PRT582 – ASSIGNMENT 1

SOFTWARE UNIT TESTING REPORT

Github link: https://github.com/mentain/PRT582_Assignment/

Table of Contents

INTRODUCTION:

The Scissor Paper Rock game is the type of game in which a player gets to choose one of the options between scissor, paper and rock. This is then compared against the computer's selection and determine who the winner is.

The winning rules are listed below as follows:

- rock vs paper - paper wins
- rock vs scissor - rock wins
- paper vs scissor - scissor wins.

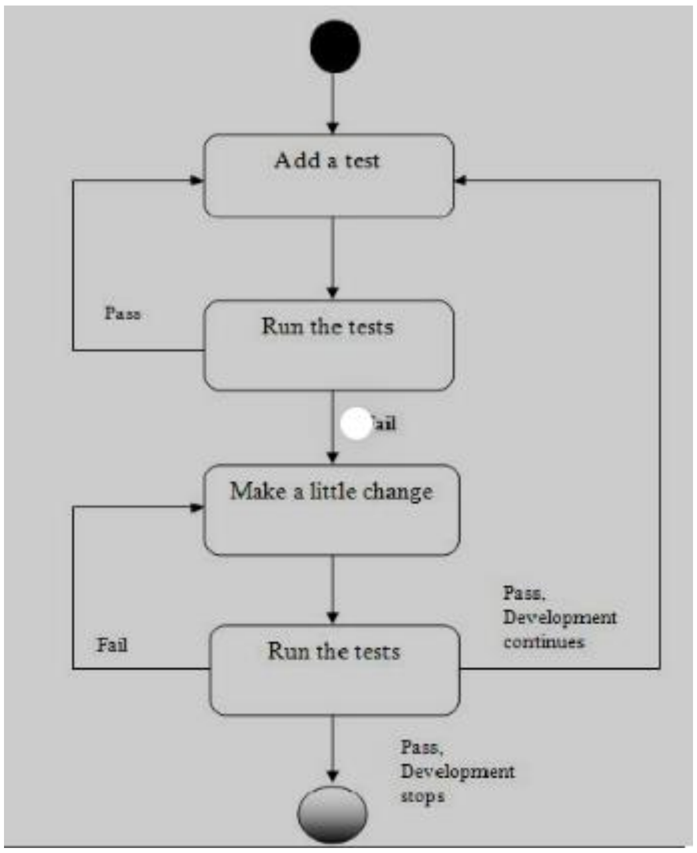
The game will have the following features, which are the requirements:

1. The computer randomly picks one of the options of scissor, paper, and rock.
2. Player is then given the option to pick/type one of the options of scissor, paper, and rock.
3. One point is given to the winner.
4. The first to get five points wins the game. The total number of rounds played in total will also be displayed.
5. Once the winner is determined, the player is asked to quit or restart the game

For this sort of program, I have decided to use Python to build the all the functions of the application. Pytest will be used to test the code automatically and Test-Driven Development Process (TDD) will be applied to specify and validate what the code is expected do.

Specially, the TDD process will be followed by these following steps:

1. Add a Test.
2. Run all tests and see if any new test fails.
3. Write some code.
4. Run tests and Refactor code.
5. Repeat.


Add a test
Run the tests
Pass
Fail
Make a little change
Run the tests
Fail
Pass, Development continues
Pass, Development stops

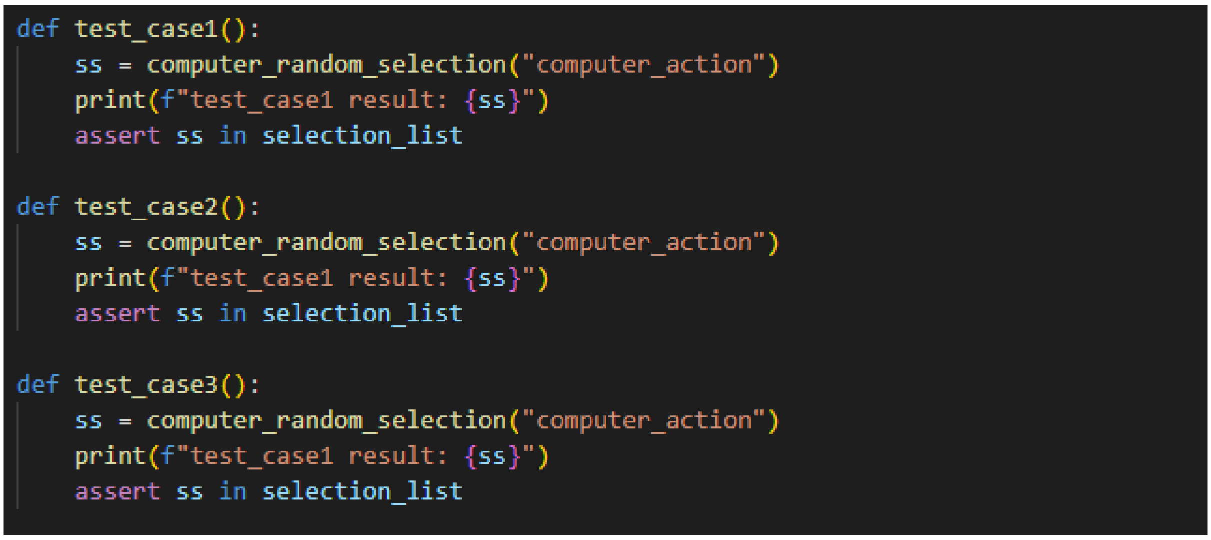
PROCESS:

The Pytest needs to be installed into the local machine before execution.
As we follow the TDD process, the test cases will be created for each specific requirement using Pytest library.

1st Requirement:

For the 1st requirement, the computer randomly picks one of the options of scissor, paper, and rock. Therefore, I created the test cases for the requirement as shown below:

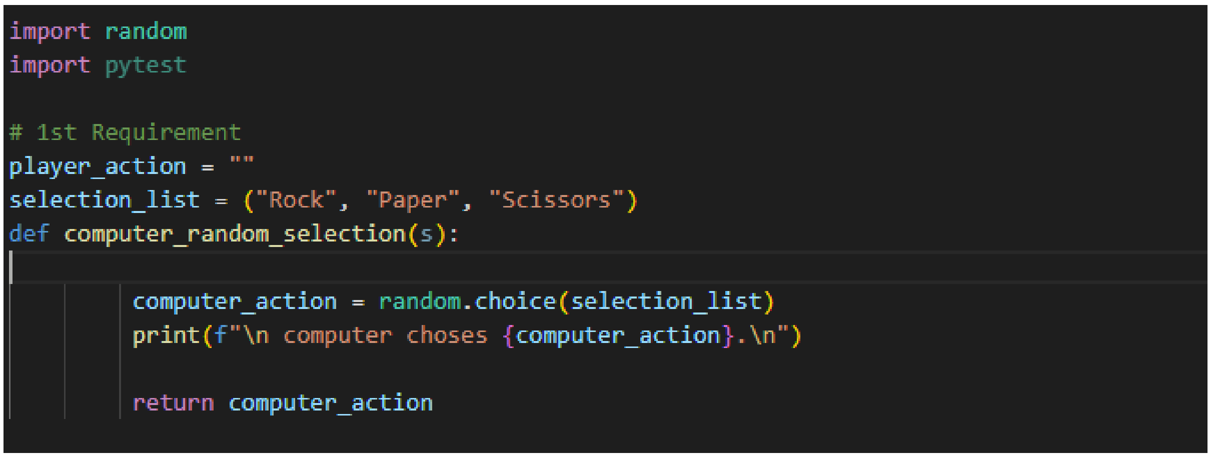
The computer randomly picks one of the options of scissor, paper, and rock		
Test case ID	**Description**	**Expected result**
TC1	Computer select Rock	Output is “Rock”
TC2	Computer select Scissors	Output is “Scissors”
TC3	Computer select Paper	Output is “Paper”



```python
def test_case1():
    ss = computer_random_selection("computer_action")
    print(f"test_case1 result: {ss}")
    assert ss in selection_list

def test_case2():
    ss = computer_random_selection("computer_action")
    print(f"test_case1 result: {ss}")
    assert ss in selection_list

def test_case3():
    ss = computer_random_selection("computer_action")
    print(f"test_case1 result: {ss}")
    assert ss in selection_list
```

