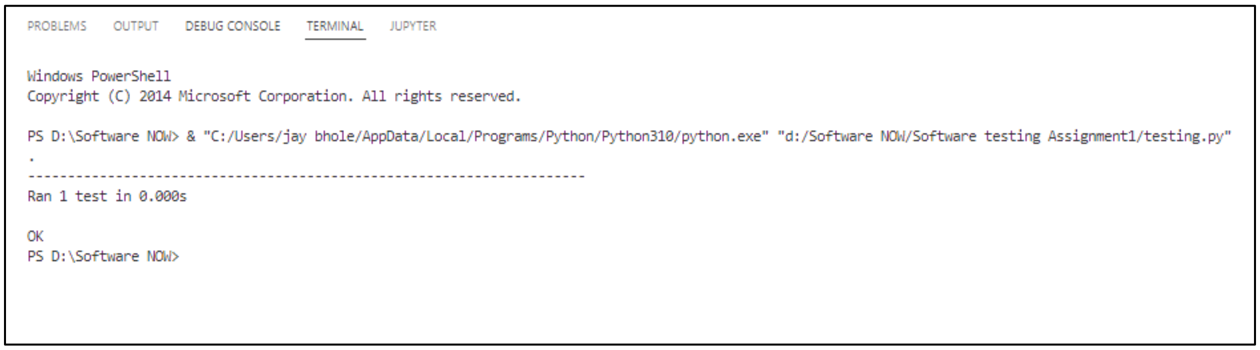
```
# function declaration for selecting winner
def game_end(self):

    #if any one got 5 point either human or computer then game end!!
    if self.human_score == 5 or self.machine_score == 5:
        return True
    return False
```

Figure 12: Game end code in game file



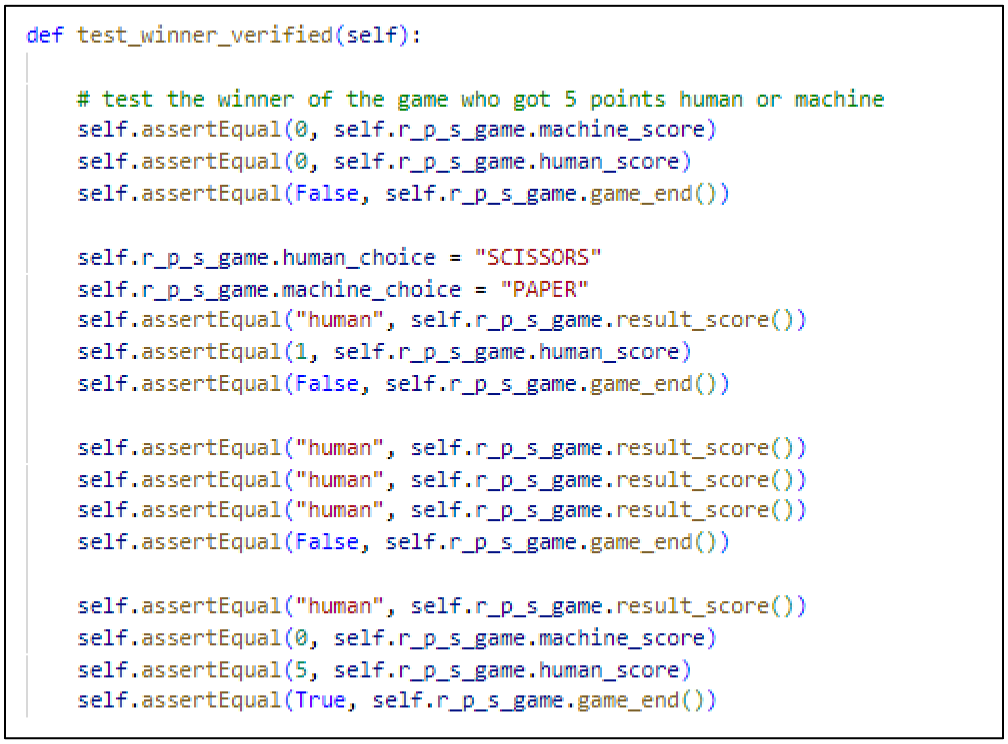
```
def test_game_end(self):

    # Check the function by providing 5 points to human player
    self.r_p_s_game.human_score = 5
    self.assertEqual(True, self.r_p_s_game.game_end())
```

