Unit: S222 PRT582 SOFTWARE ENGINEERING: PROCESS AND TOOLS

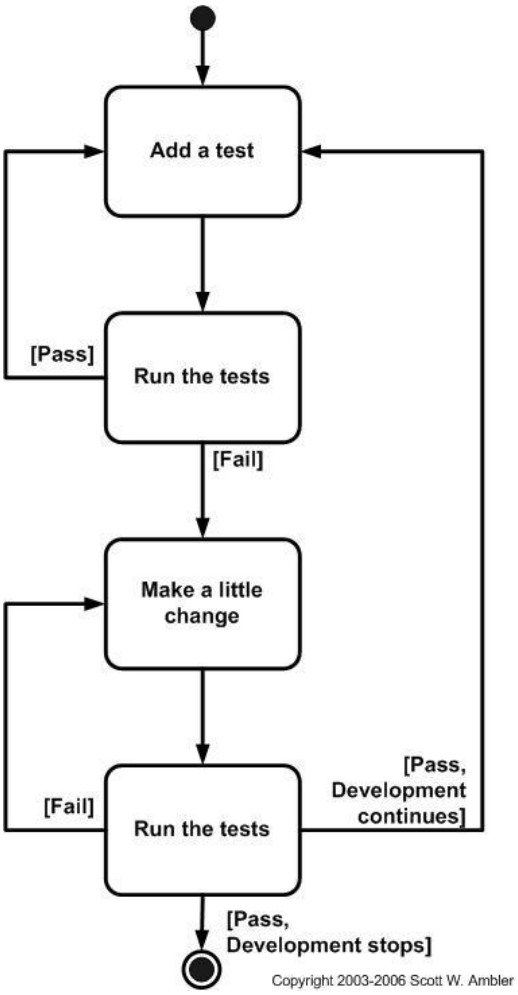
Software Unit Testing Report:

Scissor Paper Rock game using TDD in Python Programming Language

Submitted by :
Name: Md Hasan Al Mahmud
Student ID: 360863

Introduction:

Python is a high-level, general-purpose programming language. It can be used for developing software. The contemporary environment of software development is diverse enough to have different approaches to coding. Among these, modern and agile approach is test-driven development (TDD). Test Driven Development (TDD) is an approach to software development combining test-first development and refactoring. In other words, you write a test before you write just enough coding to fulfil that test. The steps of test-first development are shown below:



The objective of this task is to write a Scissor Paper Rock game using Test Driven Development in Python. The game rules are mentioned below-

The player has to choose one of the options between scissor, paper and rock. This is then compared against the computer's random selection and determine who the winner is.
Winning rules are as follows:

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

The basic game requirements are:

i. The computer randomly picks one of the options of scissor, paper and rock.
ii. Player is then given the option to pick/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. The total number of rounds played in total will also be displayed.
v. Once the winner is determined, the player is asked to quit or restart the game
vi. Player can also quit the game at any time.

Process:

I have written a program for the game mentioned above using python and TDD. I have created two files. The first is the test file and the second is the file that contains the actual functional program.

Step #1:

To start, I have written a test file with the filename test_ rockPaperScissor.py. I imported unit testing framework to test the test cases.

As the first requirement is 'The computer randomly picks one of the options of scissor, paper and rock', I have created the first test case and run it. The code is-

```
1  # Test for Scissor Paper Rock game using Test Driven Development
2
3  import unittest
4  from rockPaperScissor import RockPaperScissor
5
6
7  class GameTestCase(unittest.TestCase):
8      game = RockPaperScissor()
9
10 def test_computerChoice(self):
11         self.assertIn(self.game.computerChoice(), ["r", "p", "s"], msg="Invalid option chosen by computer.") # Testing if computer choose these o
12
13
14 if __name__ == '__main__':
15     unittest.main()
```

But the error occurred in output showing module not found which is shown below-

```
PS C:\Users\Mify's Laptop> & "C:/Users/Mify's Laptop/AppData/Local/Programs/Python/Python38/python.exe" e:/assessment1/test_rockPaperScissor.py
Traceback (most recent call last):
  File "e:/assessment1/test_rockPaperScissor.py", line 5, in <module>
    from rockPaperScissor import RockPaperScissor
ModuleNotFoundError: No module named 'rockPaperScissor'
PS C:\Users\Mify's Laptop>
```

Actually I have just started and have not created any actual program file for running functions. Then I created a functional code file named rockPaperScissor.py. I have created a class and a method for running the test case which is shown below-

```python

    import random

    class RockPaperScissor():

        gamer = ["computer", "player", "tie"]
        choiceList = ["r", "p", "s"]  # r for rock, p for paper, s for sissor

        def computerChoice(self):    # random choice by computer
            return random.choice(self.choiceList)
```

