10/14/2022

Software Unit Testing Report

Scissor Paper Rock game using TDD in Python Programming Language

**Table of Contents**

[1. Introduction: 2](#_Toc116609963)

[2. Process: 3](#_Toc116609964)

[2.1 Functionality 1: 3](#_Toc116609965)

[2.2 Functionality 2: 5](#_Toc116609966)

[2.3 Functionality 3: 6](#_Toc116609967)

[2.4 Functionality 4: 11](#_Toc116609968)

[2.5 Functionality 5: 13](#_Toc116609969)

[3. Pylint implementation: 15](#_Toc116609970)

[4. Conclusion: 15](#_Toc116609971)

# **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.

# **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.

## **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

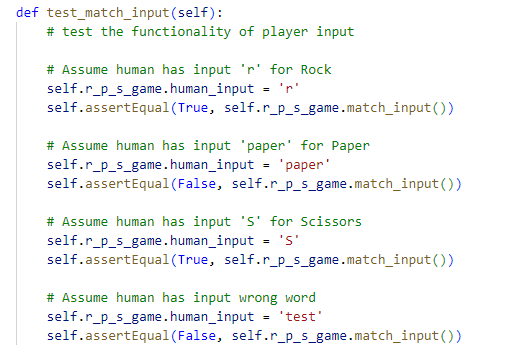


Figure 1: test\_match\_input test case from testing.py file