Figure 13: testing game end in test file



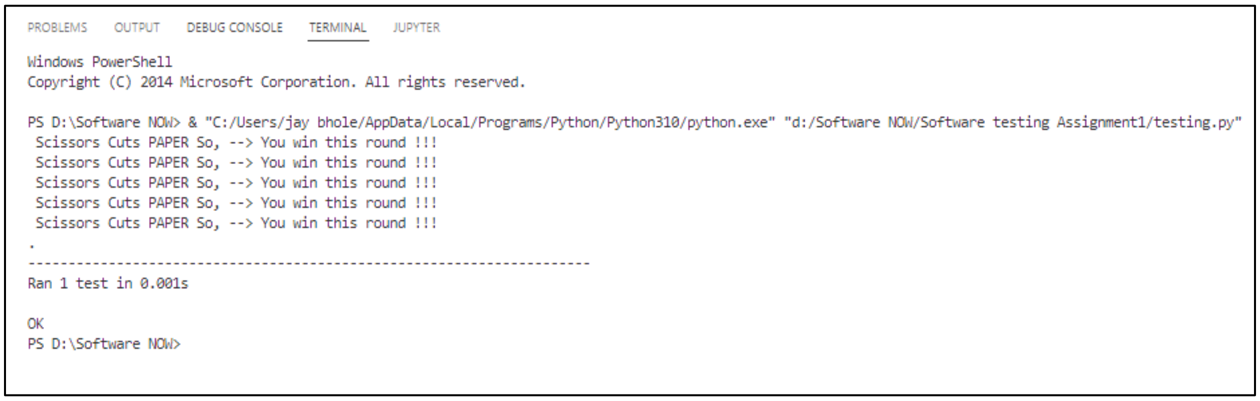
```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS D:\Software NOW>
```

Figure 14: Test case output of end game results



```
def test_winner_verified(self):

    # test the winner of the game who got 5 points human or machine
    self.assertEqual(0, self.r_p_s_game.machine_score)
    self.assertEqual(0, self.r_p_s_game.human_score)
    self.assertEqual(False, self.r_p_s_game.game_end())

    self.r_p_s_game.human_choice = "SCISSORS"
    self.r_p_s_game.machine_choice = "PAPER"
    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual(1, self.r_p_s_game.human_score)
    self.assertEqual(False, self.r_p_s_game.game_end())

    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual(False, self.r_p_s_game.game_end())

    self.assertEqual("human", self.r_p_s_game.result_score())
    self.assertEqual(0, self.r_p_s_game.machine_score)
    self.assertEqual(5, self.r_p_s_game.human_score)
    self.assertEqual(True, self.r_p_s_game.game_end())
```

Figure 15: testing and verifying winner case

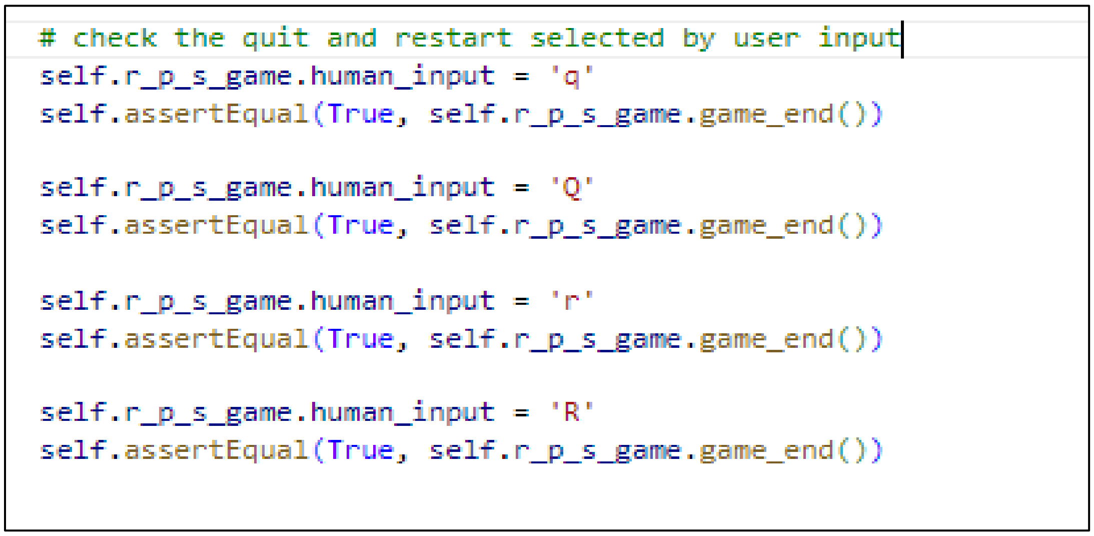


```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
 Scissors Cuts PAPER So, --> You win this round !!!
 Scissors Cuts PAPER So, --> You win this round !!!
 Scissors Cuts PAPER So, --> You win this round !!!
 Scissors Cuts PAPER So, --> You win this round !!!
 Scissors Cuts PAPER So, --> You win this round !!!
.
----------------------------------------------------------------------
Ran 1 test in 0.001s

OK
PS D:\Software NOW>
```

