



Unit Code: PRT582

Software Unit Testing Report

“Guess It Win It” game using TDD

Submitted by,

Sharon Cheethayil Sasidharan

Student ID: s366502

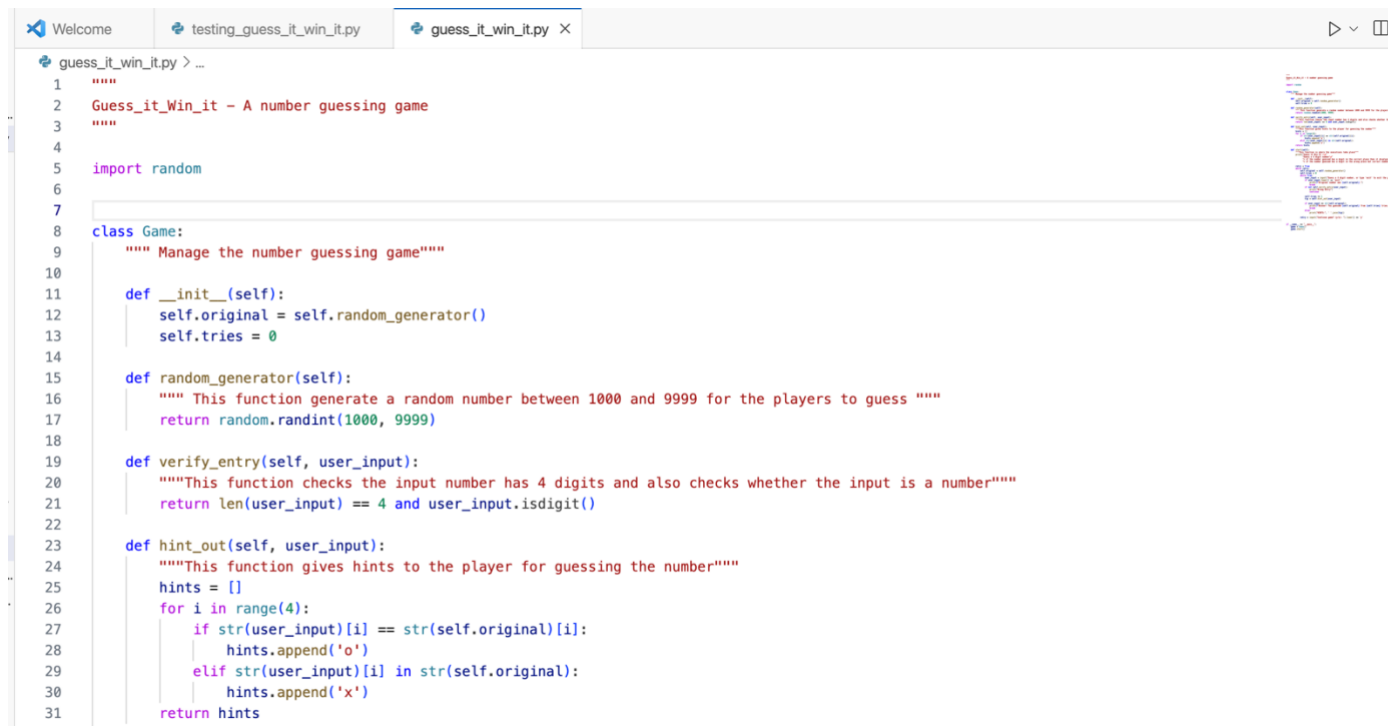
Introduction

The main goal of this project is to create a number guessing game, "Guess it Win it." Using an automated unit testing framework provided by the 'unit test' library, the primary focus is on assuring the accuracy and dependability of the game. In this Project, I used automated tools like Flake8 and Pylint to ensure uniform code quality and readability and I followed Test Driven Development (TDD) for making code that is I write test before the original code so that it describe how the code works.

Project requirements and objectives

A clear set of goals and specifications are what are driving this project:

- Create a guessing game with a dynamic 4-digit number that is generated at random between 1000 and 9999. Player can predict the number in several tries and is given an option to quit after guessing.
- Implement a hint system that provides players with feedback after each guess. The system should show accurately located digit as ('o') and correctly predicted but incorrectly placed as ('x').
- Create a framework for automated unit testing to check the accuracy and functioning of full requirements and logic
- Use Flake8 with Pylint to check that code complies with coding standards, fostering readable and consistent programming.



```
1  """
2  Guess_it_Win_it - A number guessing game
3  """
4
5  import random
6
7
8  class Game:
9      """ Manage the number guessing game """
10
11      def __init__(self):
12          self.original = self.random_generator()
13          self.tries = 0
14
15      def random_generator(self):
16          """ This function generate a random number between 1000 and 9999 for the players to guess """
17          return random.randint(1000, 9999)
18
19      def verify_entry(self, user_input):
20          """This function checks the input number has 4 digits and also checks whether the input is a number"""
21          return len(user_input) == 4 and user_input.isdigit()
22
23      def hint_out(self, user_input):
24          """This function gives hints to the player for guessing the number"""
25          hints = []
26          for i in range(4):
27              if str(user_input)[i] == str(self.original)[i]:
28                  hints.append('o')
29              elif str(user_input)[i] in str(self.original):
30                  hints.append('x')
31          return hints
```