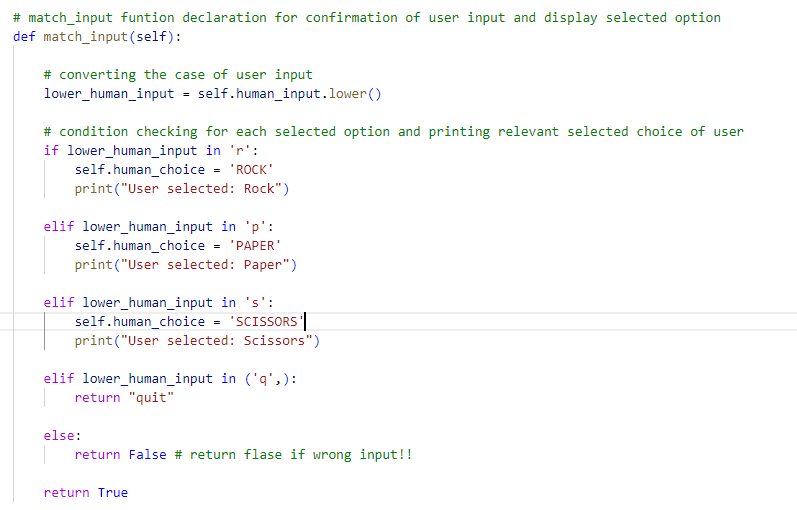


Figure 2: match\_input functionality from the coding game

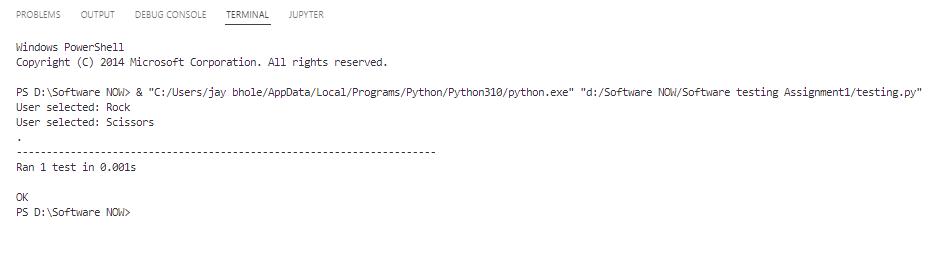


Figure 3: Testcase scenario of user entering proper input of choice

## **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.

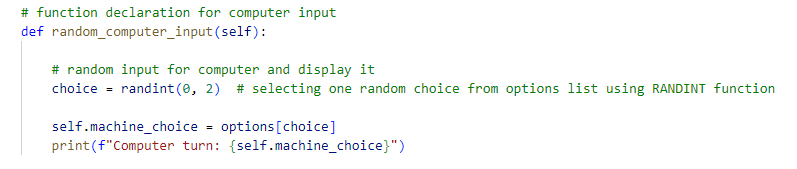


Figure 4: game function of random input from computer

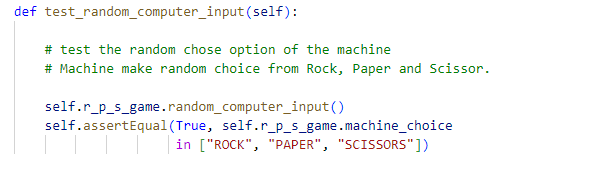


Figure 5: test case to verify the random input of the computer

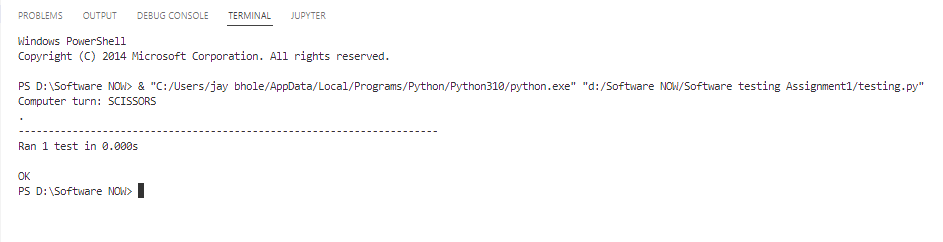


Figure 6: Output of the testing case

## **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:

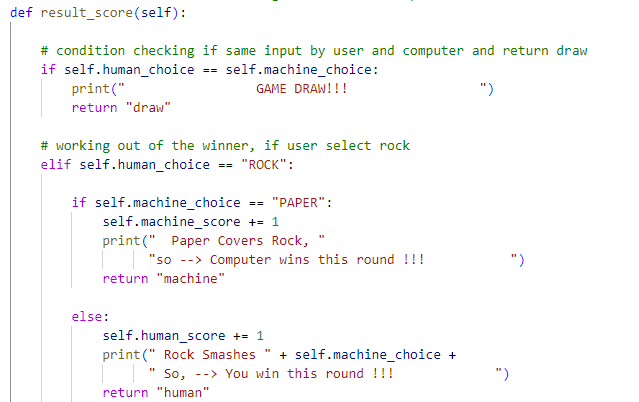


Figure 7.1

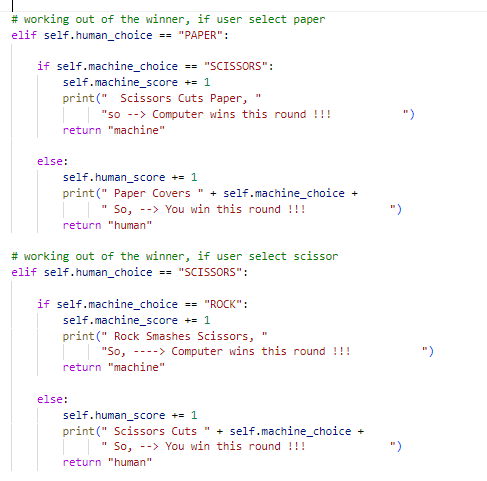


Figure 7.2: checking functionality from the game coding