Then I run the unit testing code and successfully it passed which is shown below-

```
PS C:\Users\Mify's Laptop> & "C:/Users/Mify's Laptop/AppData/Local/Programs/Python/Python38/python.exe" e:/assessment1/test_rockPaperScissor.py
.
----------------------------------------------------------------------
Ran 1 test in 0.000s

OK
```

Step #2:

As per 2nd requirement- 'Player is then given the option to pick/type one of the options of scissor, paper and rock', I have created another method in functional program file and subsequent program in unit testing program file. Earlier, the actual program file had error and this is identified when I run the test file. As per error message I made the correction and run the test file again and this time the code run successfully which are shown below-

Functional code:

```python

    import random


    class RockPaperScissor():

        gamer = ["computer", "player", "tie"]
        choiceList = ["r", "p", "s"]  # r for rock, p for paper, s for sissor

        def computerChoice(self):   # random choice by computer
            return random.choice(self.choiceList)

        @staticmethod
        def PlayerChoice():  # choice by user/player/human
            playerInput = input("\nPlease enter r for rock, or p for paper, or s for scissors or q for quit: " )
            return playerInput
```

Testing code:

```python
# Test for Scissor Paper Rock game using Test Driven Development

import unittest
from unittest import mock
from rockPaperScissor import RockPaperScissor


class GameTestCase(unittest.TestCase):
    game = RockPaperScissor()

    def test_computerChoice(self):
        self.assertIn(self.game.computerChoice(), ["r", "p", "s"], msg="Invalid option chosen by computer.") # Testing if computer choose these c

    def test_playerChoice(self):
        with mock.patch('builtins.input', return_value="p"):
            assert self.game.PlayerChoice() in ["r", "p", "s", "q"]  # testing player's input
```

Output:

```
PS C:\Users\Mify's Laptop> & "C:/Users/Mify's Laptop/AppData/Local/Programs/Python/Python38/python.exe" e:/assessment1/test_rockPaperScissor.py
..
----------------------------------------------------------------------
Ran 2 tests in 0.001s

OK
PS C:\Users\Mify's Laptop>
```

Step #3:

I followed the same process for all of the requirements. Third requirement is ‘One point is given to the winner of that round ’. I created another method and test it using unit testing tool until it run successfully. These are shown below-

Functional code:

```python

import random

class RockPaperScissor():

    gamer = ["computer", "player", "tie"]
    choiceList = ["r", "p", "s"]  # r for rock, p for paper, s for sissor


    def computerChoice(self):   # random choice by computer
        return random.choice(self.choiceList)

    @staticmethod
    def PlayerChoice():  # choice by user/player/human
        playerInput = input("\nPlease enter r for rock, or p for paper, or s for scissors or q for quit: " )
        return playerInput

    @staticmethod
    def pointAdd(currentPoint):
        currentPoint += 1  # addition of point for winning a round
        return currentPoint
```

Test code:

```python
# Test for Scissor Paper Rock game using Test Driven Development

import unittest
from unittest import mock
from rockPaperScissor import RockPaperScissor


class GameTestCase(unittest.TestCase):
    game = RockPaperScissor()

    def test_computerChoice(self):
        self.assertIn(self.game.computerChoice(), ["r", "p", "s"], msg="Invalid option chosen by computer.") # Testing if computer choose these o

    def test_playerChoice(self):
        with mock.patch('builtins.input', return_value="p"):
            assert self.game.PlayerChoice() in ["r", "p", "s", "q"]  # testing player's input


    def test_pointAdd(self):
        self.assertEqual(3, self.game.pointAdd(2), msg="Incorrect point addition.")  # testing point adition to winner of a round


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

Output:

```
PS C:\Users\Mify's Laptop> & "C:/Users/Mify's Laptop/AppData/Local/Programs/Python/Python38/python.exe" e:/assessment1/test_rockPaperScissor.py
...
----------------------------------------------------------------------
Ran 3 tests in 0.001s

OK
```

Step #4:

The fourth requirement is – the first to get five points wins the game. As per condition, winning rules are as follows:

- rock vs paper -> paper wins
- rock vs scissor -> rock wins
- paper vs scissor -> scissor wins.
- For same choice, result of the round will be ‘Tie’.

Accordingly code including two methods is prepared and run successfully. One method is for setting the winning point required to win the match and another is for determining the winner. These are shown below-

Functional code:

```
def winningPoint(self, currentPoint):
    pointAchieved = False
    if currentPoint == 5:
        pointAchieved = True  # if either player or computer achieve 5 points, wins the game
    return pointAchieved

def setWinner(self, playerChoice, computerChoice):
    winner = ''

    if playerChoice==computerChoice:
        winner=self.gamer[2]   # computer and user both have same choice, so nobody gets point at this round

    elif playerChoice == 'r':
        if computerChoice =='p':
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point
        else :
            winner=self.gamer[1]   # player wins this round as per given condition, so will get 1 point
    elif playerChoice =='p':
        if computerChoice =='s':
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point
        else:
            winner=self.gamer[1]   # computer wins this round as per given condition, so will get 1 point
    elif playerChoice =='s':
        if computerChoice =='p':
            winner=self.gamer[1]   # computer wins this round as per given condition, so will get 1 point
        else:
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point

    return winner
```

Test code:

```
def test_winningPoint(self):
    self.assertFalse(self.game.winningPoint(4))
    self.assertTrue(self.game.winningPoint(5))   # Testing if either computer or player gets 5 points to win the game


def test_setWinner(self):
    #Testing of compaison and tsting of determination of winner

    self.assertEqual("tie", self.game.setWinner("r", "r"), msg="Invalid winner")
    self.assertEqual("computer", self.game.setWinner("r", "p"), msg="Invalid winner")
    self.assertEqual("player", self.game.setWinner("r", "s"), msg="Invalid winner")
    self.assertEqual("player", self.game.setWinner("p", "r"), msg="Invalid winner")
    self.assertEqual("tie", self.game.setWinner("p", "p"), msg="Invalid winner")
    self.assertEqual("computer", self.game.setWinner("p", "s"), msg="Invalid winner")
    self.assertEqual("computer", self.game.setWinner("s", "r"), msg="Invalid winner")
    self.assertEqual("player", self.game.setWinner("s", "p"), msg="Invalid winner")
    self.assertEqual("tie", self.game.setWinner("s", "s"), msg="Invalid winner")
```

Output:

```
PS C:\Users\Mify's Laptop> & "C:/Users/Mify's Laptop/AppData/Local/Programs/Python/Python38/python.exe" e:/assessment1/test_rockPaperScissor.py
.....
----------------------------------------------------------------------
Ran 5 tests in 0.001s

OK
```

Step #5:

The other requirements are – 'once the winner is determined, the player is asked to quit or restart the game', 'Player can also quit the game at any time' and 'The total number of rounds played in total will also be displayed'. Accordingly codes are designed with a method (named 'start') for starting the game, keeping quit option, taking inputs and showing relevant messages and outputs. Another method named 'replay' is created for option to the player for asking to quit or replay after a game is finished. Then the functional code is tested by unit testing code as per previous process until test code is successfully run. Then the 'main function' is added for running the whole program as per standard practice. The final codes is shown below-

Unit testing code (test_ rockPaperScissor.py):

```
# Test for Scissor Paper Rock game using Test Driven Development

import unittest
from unittest import mock
from rockPaperScissor import RockPaperScissor


class GameTestCase(unittest.TestCase):
    game = RockPaperScissor()

    def test_computerChoice(self):
        self.assertIn(self.game.computerChoice(), ["r", "p", "s"], msg="Invalid option chosen by computer.") # Testing if computer choose these o

    def test_playerChoice(self):
        with mock.patch('builtins.input', return_value="p"):
            assert self.game.PlayerChoice() in ["r", "p", "s", "q", "rs"]  # testing player's input


    def test_pointAdd(self):
        self.assertEqual(3, self.game.pointAdd(2), msg="Incorrect point addition.")  # testing point adition to winner of a round

    def test_winningPoint(self):
        self.assertFalse(self.game.winningPoint(4))
        self.assertTrue(self.game.winningPoint(5))   # Testing if either computer or player gets 5 points to win the game
```

```
    def test_setWinner(self):
        #Testing of compaison and tsting of determination of winner

        self.assertEqual("tie", self.game.setWinner("r", "r"), msg="Invalid winner")
        self.assertEqual("computer", self.game.setWinner("r", "p"), msg="Invalid winner")
        self.assertEqual("player", self.game.setWinner("r", "s"), msg="Invalid winner")
        self.assertEqual("player", self.game.setWinner("p", "r"), msg="Invalid winner")
        self.assertEqual("tie", self.game.setWinner("p", "p"), msg="Invalid winner")
        self.assertEqual("computer", self.game.setWinner("p", "s"), msg="Invalid winner")
        self.assertEqual("computer", self.game.setWinner("s", "r"), msg="Invalid winner")
        self.assertEqual("player", self.game.setWinner("s", "p"), msg="Invalid winner")
        self.assertEqual("tie", self.game.setWinner("s", "s"), msg="Invalid winner")