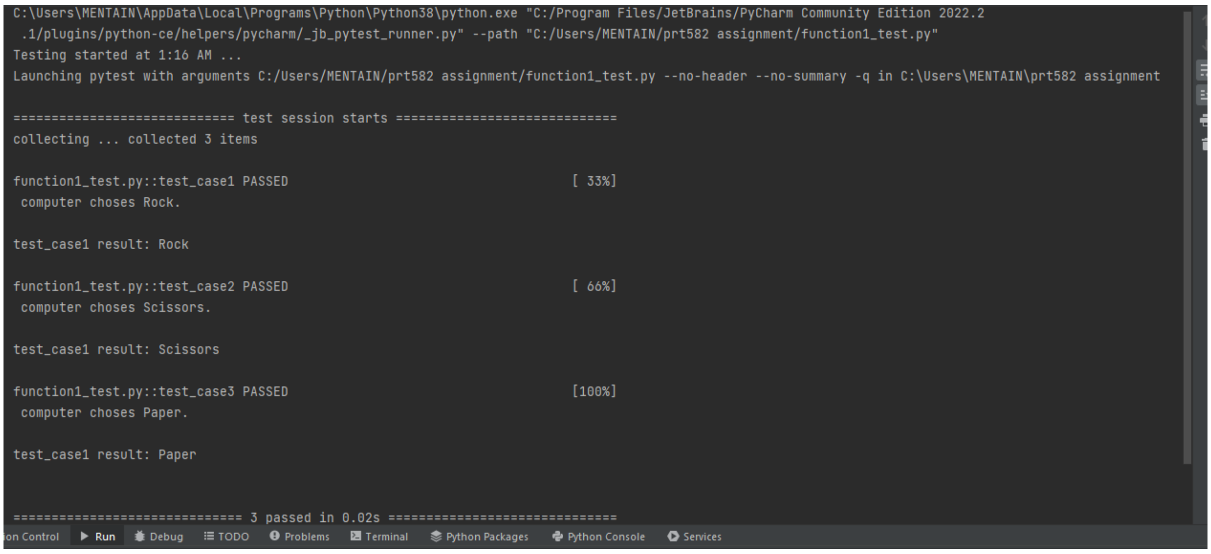
These Test cases will test whether “computer_action” will randomly select either Rock, Paper, or Scissors respectively.

In the next step, the function was built base on the test cases and the given 1st requirement:



```
import random
import pytest

# 1st Requirement
player_action = ""
selection_list = ("Rock", "Paper", "Scissors")
def computer_random_selection(s):

        computer_action = random.choice(selection_list)
        print(f"\n computer choses {computer_action}.\n")

        return computer_action
```

Specifically, I used the Random method to select the preferred string randomly for the computer from selection_list. I then created a while loop for the code to keep running if the condition is True.
The output is printed after the loop finishes.
I ran the tests after finishing building the function. Here are the test results of the test cases:



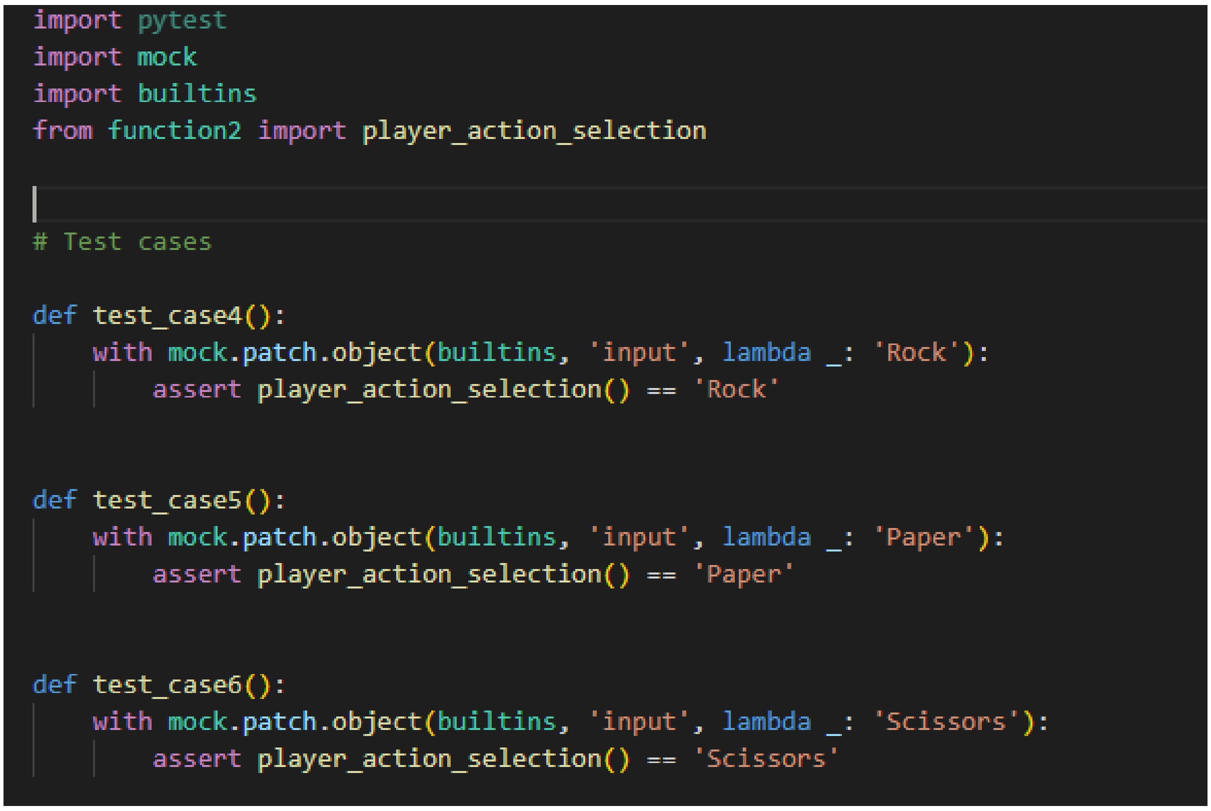
```
C:\Users\MENTAIN\AppData\Local\Programs\Python\Python38\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2
 .1/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --path "C:/Users/MENTAIN/prt582 assignment/function1_test.py"
Testing started at 1:16 AM ...
Launching pytest with arguments C:/Users/MENTAIN/prt582 assignment/function1_test.py --no-header --no-summary -q in C:\Users\MENTAIN\prt582 assignment

============================ test session starts =============================
collecting ... collected 3 items

function1_test.py::test_case1 PASSED                                         [ 33%]
 computer choses Rock.

test_case1 result: Rock

function1_test.py::test_case2 PASSED                                         [ 66%]
 computer choses Scissors.

test_case1 result: Scissors

function1_test.py::test_case3 PASSED                                         [100%]
 computer choses Paper.

test_case1 result: Paper


============================== 3 passed in 0.02s ==============================
```

Following the above result, all the test cases are passed so the function is built successfully.