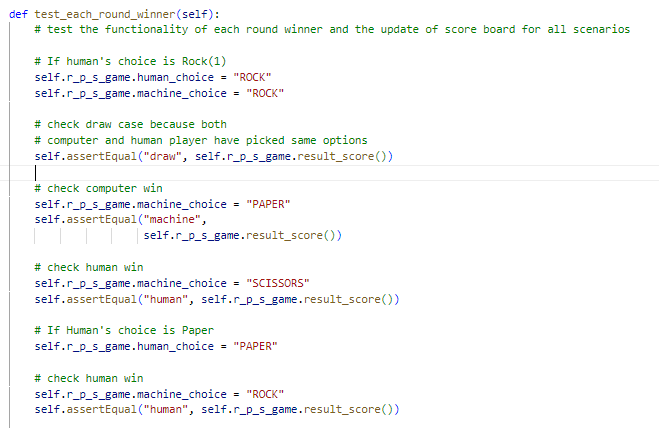


Figure 8.1

Text

Description automatically generated

Figure 8.2: checking functionality from test case

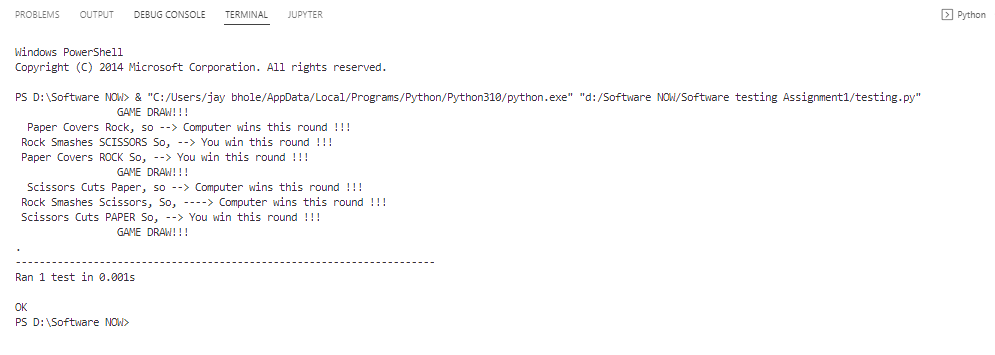


Figure 9: test case result of each input and possibility rounds

Text

Description automatically generated

Figure 10: test scoring functionality before and after winning

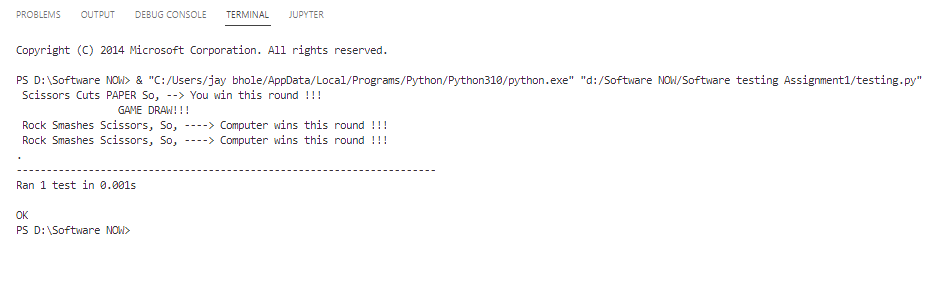


Figure 11: checking the scoring functionality

## **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

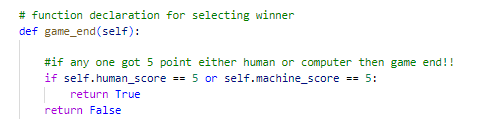


Figure 12: Game end code in game file

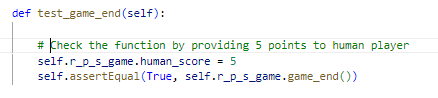


Figure 13: testing game end in test file

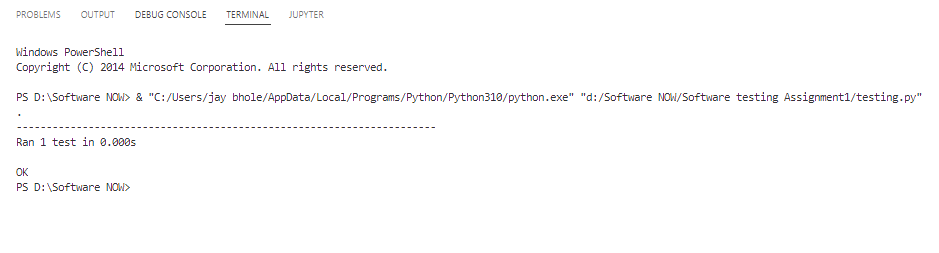


Figure 14: Test case output of end game results