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

Actual functional code (rockPaperScissor.py):

```python
import random

class RockPaperScissor():

    gamer = ["computer", "player", "tie"]
    choiceList = ["r", "p", "s"]  # r for rock, p for paper, s for sissor

    def computerChoice(self):   # random choice by computer
        return random.choice(self.choiceList)

    @staticmethod
    def PlayerChoice():  # choice by user/player/human
        playerInput = input("\nPlease enter r for rock, or p for paper, or s for scissors or q for quit: " )
        return playerInput


    @staticmethod
    def pointAdd(currentPoint):
        currentPoint += 1  # addition of point for winning a round
        return currentPoint

    def winningPoint(self, currentPoint):
        pointAchieved = False
        if currentPoint == 5:
            pointAchieved = True  # if either player or computer achieve 5 points, wins the game
        return pointAchieved
```

```python
def setWinner(self, playerChoice, computerChoice):
    winner = ''

    if playerChoice==computerChoice:
        winner=self.gamer[2]   # computer and user both have same choice, so nobody gets point at this round

    elif playerChoice == 'r':
        if computerChoice =='p':
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point
        else :
            winner=self.gamer[1]   # player wins this round as per given condition, so will get 1 point
    elif playerChoice =='p':
        if computerChoice =='s':
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point
        else:
            winner=self.gamer[1]   # computer wins this round as per given condition, so will get 1 point
    elif playerChoice =='s':
        if computerChoice =='p':
            winner=self.gamer[1]   # computer wins this round as per given condition, so will get 1 point
        else:
            winner=self.gamer[0]   # computer wins this round as per given condition, so will get 1 point

    return winner
```

```
def start(self):

    playerPoint = computerPoint =roundNumber = 0

    while playerPoint != 5 or computerPoint != 5:

        playerChoice = self.PlayerChoice()  # input player choice

        if playerChoice in self.choiceList:  # checking if choice is valid i.e, rock paper or scissor

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

            computerChoice = self.computerChoice()
            print("Computer chose:", computerChoice)

            winner = self.setWinner(playerChoice, computerChoice)

            if winner == 'player':
                playerPoint = self.pointAdd(playerPoint)  # adding score to winner
                print("Congratz! You have won this round !")
                print("Your point is:", playerPoint)
                print("Computer's point is:", computerPoint)

            elif winner == 'computer':
                computerPoint = self.pointAdd(computerPoint)
                print("Computer have won this round")
                print("Your point is:", playerPoint)
                print("Computer's point is:", computerPoint)

            else:
                print("Tie!")  # tie
```

```
            if self.winningPoint(playerPoint) or self.winningPoint(computerPoint): # if anyone achieved winning point

                if self.winningPoint(playerPoint):
                    print("\n\nCongratz! you won the match !!")
                else:
                    print("\n\nComputer won the Game ! ")

                print("\nNumber of total rounds played: {}\n".format(roundNumber))

                break


        elif playerChoice == 'q':  # player may quit the game anytime
            break
        else:  # if invalid user input
            print("Invalid input! Please type r, p, s or q as your input.")

    def replay(self):   # player may quit or restart the game after winner is determined

        quitGame = False

        while not quitGame:  # play again until user quit the game
            self.start()

            gameContinuation = input("\nTo exit the game enter q or to restart enter rs: ").lower().strip()
```

```
            if gameContinuation == "q":
                quitGame = True
            elif gameContinuation == "rs":
                quitGame = False
            else:
                print("Invalid input! Please enter q or rs")


def main():

    game = RockPaperScissor()
    game.replay()


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

Then I checked the output/results of the actual python program code for all of the options and checked if all of the requirements are fulfilled. Sample outputs after running the code are shown below-

Output:

```
Please enter r for rock, or p for paper, or s for scissors or q for quit: s
Round: 7
Computer chose: p
Congratz! You have won this round !
Your point is: 4
Computer's point is: 1

Please enter r for rock, or p for paper, or s for scissors or q for quit: p
Round: 8
Computer chose: r
Congratz! You have won this round !
Your point is: 5
Computer's point is: 1


Congratz! you won the match !!

Number of total rounds played: 8


To exit the game enter q or to restart enter rs: q
PS C:\Users\Mify's Laptop>
```

After running the code several times like above sample I found that there is no errors and the code gives output as per all of the requirements. So, I can say that I have successfully programmed a code using TDD (test-driven development) in python programming language.

Conclusion:

From the experience achieved by designing of this game I found that TDD is a smart and agile process to program codes. It is following a systematic organized way and solving a problem part by part. Testing and debugging is easier for this. So it is time effective, more accurate and smart way for programming. Also, one person can easily understand the half cooked code designed by other and start working for rest. That's why in case of group work this process can be very effective and it is actually the agile process. So for better coding experience and better quality software, TDD is a very good technique which may be recommended to follow.

I have uploaded this report and program code in GitHub. Link is given below-
https://github.com/ham2k7/PRT582-assignment1.git