Figure 16: testing of the whole game winner

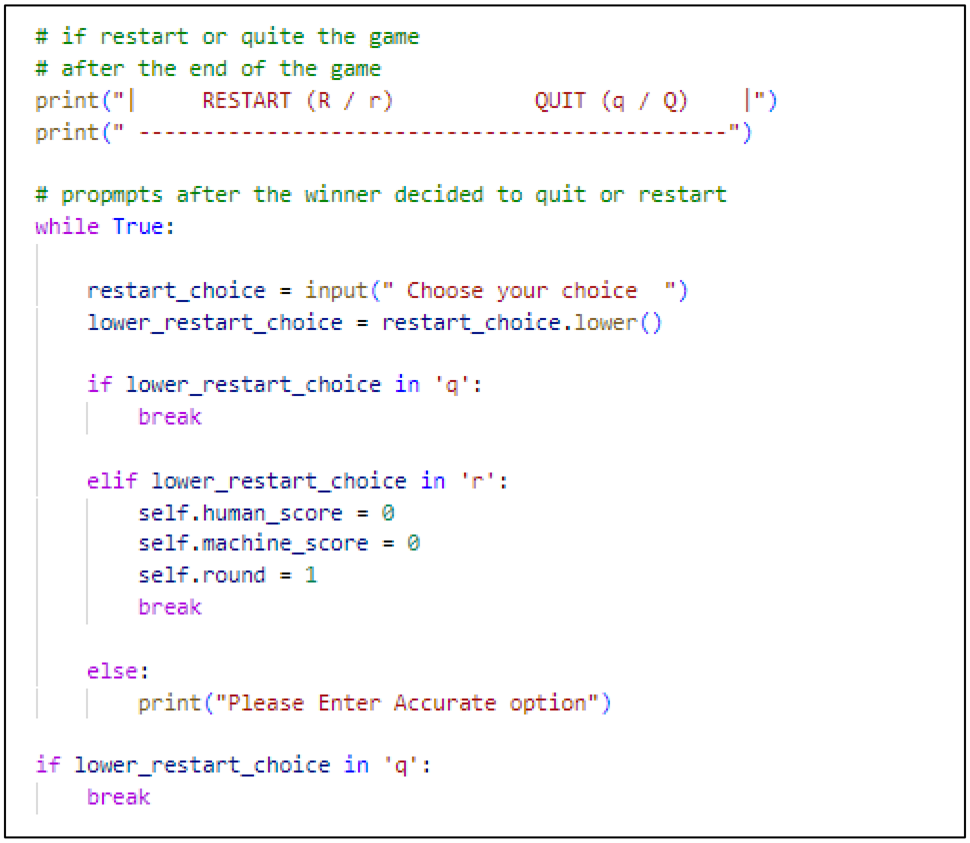
2.5 Functionality 5:

Then finally, there will be option for the user to restart or quite the game after the final results of the game. For this purpose, the user will enter any option restart or quite the game will ask the user to enter the accurate option until user not entered the R or Q/q option R for restart and Q for quite the game. On the basis of the user option the game would proceed further if user entered the R to restart the game, then the game reset all the values of the variables which again starts the game because the game end condition will not satisfy. And if user entered the q option, then the game will stop and closed. Here is logic and code of this functionality.



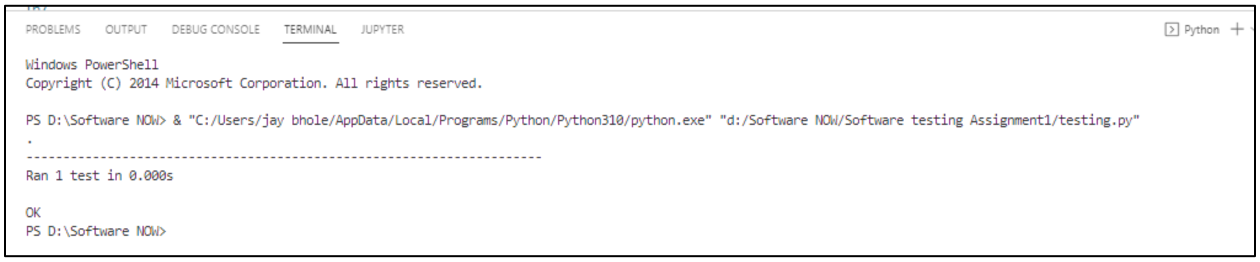
```
# check the quit and restart selected by user input
self.r_p_s_game.human_input = 'q'
self.assertEqual(True, self.r_p_s_game.game_end())

self.r_p_s_game.human_input = 'Q'
self.assertEqual(True, self.r_p_s_game.game_end())

self.r_p_s_game.human_input = 'r'
self.assertEqual(True, self.r_p_s_game.game_end())

self.r_p_s_game.human_input = 'R'
self.assertEqual(True, self.r_p_s_game.game_end())
```

