Prashil Koirala

S366956

PRT582

Software Unit testing report

# Introduction

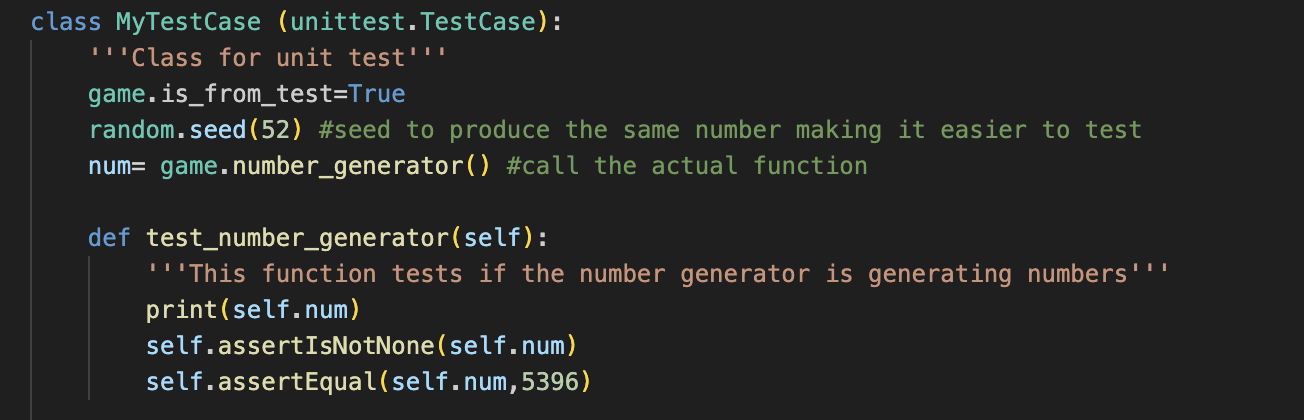
In this report, I describe how I used TDD in python to create a number guessing game. Firstly, I will outline the requirements of the game:

1. The game randomly generates a 4-digit number.
2. The program will keep asking the user to guess the number until the player guesses it correctly or has quitted.
3. The program will give hints to the used based on their inputs. An X means a digit is in the wrong spot, a circle means a digit is in the right spot and \_ means the random number does not contain the digit.
4. Once the game is over, the program will display in how many attempts the user won the game.
5. The user can quit anytime by pressing enter.
6. The program asks if the user wants to play again after they win.

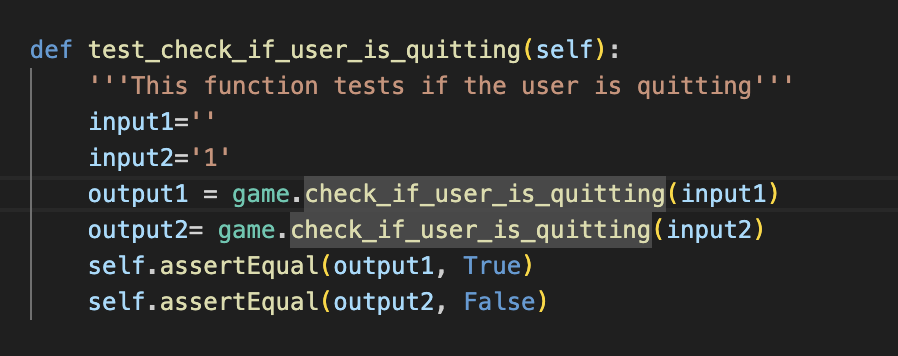
I will be using ‘unit test’ module of python as the testing tool.

