

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

**PROF: Charles Yeo**

**Assessment 1**

**SUBMITTED BY:**

**STUDENT NAME: Mohammed Adnan**

**STUDENT NUMBER: S360958**

**STUDY LOCATION: Casuarina Campus**

Rock, Paper, Scissor using TDD and Unit Testing

|  |  |  |
| --- | --- | --- |
| Student Name: Mohammed Adnan |  | Student ID: S360958 |

# Introduction

Two players choose a move simultaneously while playing the game rock, paper, scissors. The winner is then determined by comparing this against the computer's choice. The project was developed using a test-driven development methodology. Python is the programming language, and Pytest and flake8 are the testing frameworks for TDD and unit testing.

The following table lists the winning guidelines:

| **Player choice** | **Computer** | **Result** |
| --- | --- | --- |
| Rock | Scissors | Player choice Win |
| Paper | Rock | Player choice Win |
| Scissors | Paper | Player choice Win |
| Rock | Rock | Draw |
| Paper | Paper | Draw |
| Scissors | Scissors | Draw |
| Rock | Paper | Computer win |
| Paper | Scissors | Computer win |
| Scissors | Rock | Computer win |

The following list includes some of the key requirements:

1. Scissors, Paper, or Rock is one of the possibilities that the computer chooses at random.
2. The player is then given the chance to select/enter one of the options of rock, paper, or scissors.
3. The victor receives one point.
4. The winner is the first player to score five points. Additionally, the overall number of rounds played will be shown.
5. The player is prompted to end or restart the game when the winner is announced.
6. Player may also leave the game whenever they choose.

**Test Driven Development (TDD):** To specify and verify what the code will do, test cases are created using the test-driven development (TDD) methodology. Simple terms, test cases for every functionality are built, tested, and if the test fails, new code is produced to pass the test and create clean, simple code.

**Flake8:** The Python package Flake8 includes PyFlakes, PyCodeStyle, and Ned Batchelder's McCabe script. It is a great toolkit for checking your code base for cyclomatic complexity and programming errors like "library imported but unused" and "Undefined name," as well as coding style (PEP8).

**Pytest:** Common activities need less code when using pytest, while sophisticated jobs can be completed using several time-saving commands and plugins. It will even automatically execute your current tests, including those created with unittest.

# Process

Writing test cases for each unit (or function) was the first step in the process for developing it. The function would initially have no logic and would fail the test scenario (s). The function is then created by creating the necessary code, and it is evaluated using the test cases. Additional test cases are written to confirm the function's correctness after it passes all of the test cases that were written for it (primarily to guarantee it works for special or corner instances). As a result, each unit is created to pass the test cases, and unit tests are used to verify accuracy.

3 functions were developed using TDD:

take\_player\_computer\_inputs

takeComputerInput

compareAndCountWins

runGameLogic

is\_Win

Test case 1:



Test Result:

Text

Description automatically generated with low confidence

Testcase 2 :

Graphical user interface, text, application

Description automatically generated

Test Result:

Text

Description automatically generated with medium confidence

## Conclusion

The use of test-driven development was found to be advantageous while writing the code with various inputs and scenarios in mind. Writing test cases before writing code has resulted in the development of less prone to overlook some edge situations code. Additionally, creating test cases aided in swiftly identifying and addressing errors (as shown in one of the screenshots). It has been discovered that the success of TDD depends on writing effective test cases. As flaws are discovered right immediately while the code is in its early phases, TDD's ability to uncover them rapidly may increase the rate of development.

Overall, TDD speeds up development if the test cases are cleverly structured and contributes to the delivery of a sturdy product.

Github link: https://github.com/8977201631/PRT582.git