1. This function generates a number between 1000 and 9999 so that the player can guess the number. The Game class, a coordinator of the game's complexities, is the keystone of this file. This class includes a number of crucial steps that together make up the gameplay. The game begins with the call to `__init__(self)`. It starts the game by creating a mysterious four- and initialising the attempt counter. The `verify_entry` function makes sure that the input is made up of 4 digits and is numeric.

```
def test_random_generation_testcase(self):
    """Test whether the randomly generated number is valid for the game"""
    random_number = self.guessitwinit.random_generator()
    self.assertTrue(1000 <= random_number <= 9999)
    print("Unit Test case Random Number generation: Passed ")
```

2. The core of the gameplay's strategic component is the `hint_out` function, which can be accessed via the `user_input`. The function gives hints based on player input and leads players to the solution by displaying a 'o' for correctly positioned digits and a 'x' for correctly guessed but misplaced numbers.

```
def hint_out(self, user_input):
    """This function gives hints to the player for guessing the number"""
    hints = []
    for i in range(4):
        if str(user_input)[i] == str(self.original)[i]:
            hints.append('o')
        elif str(user_input)[i] in str(self.original):
            hints.append('x')
    return hints
```

3. The flow of the game is determined by the `start()`. This is the function that determines and validates the user entry. This component is crucial because in this component we check the users desire to continue the game or quit the game and also wrong entries are determined.

```
def start(self):
    """This function is where the executions take place"""
    print("Guess it Win It -\n")
    print("Guess a 4 digit number\n")
    print("* if the number guessed has a digit in the correct place then it displays - 0\n")
    print("* if the number guessed has a digit in the wrong place but correct number then it displays - X ")

    retry = True
    while retry:
        self.original = self.random_generator()
        self.tries = 0
        while True:
            user_input = input("Guess a 4 digit number, or type 'exit' to exit the game: ")
            if user_input.lower() == 'exit':
                print(f"Original number was {self.original}.")
                break
            if not self.verify_entry(user_input):
                print("Wrong Entry")
                continue

            self.tries += 1
            tip = self.hint_out(user_input)

            if user_input == str(self.original):
                print(f"Winner! You guessed {self.original} from {self.tries} tries.")
                break
            else:
                print("HINTS:", ' '.join(tip))

        retry = input("Continue game? (y/n): ").lower() == 'y'
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
● (base) sharoncs@sharoncs-MBP Documents % /usr/bin/python3 /Users/sharoncs/Documents/testing_guess_it_win_it.py
Unit Test case datatypes: Passed
.Unit Test case function hint_out: Passed
.Unit Test case user entry functions: Passed
.Unit Test case Random Number generation: Passed
.
-----
Ran 4 tests in 0.000s
OK
○ (base) sharoncs@sharoncs-MBP Documents %
```

I created 4 test cases that check the random number creation, datatypes of variables, the hint generation function and a function to validate and verify the user inputs

test_hint_out_testcase: The objective of this test is to validate that the hint_out function generates hints that only contain the characters 'o' and 'x' in accordance with the game's rules.

test_random_generation_testcase: Carefully examines the random_generator function's consistency in generating valid 4-digit values within the prescribed range.

test_datatype_testcase: The focus of this test is on how well-coordinated the data structures are inside the Game class. A functioning system needs data type validation to work in a good way.

test_input_testCase: This test carefully examines the verify_entry function, evaluating its capability to precisely assess both valid and incorrect user inputs.

```
def test_input_testcase(self):
    """Test input functions that take values from the user"""
    self.guessitwinit.user_input = '1111'
    self.assertTrue(self.guessitwinit.verify_entry(self.guessitwinit.user_input))

    self.guessitwinit.user_input = 'Wrongentry'
    self.assertFalse(self.guessitwinit.verify_entry(self.guessitwinit.user_input))
    print("Unit Test case user entry functions: Passed")

def test_random_generation_testcase(self):
    """Test whether the randomly generated number is valid for the game"""
    random_number = self.guessitwinit.random_generator()
    self.assertTrue(1000 <= random_number <= 9999)
    print("Unit Test case Random Number generation: Passed ")

def test_datatype_testcase(self):
    """Test and verify data types"""
    self.assertTrue(isinstance(self.guessitwinit.hint_out(1234), list))
    self.assertTrue(isinstance(self.guessitwinit.tries, int))
    self.assertTrue(isinstance(self.guessitwinit.original, int))
    print("Unit Test case datatypes: Passed")

def test_hint_out_testcase(self):
    """Test the hint given out to the user"""
    user_input = "1243"
    hints = self.guessitwinit.hint_out(user_input)
    self.assertTrue(all(item in ['x', 'o'] for item in hints))
    print("Unit Test case function hint_out: Passed")
```

QUALITY CHECK:

Flake8 and Pylint, two automated code quality evaluation tools, are used to examine and validate the code.