# Process

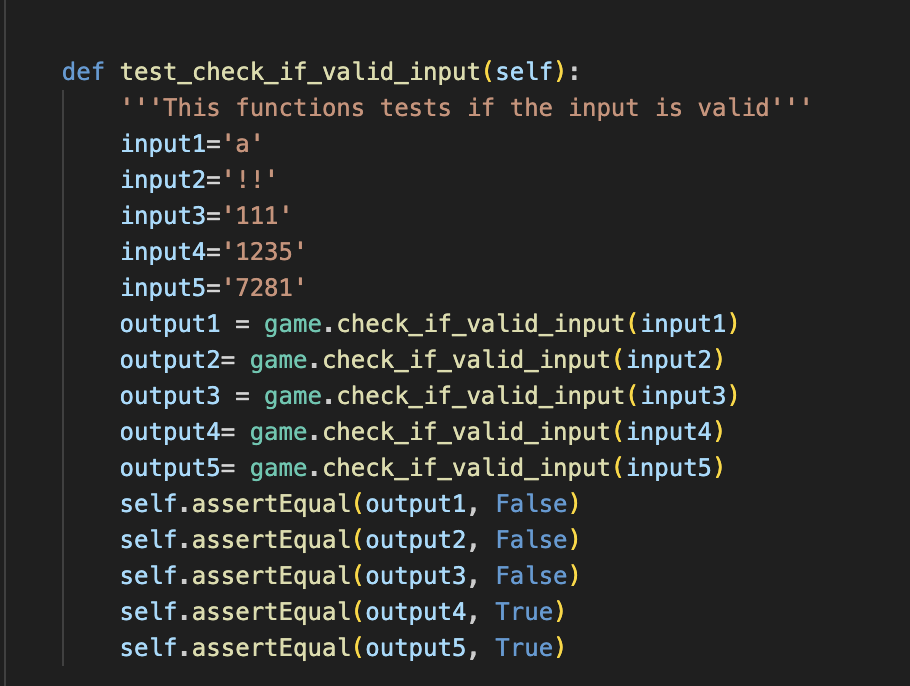
I started by writing the first simple test case for my program. It was to check whether the game generates a random number or not.



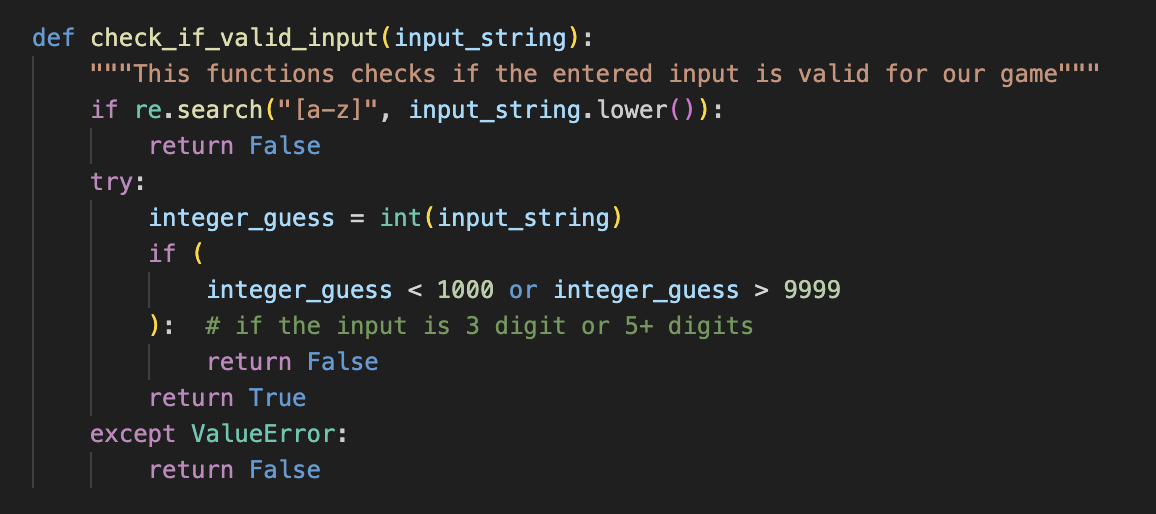
I used a seed so I can check whether the random number is equal every time I run the test. This test failed in the beginning. Then I wrote the code to make it pass. Then, I wrote the test to see if the user wanted to quit the game. The user would have to press enter to quit. Any other input meant the user was not quitting.