2nd Requirement:

For the 2nd requirement, the player picks one from any of the options of scissor, paper, and rock. Therefore, I created the test cases for the requirement as shown below:

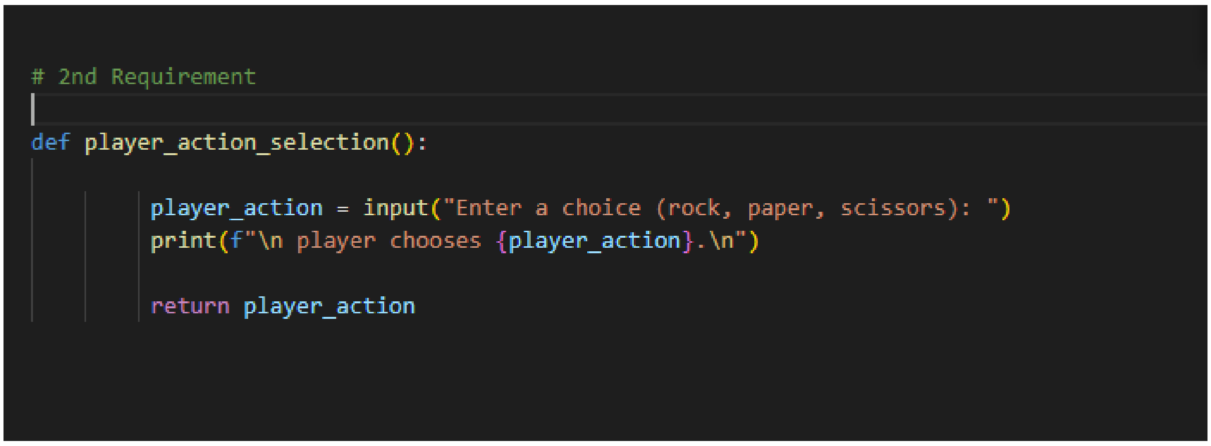
Player is then given the option to pick one of the options of scissor, paper, and rock.		
Test case ID	**Description**	**Expected result**
TC4	Player selects Rock	Output is “Rock”
TC5	Computer Select Paper	Output is “Paper”
TC6	Computer select Scissors	Output is “Scissors”



```
import pytest
import mock
import builtins
from function2 import player_action_selection


# Test cases

def test_case4():
    with mock.patch.object(builtins, 'input', lambda _: 'Rock'):
        assert player_action_selection() == 'Rock'


def test_case5():
    with mock.patch.object(builtins, 'input', lambda _: 'Paper'):
        assert player_action_selection() == 'Paper'


def test_case6():
    with mock.patch.object(builtins, 'input', lambda _: 'Scissors'):
        assert player_action_selection() == 'Scissors'
```

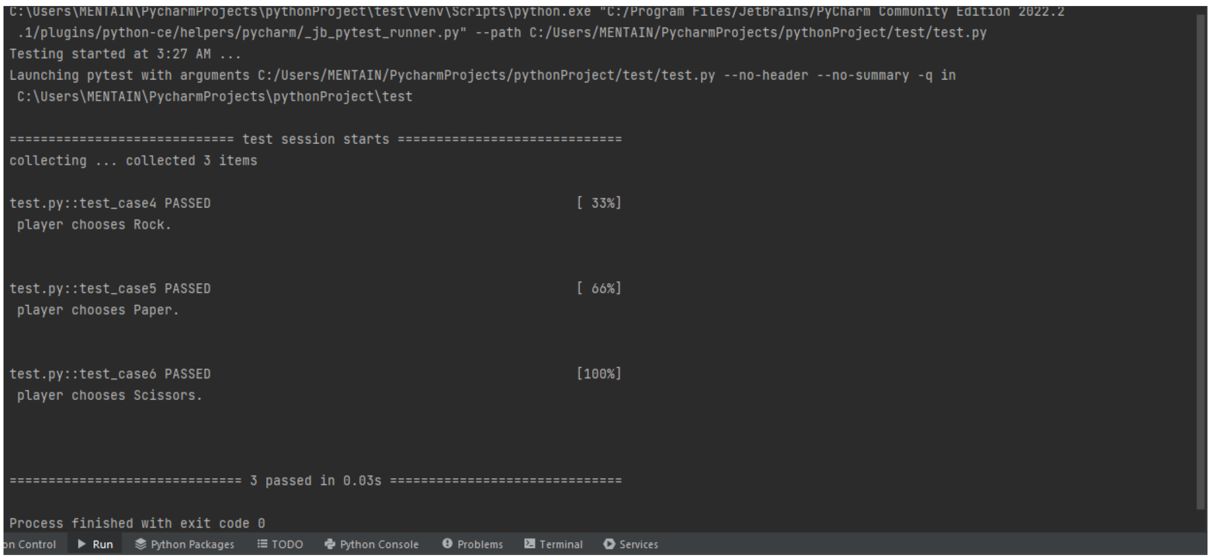
I created a function to accept user input.

I ran the tests after finishing building the function. Here are the test results of the test cases:



```
# 2nd Requirement

def player_action_selection():

            player_action = input("Enter a choice (rock, paper, scissors): ")
            print(f"\n player chooses {player_action}.\n")

            return player_action
```

The tests were run automatically, and the results are shown below:



```
C:\Users\MENTAIN\PycharmProjects\pythonProject\test\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2
 .1/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --path C:/Users/MENTAIN/PycharmProjects/pythonProject/test/test.py
Testing started at 3:27 AM ...
Launching pytest with arguments C:/Users/MENTAIN/PycharmProjects/pythonProject/test/test.py --no-header --no-summary -q in
 C:\Users\MENTAIN\PycharmProjects\pythonProject\test

============================ test session starts =============================
collecting ... collected 3 items

test.py::test_case4 PASSED                                               [ 33%]
 player chooses Rock.


test.py::test_case5 PASSED                                               [ 66%]
 player chooses Paper.


test.py::test_case6 PASSED                                               [100%]
 player chooses Scissors.



============================= 3 passed in 0.03s ==============================

Process finished with exit code 0
```

Following the above result, all the test cases are passed so the function is built successfully.