Figure 17: testing file of restart and quit functionality



```
# if restart or quite the game
# after the end of the game
print("|     RESTART (R / r)          QUIT (q / Q)    |")
print(" ---------------------------------------------")

# propmpts after the winner decided to quit or restart
while True:

    restart_choice = input(" Choose your choice  ")
    lower_restart_choice = restart_choice.lower()

    if lower_restart_choice in 'q':
        break

    elif lower_restart_choice in 'r':
        self.human_score = 0
        self.machine_score = 0
        self.round = 1
        break

    else:
        print("Please Enter Accurate option")

if lower_restart_choice in 'q':
    break
```

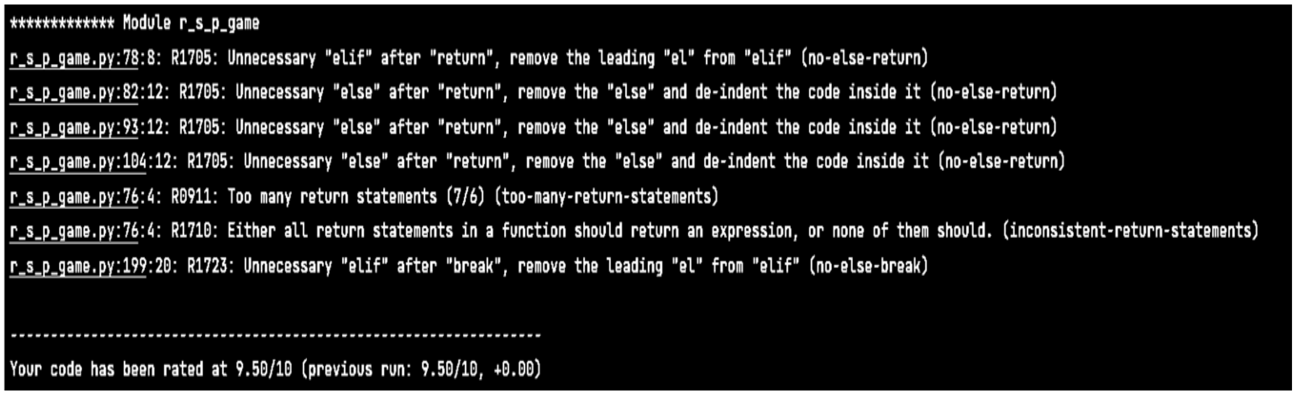
Figure 18: Checking the restart and quit functionality in coding file



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   JUPYTER

Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. All rights reserved.

PS D:\Software NOW> & "C:/Users/jay bhole/AppData/Local/Programs/Python/Python310/python.exe" "d:/Software NOW/Software testing Assignment1/testing.py"
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
PS D:\Software NOW>
```

Figure 19: Testing file of restart and quit result

3. Pylint implementation:



```
************* Module r_s_p_game
r_s_p_game.py:78:8: R1705: Unnecessary "elif" after "return", remove the leading "el" from "elif" (no-else-return)
r_s_p_game.py:82:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
r_s_p_game.py:93:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
r_s_p_game.py:104:12: R1705: Unnecessary "else" after "return", remove the "else" and de-indent the code inside it (no-else-return)
r_s_p_game.py:76:4: R0911: Too many return statements (7/6) (too-many-return-statements)
r_s_p_game.py:76:4: R1710: Either all return statements in a function should return an expression, or none of them should. (inconsistent-return-statements)
r_s_p_game.py:199:20: R1723: Unnecessary "elif" after "break", remove the leading "el" from "elif" (no-else-break)

------------------------------------------------------------------
Your code has been rated at 9.50/10 (previous run: 9.50/10, +0.00)
```

Figure 20: Pylint recommendation of program

My code is 9.5/10 is perfect and accurate according to the Pylint the remaining comments is telling about the return statement in the result score function because each if matching statement is returning the string value which I've used to verify the test cases so therefore these comments are still there.

4. Conclusion:

To reiterate, the complete introduction and the process of the game have been discussed and explained above. Thus, the TDD (test driven development) is very useful for better coding styles and understanding. Because with the help of TDD one can check the generic behavior of their coding without hard coded and one can use their functions as many times as he wants. I've learned many things during the development of this game like loops, if else, complex logic different scenarios and the major thing is that now I’m capable of writing the unit test cases with the code at the same time. The complex thing is that to write such functions which will also satisfy the TDD development mode.