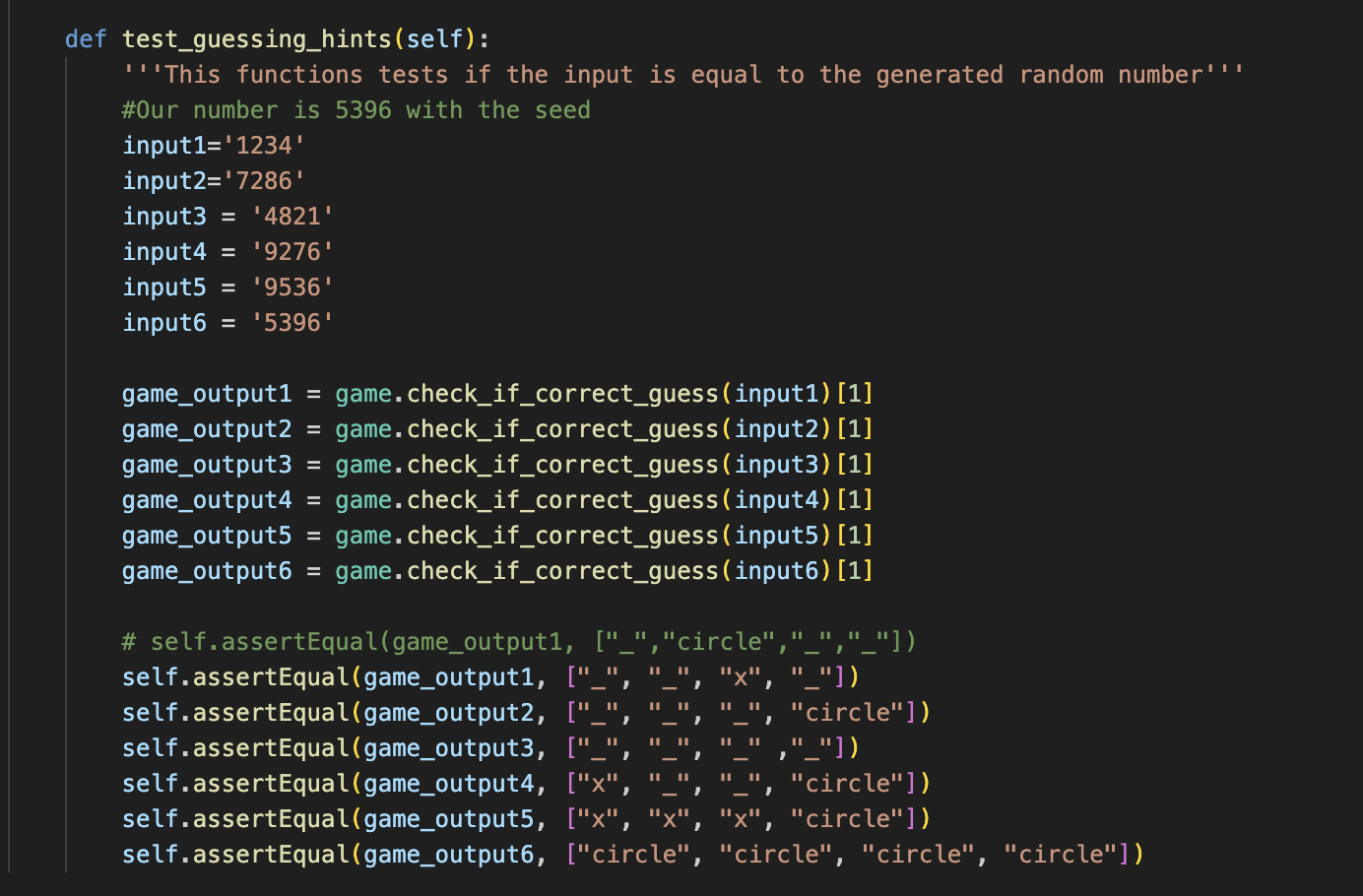
The next step in my program was to write a test to test the validity of the inputs. Getting user input for every run of the test was not feasible, so I wrote the test and program in such a way that I could give hard-coded inputs for the program to test. Only 4-digit numbers are treated as valid inputs by the program.