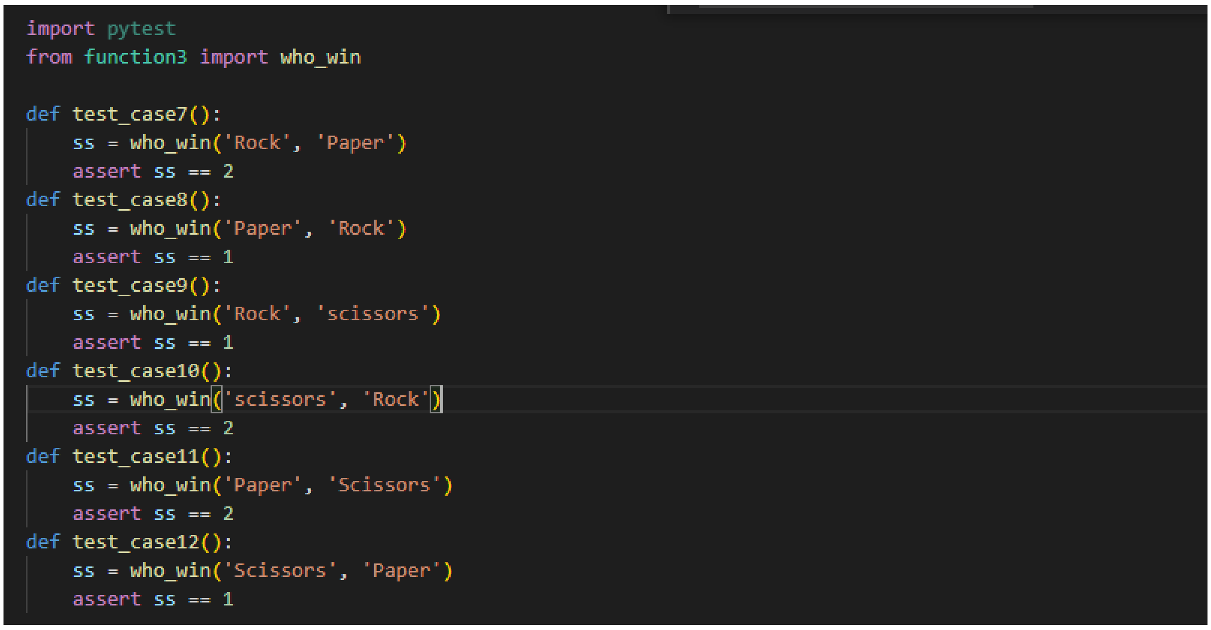
3rd Requirement:

The 3rd requirement, this requirement is used to determine who wins each round after a selection between the player and computer for all possible scenario or outcome.

The test cases are created to test the function:

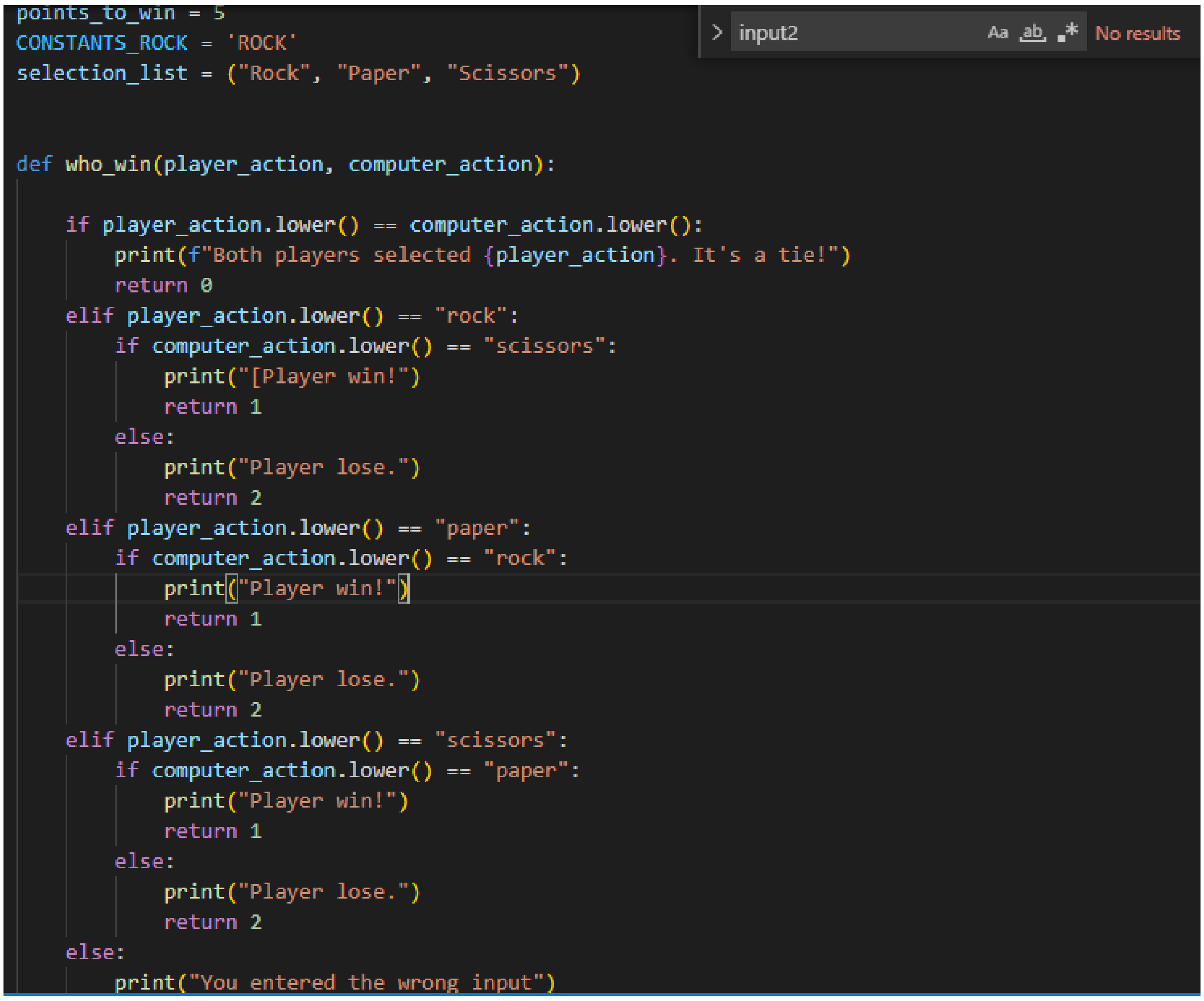
Determining the winner of each round for all possible outcomes		
Test case ID	**Description**	**Expected result**
TC7	player wins this round	Player win
TC8	computer wins this round	Computer win
TC9	player wins this round	Player win
TC10	computer wins this round	Computer win
TC11	player wins this round	Player win
TC12	computer Wins this round	Computer win

The test cases will confirm who wins each game round between the computer or player after the selection of rock, paper, or scissors.



```python
import pytest
from function3 import who_win

def test_case7():
    ss = who_win('Rock', 'Paper')
    assert ss == 2
def test_case8():
    ss = who_win('Paper', 'Rock')
    assert ss == 1
def test_case9():
    ss = who_win('Rock', 'scissors')
    assert ss == 1
def test_case10():
    ss = who_win('scissors', 'Rock')
    assert ss == 2
def test_case11():
    ss = who_win('Paper', 'Scissors')
    assert ss == 2
def test_case12():
    ss = who_win('Scissors', 'Paper')
    assert ss == 1
```

