**ROCK PAPER SCISSOR GAME IN PYTHON**

**Sinmei Li**

**(S360233)**

**Introduction:**

This is a rock-aper-scissor game project written in Python. Player will compete with computer, and the first to get 5 points wins the game. In this project I use TDD method to develop and test this program.

**Objective:**

This game allows a single player to play with a computer without time limitation. The game is easy to learn and is for all aged people. This is also my first Python project that implements TDD method.

**Requirement of this game:**

1. The computer randomly picks one of the options of scissor, paper, and rock.
2. Player is given the option to pick one of the options of scissor, paper, and rock.
3. One point is given to the winner.
4. The first to get 5 points wins the game.
5. At the end of the game, the total number of rounds played will also be displayed.
6. Once the winner is determined, the player is asked to quit or restart the game.
7. Player can also quit the game at any time.

**Tools used in this project:**

* Linting tools: Flake8 and Pylint
* TDD automated test tool: PyUnit

**To play the game: run call\_rps.game.py**

The player chooses of the options between scissor, paper, and rock. Then this compared against computer’s selection and determine who the winner is.

Game rules are as below:

* rock vs paper -> paper wins
* rock vs scissor -> rock wins
* paper vs scissor -> scissor wins

**Game procedure are as follows:**

1. Computer randomly picks one of the options of scissor, paper, and rock.
2. Player types a number to pick one of the options of scissor, paper, and rock.
3. Compare player’s selection against computer’s selection.
4. Determine the winner of this round, and 1 point is given to the winner.
5. The first to get 5 points is the winner of this game.
6. At the end, player can choose to continue playing the game or quit the game.

**Process:**

I program this rock-aper-scissor game project using python and TDD, and I create three files. The purpose of three python files for this game are:

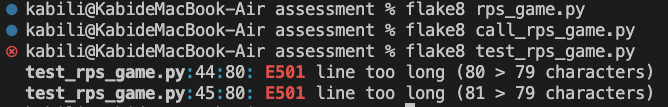
1. **test\_rps\_game.py:** the test file
2. **rps\_game.py: the** modules that record Python codes of the project
3. **call\_rps\_game.py:** the script file that is executed to run the game

To improve the quality of my codes, I apply two linting tools to improve my codes, Flake8 to check for my code base in logical errors, and Pylint to meet code standard and refactor codes in alternative way. By resolving the problems raised by Flake8 and Pylint, I start to pay attention to minor details like whitespace, and l also learn knowledge about naming style and start thinking of manipulating variables to print out message instead of repeating writing the same codes.

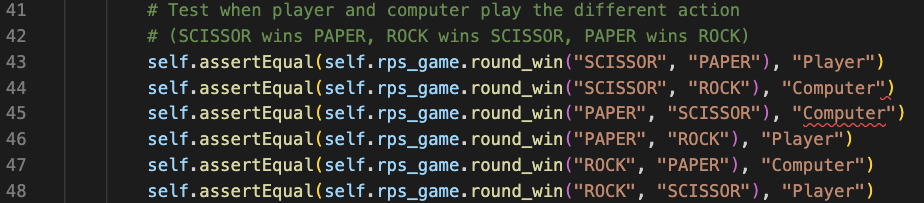
**Linting tool: Flake8**

When I first run Flake8 in rps\_game.py, it shows more than 20 problems to fix. Most of the problems are about trailing whitespace, missing whitespace, and line too long. The most difficult problem for me to resolve is “line too long” because it’s hard for a beginner like me to shorten lines with advance code. I investigate each line to find alternative ways to shorten the line like changing variable names, stating comment more concisely, or changing coding structure. However, after resolving the problems, I still have two “line too long” problems left in test\_rps\_game.py because the contents are all variable and functions which are not able to modify.





\*Screenshot of running Flake8 on the three python files.



\*Line too long problems of Flake8 in line 44 and line 45 in test\_rps\_game.py

**Linting tool: Pylint**

When I first run Pylint, there are even more problems than in Flake8, and most of the problems are not conforming to specific naming style, missing docstring and unnecessary else after return. Through this tool, I learn how to name Module, Method, and Argument with snake\_case naming style, and to name Class with Pascal Case naming style. Another new knowledge to me is “no-else-return” to remove the unnecessary else block if an if black contains a return statement.

Text

Description automatically generated

\*Screenshot of running Pylint on the three python files.

**TDD automated test tool: PyUnit**

Based on the requirement of rock-aper-scissor game, I make a flow chart to construct my codes and which parts I should test. To make it easier to play the game, I choose to make player enter a specific number to play an action instead of typing strings, for example, number 1 means “ROCK”. Please refer test\_rps\_game.py to get the test codes. I run three unit test and the purpose are:

1. Checking if the input is valid
2. Checking if the input number matches the assigned action
3. Checking if the game rules are followed and it returns the right winner.

