

**UNIT NAME: SOFTWARE ENGINEERING: PROCESS AND TOOLS (PRT582)**

**PROF: Charles Yeo**

**Assessment 1**

**SUBMITTED BY:**

**STUDENT NAME: Shanthi Annapareddy**

**STUDENT NUMBER: S355324**

**STUDY LOCATION: Casuarina Campus**

Rock, Paper, and Scissors game using TDD and Unit Testing

|  |  |  |
| --- | --- | --- |
| Student Name: Shanthi Annapareddy |  | Student ID: S355324 |

# Introduction

This project aims to develop a game that has two players. One player is the user, and the other player is the computer. The player will have three choices of inputs, and the same goes for the computer. Usually, this is a hand game in which players can form three hand shapes, and the shapes are paper, scissors, and rock. Depending upon the shapes, the winner or loser of the game is declared, and the game has three outcomes: Win, Lose or Tie. In this game, we used test-driven development approach to develop the project, and we used python to develop the code in visual studio. The framework we used to do the TTD, and unit testing for the project is Pytest. We also used Flake8 python on the project.

TTD software development approach in which test cases are developed to specify and verify what the code does. Simply put, first write a test case for each feature, test it, and if the test fails, write new code to make the test pass, making your code simple and error-free. Test-driven development begins by designing and developing tests for each small feature of your application. TDD frameworks direct developers to write new code only when automated tests fail. This avoids code duplication. The full form of TDD is Test Driven Development**.** This project was developed using Python. This is due to its popularity, availability, ease of use, and library compatibility (because Scrabble scoring is mostly string-based). Because pytest was so straightforward, it was simple to start building unit test cases and TDD.

Rules of the game:

|  |  |  |
| --- | --- | --- |
| **Player 1** | **Player 2** | **Win/Lose/Tie** |
| Rock | Rock | Tie |
| Rock | Paper | Lose |
| Rock | Scissors | Win |
| Paper | Rock | Win |
| Paper | Paper | Tie |
| Paper | Scissors | Lose |
| Scissors | Rock | Lose |
| Scissors | Paper | Win |
| Scissors | Scissors | Tie |

Some of the most important requirements are listed in the table below:

|  |  |
| --- | --- |
| Requirement | Function |
| User input | * Checking the input given by the user to see if it is in the choices * Whether users’ input is being taken in lower case * Checking the computers random selection. |
| Action process | * After every win the player should get appoint * Make sure the number of games is present * Scores are declared after every round |
| Output validation | * Winner is declared once a player scores 5 * Used being asked to quit or playagain |

# Process

The process followed for development of each unit(function) was to start off with writing test cases for that unit. Initially, the function would have empty/hardcoded logic and would fail the test case(s). The function is then developed by writing the relevant code for it and is tested against the test cases. When the function passes all the test cases written for it, additional test cases are written (mostly to ensure it works for special/corner cases) to verify its correctness. Thus, each unit is developed to pass the test cases and unit tests are used to confirm correctness.

Writing test cases for each unit was the first step in the procedure used to develop each test case. Initially, the function would have no logic and would fail the test scenarios. The function is later created by writing the needed code, and it is evaluated using the test cases. These test cases are again redeveloped and each test is individually run to understand and resolve issues faster. Additional test cases are written (usually to guarantee it works for special/corner cases to ensure it works for it) once the function passes all the test cases written for it to confirm its correctness. As a result, unit tests are used to validate correctness and each unit is designed to pass the test cases.

**3 functions were developed using TTD:**

* test\_GameawinsWithFivePoints(self) : This method is used to validate the the no. of game i.e. to make a player win if they score 5 points.
* test\_player\_wins\_computer\_with\_rock\_smashes\_paper(self): This method is used to check if the player wins after smashing rock with paper

The screenshots of the code during the TDD and respective unit tests are given below for each of the above mentioned units. The screenshots showcase the journey of code of each unit.

### test\_GameawinsWithFivePoints

Test case:

Text

Description automatically generated

Test Result:

Text

Description automatically generated

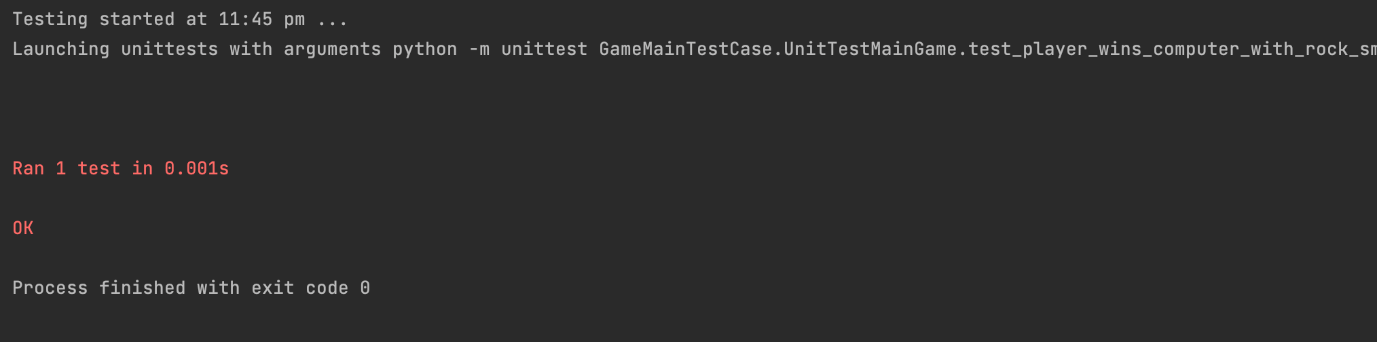
**test\_player\_wins\_computer\_with\_rock\_smashes\_paper(self):**

Code:

Graphical user interface, text, website

Description automatically generated

Test Result:



-------------------------------------------------------------------------------------------------------------------

## Conclusion

It was discovered that using test driven development was helpful when writing the code with various inputs and scenarios in mind. Writing test cases before writing code has made it less likely for programmes to overlook some edge circumstances. Additionally, creating test cases aided in detecting and fixing errors rapidly (as shown in one of the screenshots). It has been discovered that creating quality test cases is essential for TDD to work. TDD's ability to identify issues rapidly may accelerate development because bugs are discovered as soon as possible when the code is still in its infancy.

Overall, TTD has helped the game development easy and made a step-by-step process which benefited the development.

Github link: https://github.com/ShanthiAnnapareddy/Assignment-1-PRT582.git