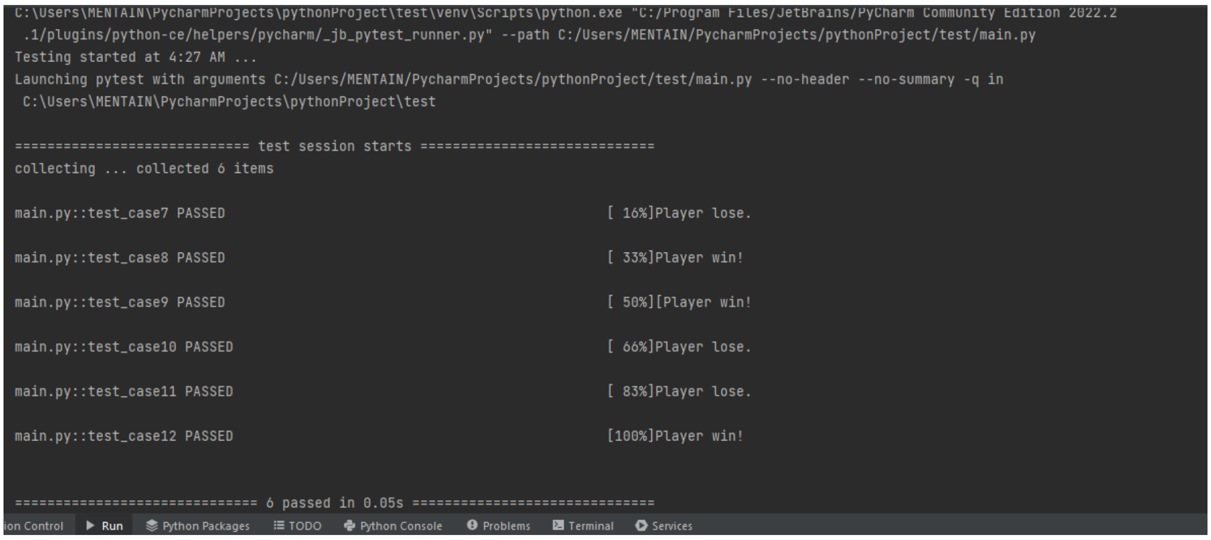
The function is built to work with the test cases and to follow the requirement:



```python
points_to_win = 5
CONSTANTS_ROCK = 'ROCK'
selection_list = ("Rock", "Paper", "Scissors")


def who_win(player_action, computer_action):

    if player_action.lower() == computer_action.lower():
        print(f"Both players selected {player_action}. It's a tie!")
        return 0
    elif player_action.lower() == "rock":
        if computer_action.lower() == "scissors":
            print("[Player win!")
            return 1
        else:
            print("Player lose.")
            return 2
    elif player_action.lower() == "paper":
        if computer_action.lower() == "rock":
            print("Player win!")
            return 1
        else:
            print("Player lose.")
            return 2
    elif player_action.lower() == "scissors":
        if computer_action.lower() == "paper":
            print("Player win!")
            return 1
        else:
            print("Player lose.")
            return 2
    else:
        print("You entered the wrong input")
```

The variable player_win and cpu_win where assigned an initial value of zero, which increases by value of 1 whenever the player of computer wins the round.

The tests were run automatically, and the results are shown below:



```
C:\Users\MENTAIN\PycharmProjects\pythonProject\test\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2
 .1/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --path C:/Users/MENTAIN/PycharmProjects/pythonProject/test/main.py
Testing started at 4:27 AM ...
Launching pytest with arguments C:/Users/MENTAIN/PycharmProjects/pythonProject/test/main.py --no-header --no-summary -q in
 C:\Users\MENTAIN\PycharmProjects\pythonProject\test

============================= test session starts ==============================
collecting ... collected 6 items

main.py::test_case7 PASSED                                               [ 16%]Player lose.

main.py::test_case8 PASSED                                               [ 33%]Player win!

main.py::test_case9 PASSED                                               [ 50%][Player win!

main.py::test_case10 PASSED                                              [ 66%]Player lose.

main.py::test_case11 PASSED                                              [ 83%]Player lose.

main.py::test_case12 PASSED                                              [100%]Player win!


============================== 6 passed in 0.05s ===============================
```

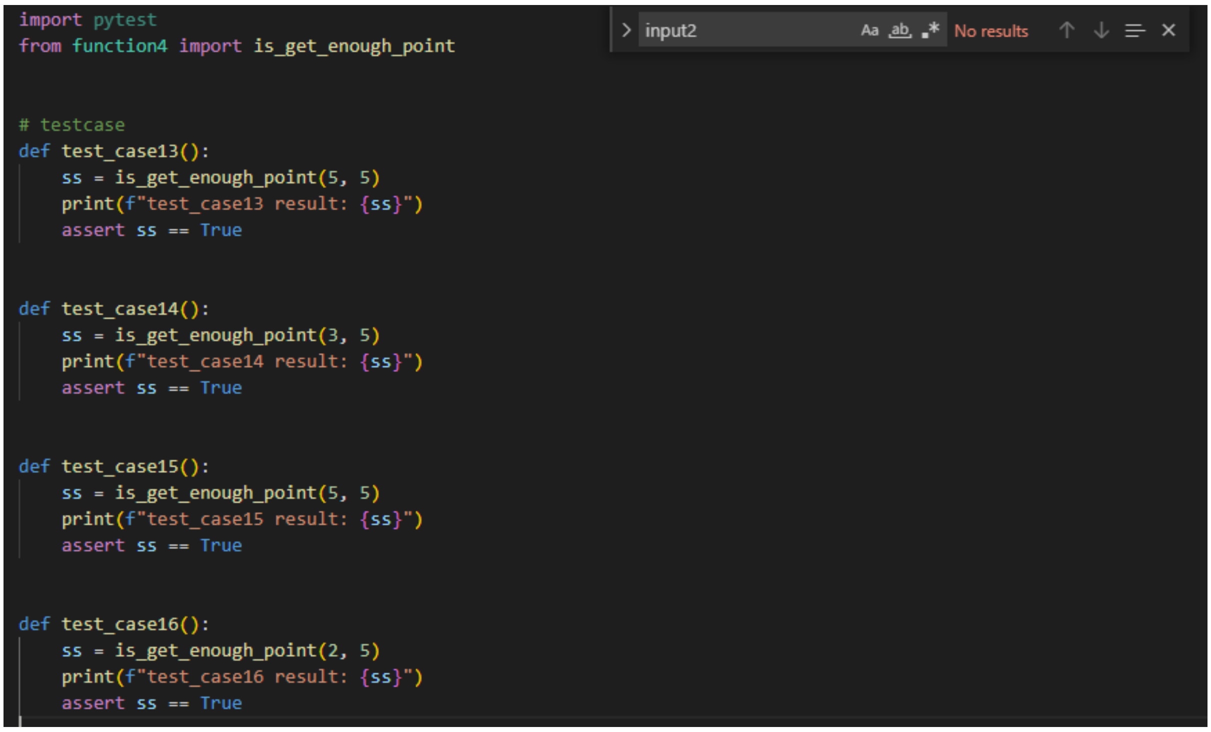
Following the above result, all the test cases are passed so the function are built successfully.