Graphical user interface, text, application

Description automatically generated

\*screenshot of run PyUnit test.

**Test cases of rock-paper-scissor game:**

I conducted 4 test cases to ensure the game processing properly. Please see appendix for detailed records.

**Screenshots of playing the rock-paper-scissor game**

1. Welcome and start the game. Player needs to enter a number to pick one of the options of scissor, paper and rock.

Text

Description automatically generated

1. The computer randomly picks one of the options of scissor, paper, and rock.

One point is given to the winner.

Text

Description automatically generated with medium confidence

1. Continue the game until the player or the computer gets five points.

The first to get five points winds the game.

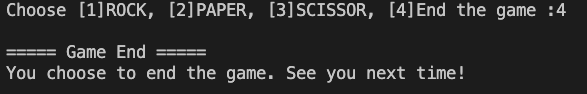
The total numbers of rounds played are displayed.

Once the winner is determined, the player is asked to quit or restart the game.

Text

Description automatically generated

1. Player can quit the game at any time.



**Conclusion:**

This is my first python project and my first TDD project. Through lecture, I know that TDD approach is to design and test before developing codes. However, I confused myself thinking that why we can’t do automated unit tests after developing codes. I didn’t realize the reasons until I started to write this project.

Honestly speaking, in the beginning I developed code first because I lacked coding experience and had no idea what functions or methods to use and how to organize my codes. After I finished my first version, my program ran successfully; however, I found that I couldn’t do unit tests because my program structure was not right to TDD approach. I realized the importance of designing and testing first, and most important of all, I learnt how to do it. I rewrote my project again with TDD approach, and this time I had further understanding about how TDD works and how to realize TDD approach in my project. From my experience, as a beginner in programming, rewriting a program with TDD approach may be a method to be familiar with TDD.

GitHub link: <https://github.com/kkkkabi/Rock_Paper_Scissor_Game.git>

**Appendix: Test Case**

Project Name: ROCK PAPER SCISSOR Game Module Name: rps\_game

Created By: Kabi Created Date: 10-12-2022

Executed by: Kabi Executed Date:12-12-2022

| **Test Case ID** | **Description** | **Test Step** | **Preconditions** | **Test Data** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TC\_Game\_01 | Verify functionality with valid input | Display welcome message |  |  | Able to see welcome message and start the game | As expected | Pass |
|  |  | Enter valid selection | Valid number | Input choose: 1 | Able to see the winning result of the round and continue the game | As expected | Pass |
|  |  | Enter valid selection(until end) | Valid number | Input choose: 1  Input choose: 2  (repeat until the end) | Able to see the winning result of each round and to play until anyone get 5 score | As expected | Pass |
|  |  | Display game result | Either one gets 5 scores |  | Able to see the winning result of the game | As expected | Pass |
|  |  | Enter to restart the game or not | Valid string | Input “Y” | Restart the game again | As expected | Pass |
| TC\_Game\_02 | Verify functionality with valid input | Display welcome message |  |  | Able to see welcome message and start the game | As expected | Pass |
|  |  | Enter valid selection | Valid number | Input choose: 1 | Able to see the winning result of the round and continue the game | As expected | Pass |
|  |  | Enter valid selection(until end) | Valid number | Input choose: 2  Input choose: 3  (repeat until the end) | Able to see the winning result of each round and to play until anyone get 5 score | As expected | Pass |
|  |  | Display game result | Either one gets 5 scores |  | Able to see the winning result of the game | As expected | Pass |
|  |  | Enter to restart the game or not | Valid string | Input “N” | quit the game | As expected | Pass |
| TC\_Game\_03 | Verify functionality with valid input | Display welcome message |  |  | Able to see welcome message and start the game | As expected | Pass |
|  |  | Enter valid selection | Valid number | Input choose: 2 | Able to see the winning result of the round and continue the game | As expected | Pass |
|  |  | Enter valid selection | Valid number | Input choose: 4 | Able to quit the game and display ending message | As expected | Pass |
|  |  | Display ending message |  |  | Able to see the ending message | As expected | Pass |
|  |  | End the game |  |  | Quit the game | As expected | Pass |
| TC\_Game\_04 | Verify functionality with one invalid input | Display welcome message |  |  | Able to see welcome message and start the game | As expected | Pass |
|  |  | Enter invalid selection | Invalid string | Input choose: “ROCK” | Prompt to choose again and continue the game | As expected | Pass |
|  |  | Enter valid selection(until end) | Valid number | Input choose: 2 | Able to see the winning result of each round and to play until anyone get 5 score | As expected | Pass |
|  |  | Display game result |  |  | Able to see the winning result of the game | As expected | Pass |
|  |  | Enter to restart the game or not |  |  | Restart the game again | As expected | Pass |