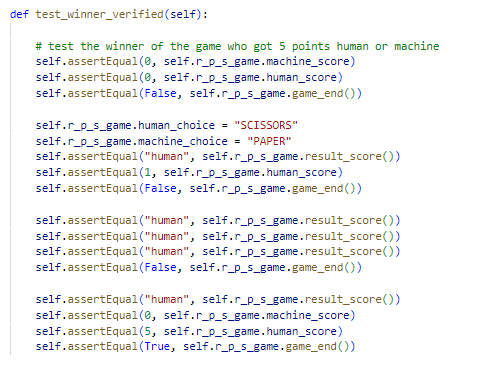


Figure 15: testing and verifying winner case

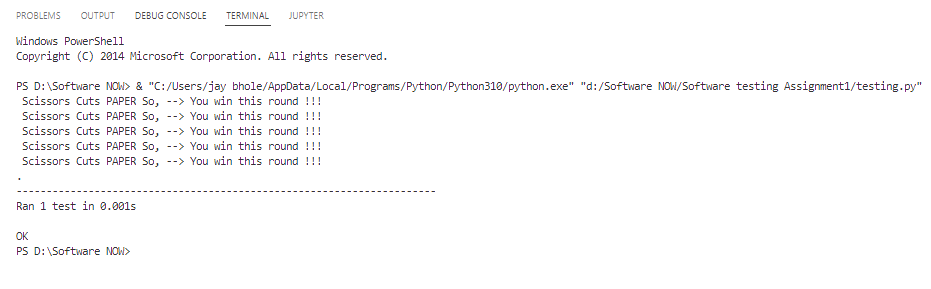


Figure 16: testing of the whole game winner

## **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.

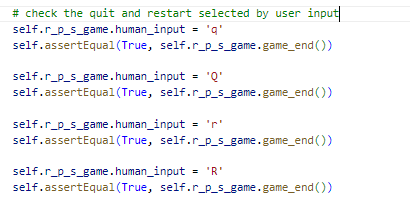


Figure 17: testing file of restart and quit functionality

Text

Description automatically generated

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

Graphical user interface, application

Description automatically generated

Figure 19: Testing file of restart and quit result

# **Pylint implementation:**

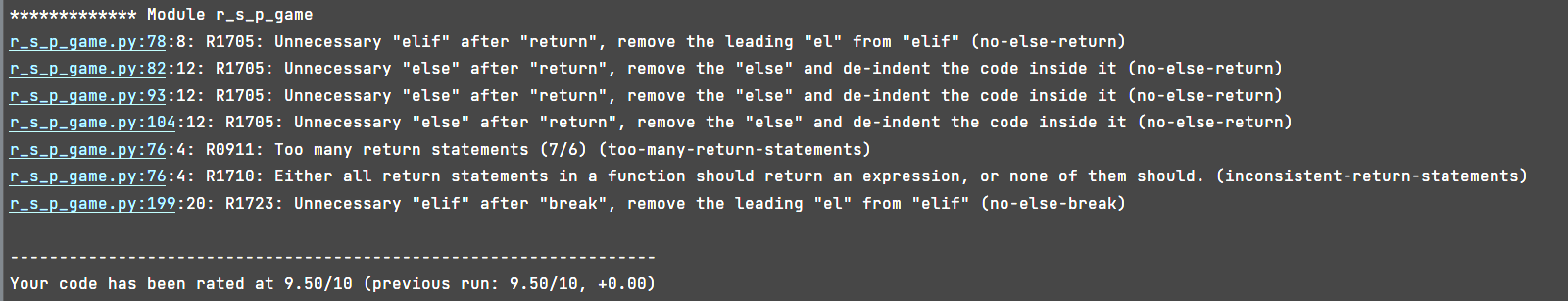


Figure 20: Pylint recommendation of program

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.

# **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.