4th Requirement:

The 4th requirement requires an if statement to check the numbers of win between the player and the computer, the first to get to five wins the game.

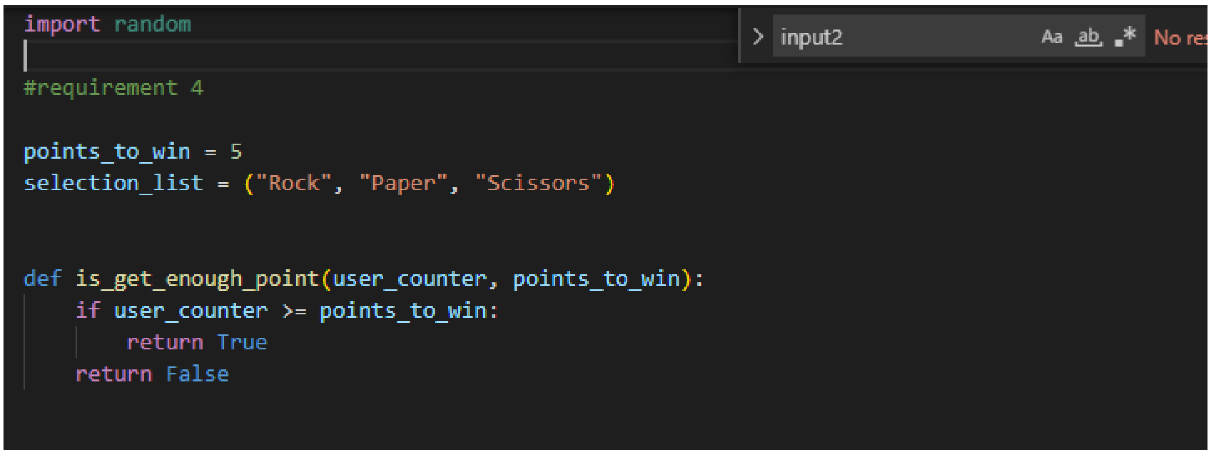
Following the TDD process, firstly, the test cases are created for this requirement:

The first to get five points wins the game.		
Test case ID	**Description**	**Expected result**
TC 13	Player gets to five wins	True
TC 14	Player gets to five wins	False
TC15	computer gets to five wins	True
TC16	computer gets to five wins	False



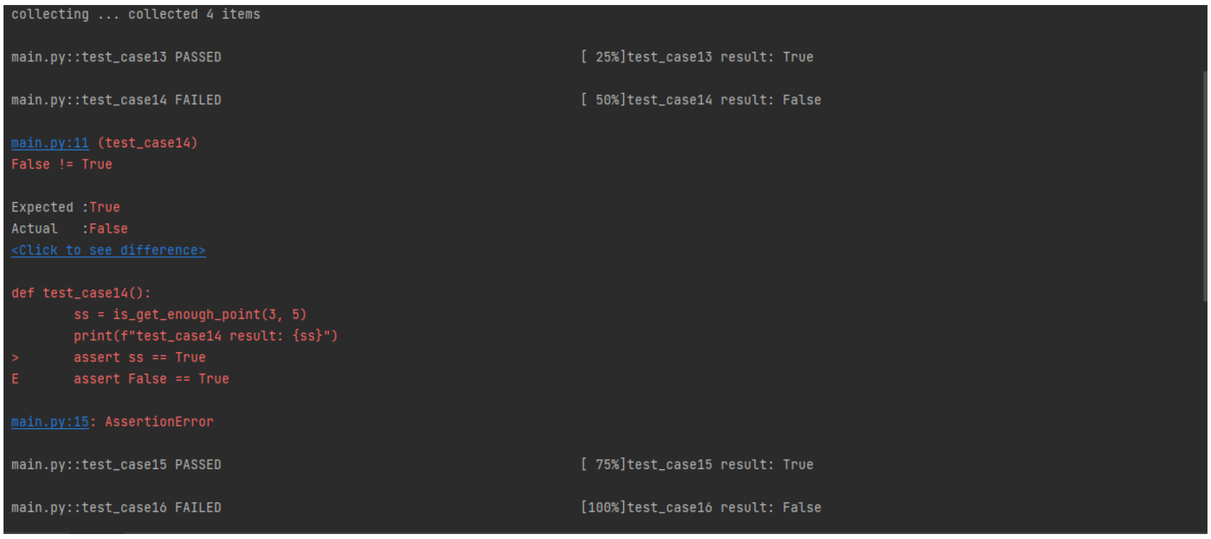
```python
import pytest
from function4 import is_get_enough_point


# testcase
def test_case13():
    ss = is_get_enough_point(5, 5)
    print(f"test_case13 result: {ss}")
    assert ss == True


def test_case14():
    ss = is_get_enough_point(3, 5)
    print(f"test_case14 result: {ss}")
    assert ss == True


def test_case15():
    ss = is_get_enough_point(5, 5)
    print(f"test_case15 result: {ss}")
    assert ss == True


def test_case16():
    ss = is_get_enough_point(2, 5)
    print(f"test_case16 result: {ss}")
    assert ss == True
```

The function is built to work with the test cases and to follow the requirement:



```python
import random

#requirement 4

points_to_win = 5
selection_list = ("Rock", "Paper", "Scissors")


def is_get_enough_point(user_counter, points_to_win):
    if user_counter >= points_to_win:
        return True
    return False
```

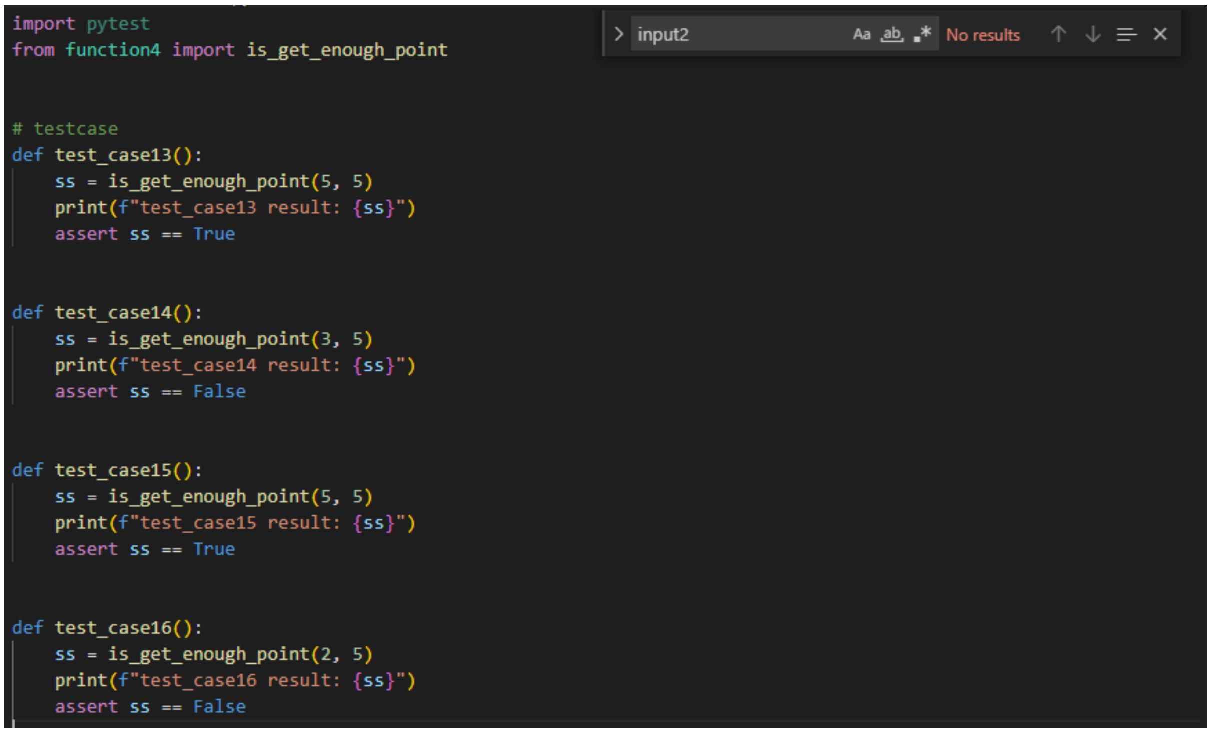
The tests were run automatically, and the results are shown below:



```
collecting ... collected 4 items

main.py::test_case13 PASSED                          [ 25%]test_case13 result: True

main.py::test_case14 FAILED                          [ 50%]test_case14 result: False

main.py:11 (test_case14)
False != True

Expected :True
Actual   :False
<Click to see difference>

def test_case14():
        ss = is_get_enough_point(3, 5)
        print(f"test_case14 result: {ss}")
>       assert ss == True
E       assert False == True

main.py:15: AssertionError

main.py::test_case15 PASSED                          [ 75%]test_case15 result: True

main.py::test_case16 FAILED                          [100%]test_case16 result: False
```

