Introduction:

Rock, paper, scissor is an indoor game, in where two players can play the game. There have three outcomes on this game- win loss and tie. Rock can defeats scissor, scissor can defeats paper and paper can defeats rock.

In this assignment game, one input can take by users and another input take by computer where computer can takes input randomly. There have three options in code. Rock will be defined by one (1), paper will be defined by two (2) and finally scissor will be defined by three (3). If user chooses rock and computer also choose rock randomly, than the game will be tie. Computer can also defeats user by choosing the proper option. If user chooses rock and computer chooses paper randomly, than output will come user loss and computer win. Again, if user chooses paper and computer chooses scissor randomly, than output will come user loss and computer win. If user chooses scissor and computer chooses rock randomly, than output will come user win and computer loss. Basically this program also follows the basic rules of rock, paper, and scissor games. This program also counts the number of games that user and computer played. When the user win five consecutive game, than output will come the total number of games that needs to become win five times and also shows the total number of win by user. The user can quit the game by using 'Q' at any time and restart the game by using 'R'.

The player can choose rock, paper and scissor and the computer have to choose one randomly, are the main requirements of this assignment. The user can quit the game by using 'Q' at any time and restart the game by using 'R'. For TDD (Test Driven Development), need to make a test case file also. Than run the tests and check in case the test pass or fails. We ought to compose a few codes to create the test pass in case the test fails. Finally, we have to be compelled to run all the tests once more and make beyond any doubt it will work superbly.

Process:

TDD (Test Driven Development) follows some steps in their cycle, the steps are-

1. Add a test case.
2. Run all the test and check.
3. Write some code to make the test pass.
4. Run the test again.
5. Refactor the code.

Step 1:

Add a test case:

At the beginning we start with the "Rock, Rock portion of the game. We know that the outcome will draw as both user and computer choose Rock, so add a new Test Case and a Function,

```
import unittest

from game import game


class MyTestCase(unittest.TestCase):


    def test_game1(self):

        self.assertEqual('Draw', self.games.rock_paper_scissor('1', '1'))
```

Step 2:

Run all the test and check:

The test will not compile, and it is saying **AttributeError: 'MyTestCase' object has no attribute 'games',** the test will count as failed.

TDD step 3

Write some code to make the test pass

```
import unittest

from game import game


class MyTestCase(unittest.TestCase):

    games = game()


    def test_game1(self):

        self.assertEqual('Draw', self.games.rock_paper_scissor('1', '1'))
```

Still out test not pass although we import the game object, we have not write anything on the game class we only write the game function, so we need to write some code to pass the test.

```
import random

choices = ["1", "2", "3"]


class game(object):

    def rock_paper_scissor(self, choice, computer_choice):

        if choice == computer_choice:

            return 'Draw'
```

Step 4:

Run all the tests again

```
Ran 1 test in 0.000s

OK
```

The test passed.

Step 1.0:

Add a test case:

Now we will test the relationship between the rock and Scissors, In this case the user choice is rock and the computer choice is scissors, so the user will win in that case

```
def test_game2(self):

    self.assertEqual('True', self.games.rock_paper_scissor('1', '3'))
```

Step 2.0

Run all the test and check:

The test will not compile, as we have not write in our code

TDD step 3.0

Write some code to make the test pass

```
import random

choices = ["1", "2", "3"]


class game(object):

    def rock_paper_scissor(self, choice, computer_choice):

        if choice == computer_choice:

            return 'Draw'

        elif choice == "1":

            if computer_choice == "3":

                return 'True
```

Step 4.0

Run all the tests again

```
Ran 1 test in 0.000s

OK
```

The test passed.

Step 1.1:

Add a test case:

Now we will test the relationship between the Scissors and Rock, In this case the user choice is Scissors and the computer choice is Rock, so the computer wins in that case.

```
def test_game3(self):

        self.assertEqual('False', self.games.rock_paper_scissor('3', '1'))
```

Step 2.1

Run all the test and check:

The test will not compile, as we have not write in our code

TDD step 3.1

Write some code to make the test pass

```
import random

choices = ["1", "2", "3"]


class game(object):

    def rock_paper_scissor(self, choice, computer_choice):

        if choice == computer_choice:

            return 'Draw'

        elif choice == "1":

            if computer_choice == "3":

                return 'True'

            else:
```

```
            return 'False'
    elif choice == "2":
        if computer_choice == "1":
            return 'True'
        else:
            return 'False'
    elif choice == "3":
        if computer_choice == "2":
            return 'True'
        else:
            return 'False'
```

Step 4.1

Run all the tests again

```
Ran 1 test in 0.000s

OK
```

The test passed.

Conclusion:

We actually know about the testing and know the process about TDD from this assignment. TDD could be a practice when a software engineer composes a useful test before building a code. With the use of TDD, developer can easily understand the code will right or wrong. TDD makes upkeep and refactoring of code much simpler. TDD makes a difference to supply clarity during the usage handle and provides a safety net once need to refactor the code, needs to be fairly written. With TDD, collaboration gets to be much simpler and more productive. TDD saves time during the extend improvement stage, and a team can settle the code right absent when identifying a breakage.

Git-hub link: https://github.com/Amin-37/Software-process-and-tools