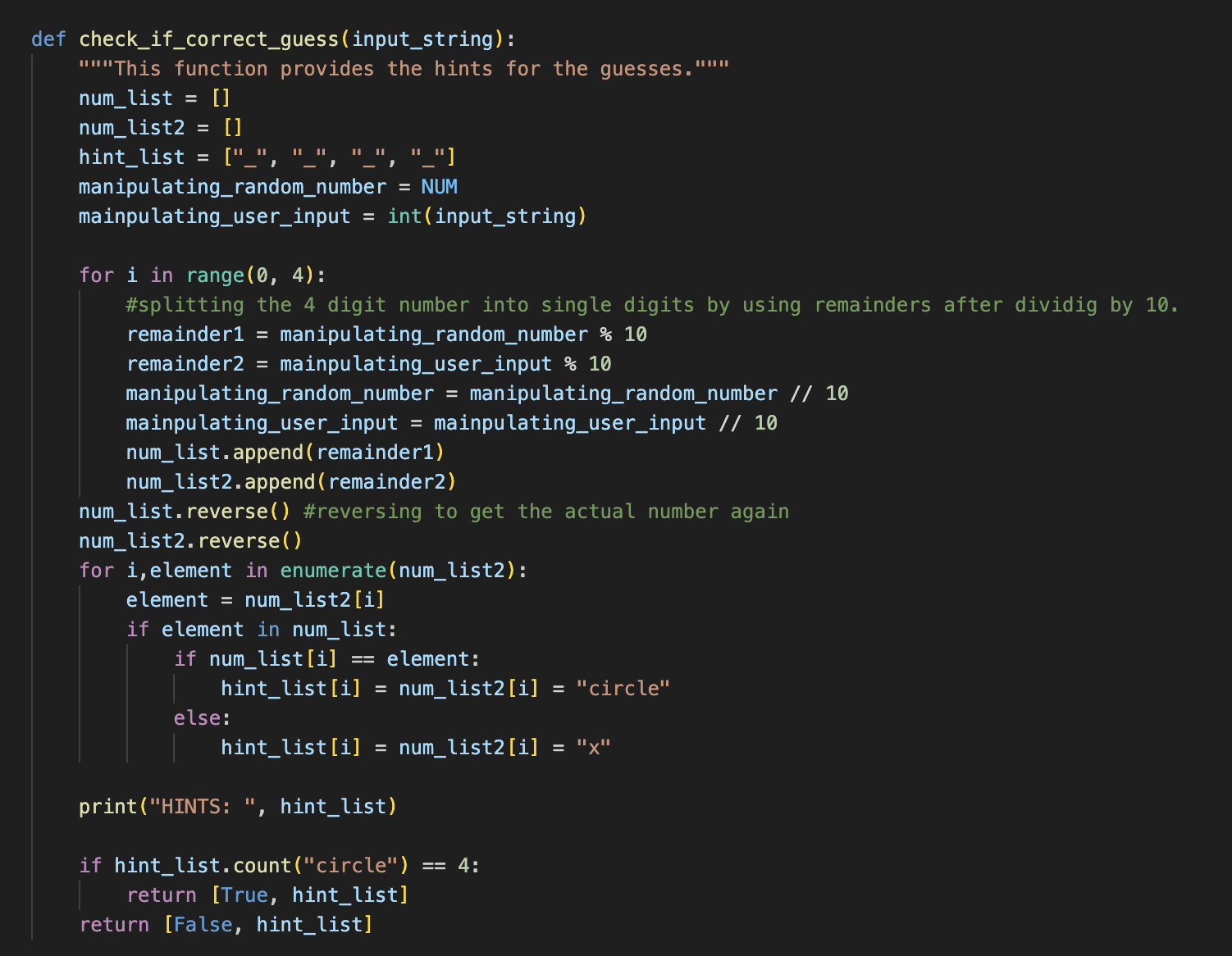


After the test failed, I wrote the code for the actual game to check input validity.



Any alphabets were immediately invalid. With a try and except block, I raised an error for any input that was not an integer. In this way, I handled the inputs.

With all this done, the next step was to guess the number and provide hints to the user. I wrote a test case first with hardcoded inputs and hardcoded number to be guessed. 

I decided that I would be returning a list of a boolean and the list of hints from the function that checks the input against the random generated number. In the test, the random number was set to 5396. So, using the assertEqual method, I was able to expect what output should be returned for each of the inputs. The tests failed and then, I coded the function. 

After this the tests passed and I had a fully functioning game. Then I used pylint to obtain a 10/10 score on my code for both game.py and game\_test.py.

# Conclusion

I learnt that with TDD, it leaves very less margin for error when it comes to unit testing. I thought of all the scenarios that need to be handled by my code before actually writing the code. Then, I made test cases, which failed. After I wrote the codes to make the tests pass, my code was basically error free as all the possible errors were handled during the process of making the tests pass. I finally used GitHub to upload the codes.