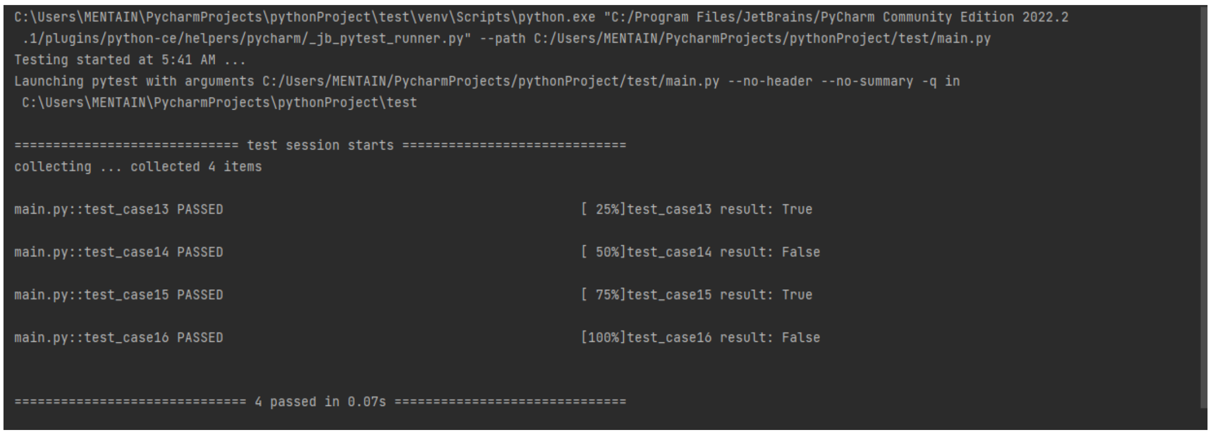
Following the above result, two of the test cases are passed, while the remaining two failed because it didn’t meet the stated Boolean conditions. Had to correct the Boolean condition to meet the requirement for a win, so the function is fixed successfully.

The corrected Testcase is shown below:



```python
import pytest
from function4 import is_get_enough_point


# testcase
def test_case13():
    ss = is_get_enough_point(5, 5)
    print(f"test_case13 result: {ss}")
    assert ss == True


def test_case14():
    ss = is_get_enough_point(3, 5)
    print(f"test_case14 result: {ss}")
    assert ss == False


def test_case15():
    ss = is_get_enough_point(5, 5)
    print(f"test_case15 result: {ss}")
    assert ss == True


def test_case16():
    ss = is_get_enough_point(2, 5)
    print(f"test_case16 result: {ss}")
    assert ss == False
```

Then the tests are run again:



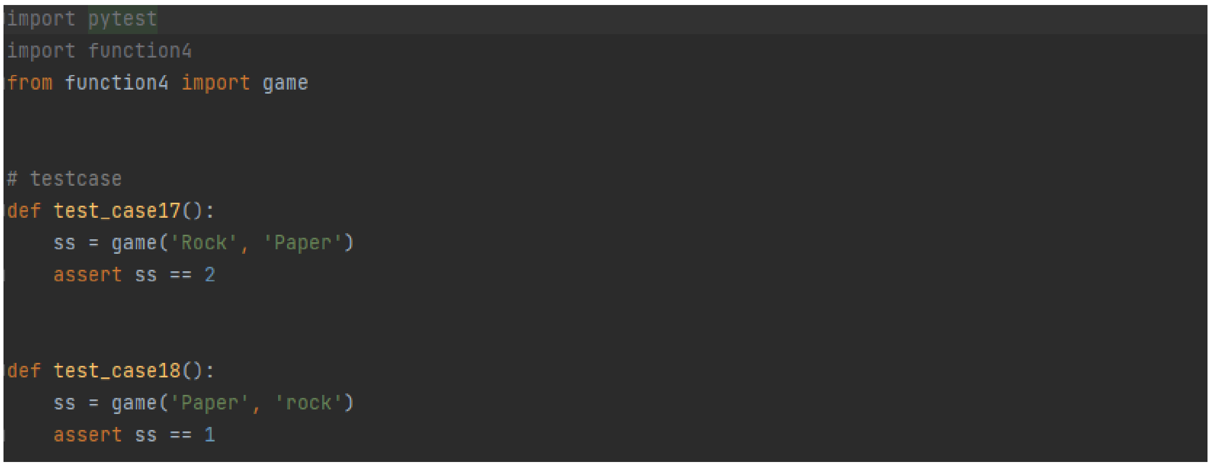
```
C:\Users\MENTAIN\PycharmProjects\pythonProject\test\venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2022.2
 .1/plugins/python-ce/helpers/pycharm/_jb_pytest_runner.py" --path C:/Users/MENTAIN/PycharmProjects/pythonProject/test/main.py
Testing started at 5:41 AM ...
Launching pytest with arguments C:/Users/MENTAIN/PycharmProjects/pythonProject/test/main.py --no-header --no-summary -q in
 C:\Users\MENTAIN\PycharmProjects\pythonProject\test

============================= test session starts ==============================
collecting ... collected 4 items

main.py::test_case13 PASSED                                              [ 25%]test_case13 result: True

main.py::test_case14 PASSED                                              [ 50%]test_case14 result: False

main.py::test_case15 PASSED                                              [ 75%]test_case15 result: True

main.py::test_case16 PASSED                                              [100%]test_case16 result: False


============================== 4 passed in 0.07s ===============================
```

THE PROGRAM:

After finishing all the functions, I combined them together added some appropriate code to make them work together.

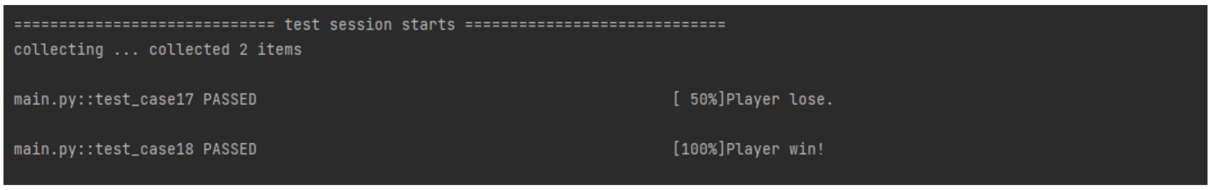
```
import random

selection_list = ["Rock", "Paper", "Scissors"]


def game(player_action, computer_action):
    player_win = 0
    cpu_win = 0
    while True:
        if player_action.lower() == computer_action.lower():
            print(f"Both players selected {player_action}. It's a tie!")
            return 0
        elif player_action.lower() == "rock":
            if computer_action.lower() == "scissors":
                player_win = player_win + 1
                print("Player win!")
                return 1
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
                return 2
        elif player_action.lower() == "paper":
            if computer_action.lower() == "rock":
                player_win = player_win + 1
                print("Player win!")
                return 1
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
                return 2
        elif player_action.lower() == "scissors":
            if computer_action.lower() == "paper":
                player_win = player_win + 1
                print("Player win!")
                return 1
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
                return 2
        else:
            print("You entered the wrong input")
        if player_win >= 5 or cpu_win >= 5:
            if cpu_win < player_win:
                print("Player won the game")
            else:
                print("Computer won the game")
            print("Final Scores:")
            print(f"Player:{player_win}")
            print(f"Comp:{cpu_win}")
            play_again = input("Play again? (yes/no): ")
            if play_again.lower() != "yes":
                break


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

The test cases were created to test if the whole program works properly:



```
import pytest
import function4
from function4 import game


# testcase
def test_case17():
    ss = game('Rock', 'Paper')
    assert ss == 2


def test_case18():
    ss = game('Paper', 'rock')
    assert ss == 1
```

The result is shown below:



```
============================ test session starts ============================
collecting ... collected 2 items

main.py::test_case17 PASSED                                            [ 50%]Player lose.

main.py::test_case18 PASSED                                            [100%]Player win!
```

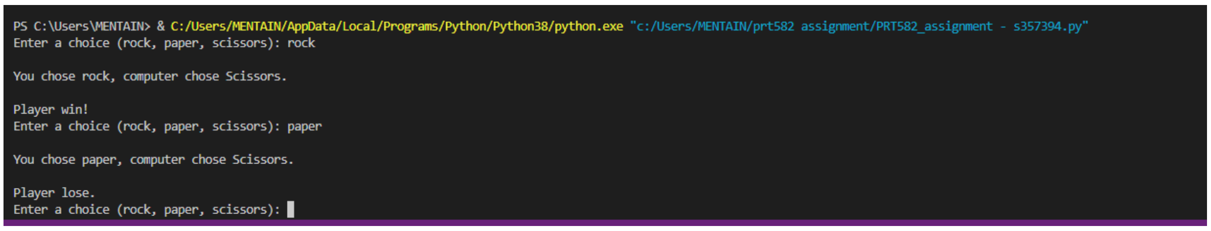
Following the result above, the program was built successfully. I then refactored the code to add the users' input.

The full code of the program is shown below:

```
import random
def main():
    player_win=0
    cpu_win=0
    while True:
        player_action = input("Enter a choice (rock, paper, scissors): ")
        selection_list = ["Rock", "Paper", "Scissors"]
        computer_action = random.choice(selection_list)
        print(f"\nYou chose {player_action}, computer chose {computer_action}.\n")

        if player_action.lower() == computer_action.lower():
            print(f"Both players selected {player_action}. It's a tie!")
        elif player_action.lower() == "rock":
            if computer_action.lower() == "scissors":
                player_win=player_win + 1
                print("Player win!")
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
        elif player_action.lower() == "paper":
            if computer_action.lower() == "rock":
                player_win=player_win + 1
                print("Player win!")
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
        elif player_action.lower() == "scissors":
            if computer_action.lower() == "paper":
                player_win=player_win + 1
                print("Player win!")
            else:
                cpu_win = cpu_win + 1
                print("Player lose.")
        else:
            print("You entered the wrong input")
        if player_win >= 5 or cpu_win >=5:
            if cpu_win < player_win:
                    print("Player won the game")
            else:
                    print("Computer won the game")
            print("Final Scores:")
            print(f"Player:{player_win}")
            print(f"Comp:{cpu_win}")
            play_again = input("Play again? (yes/no): ")
            if play_again.lower() != "yes":
             break


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

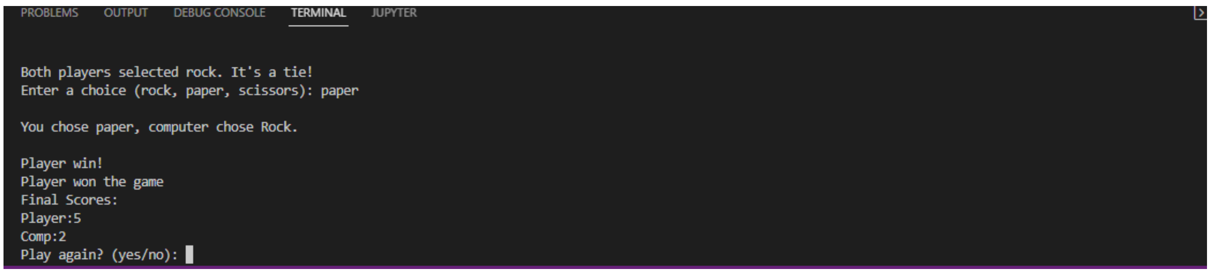
Finally, I ran the code to ensure it would work:

Scenario 1: Checking User and Computer input



```
PS C:\Users\MENTAIN> & C:/Users/MENTAIN/AppData/Local/Programs/Python/Python38/python.exe "c:/Users/MENTAIN/prt582 assignment/PRT582_assignment - s357394.py"
Enter a choice (rock, paper, scissors): rock

You chose rock, computer chose Scissors.

Player win!
Enter a choice (rock, paper, scissors): paper

You chose paper, computer chose Scissors.

Player lose.
Enter a choice (rock, paper, scissors):
```

Scenario 2: All the conditions are passed, and the winner is shown



```
Both players selected rock. It's a tie!
Enter a choice (rock, paper, scissors): paper

You chose paper, computer chose Rock.

Player win!
Player won the game
Final Scores:
Player:5
Comp:2
Play again? (yes/no):
```

CONCLUSION:

The Test-Driven Development process is being used by many developers nowadays. It helps them to build the program properly. Furthermore, with the support of many automated testing tools such as Pytest, the developers can save time testing and debugging the code. I will be practicing more of these techniques and tools to improve my coding skill.

Github link: https://github.com/mentain/PRT582_Assignment/