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

guess_it_win_it.py:37:80: E501 line too long (95 > 79 characters)
guess_it_win_it.py:38:80: E501 line too long (112 > 79 characters)
guess_it_win_it.py:45:80: E501 line too long (95 > 79 characters)
guess_it_win_it.py:57:80: E501 line too long (90 > 79 characters)
(base) sharoncs@sharons-MBP Documents % python -m pylint guess_it_win_it.py
***** Module guess_it_win_it
guess_it_win_it.py:16:0: C0301: Line too long (101/100) (line-too-long)
guess_it_win_it.py:20:0: C0301: Line too long (110/100) (line-too-long)
guess_it_win_it.py:38:0: C0301: Line too long (112/100) (line-too-long)
guess_it_win_it.py:56:16: R1723: Unnecessary "else" after "break", remove the "else" and de-indent the code inside it (no-else-break)

Your code has been rated at 9.02/10 (previous run: 8.78/10, +0.24)

(base) sharoncs@sharons-MBP Documents %
```

Got 9.02/10 rating in pylint for the game code

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

.Unit Test case Random Number generation: Passed
.
-----
Ran 4 tests in 0.000s

OK
(base) sharoncs@sharons-MBP Documents % python -m flake8 testing_guess_it_win_it.py
testing_guess_it_win_it.py:19:80: E501 line too long (85 > 79 characters)
testing_guess_it_win_it.py:22:80: E501 line too long (86 > 79 characters)
(base) sharoncs@sharons-MBP Documents % python -m pylint testing_guess_it_win_it.py

Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00)

(base) sharoncs@sharons-MBP Documents %
```

Got 10/10 rating in pylint for testing_guess_it_win_it.py

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

testing_guess_it_win_it.py:37:1: W293 blank line contains whitespace
(base) sharoncs@sharons-MBP Documents % /usr/bin/python3 /Users/sharoncs/Documents/testing_guess_it_win_it.py
Unit Test case datatypes: Passed
.Unit Test case function hint_out: Passed
.Unit Test case user entry functions: Passed
.Unit Test case Random Number generation: Passed
.
-----
Ran 4 tests in 0.000s

OK
(base) sharoncs@sharons-MBP Documents % python -m flake8 testing_guess_it_win_it.py
testing_guess_it_win_it.py:19:80: E501 line too long (85 > 79 characters)
testing_guess_it_win_it.py:22:80: E501 line too long (86 > 79 characters)
(base) sharoncs@sharons-MBP Documents %
```

Flake8 output for testing_guess_it_win_it.py

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
(base) sharoncs@sharons-MBP Documents % python -m flake8 guess_it_win_it.py
guess_it_win_it.py:16:80: E501 line too long (101 > 79 characters)
guess_it_win_it.py:20:80: E501 line too long (110 > 79 characters)
guess_it_win_it.py:37:80: E501 line too long (95 > 79 characters)
guess_it_win_it.py:38:80: E501 line too long (112 > 79 characters)
guess_it_win_it.py:45:80: E501 line too long (95 > 79 characters)
guess_it_win_it.py:57:80: E501 line too long (90 > 79 characters)
(base) sharoncs@sharons-MBP Documents %
```

Flake8 output for guess_it_win_it.py

Output for different cases

1.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Guess a 4 digit number, or type 'exit' to exit the game: 3333
HINTS: x o o x
Guess a 4 digit number, or type 'exit' to exit the game: 4444
HINTS:
Guess a 4 digit number, or type 'exit' to exit the game: 5555
HINTS:
Guess a 4 digit number, or type 'exit' to exit the game: 6666
HINTS:
Guess a 4 digit number, or type 'exit' to exit the game: 7777
HINTS:
Guess a 4 digit number, or type 'exit' to exit the game: 8888
HINTS: o x x o
Guess a 4 digit number, or type 'exit' to exit the game: 8338
Winner! You guessed 8338 from 9 tries.
Continue game? (y/n):
```

2.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Your code has been rated at 9.02/10 (previous run: 8.78/10, +0.24)
(base) sharoncs@sharons-MBP Documents % /usr/bin/python3 /Users/sharoncs/Documents/guess_it_win_it.py
Guess it Win It -
Guess a 4 digit number
* if the number guessed has a digit in the correct place then it displays - 0
* if the number guessed has a digit in the wrong place but correct number then it displays - X
Guess a 4 digit number, or type 'exit' to exit the game: 2345
HINTS: x x
Guess a 4 digit number, or type 'exit' to exit the game: exit
Original number was 3289.
Continue game? (y/n): y
Guess a 4 digit number, or type 'exit' to exit the game:
```

3.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Guess a 4 digit number
* if the number guessed has a digit in the correct place then it displays - 0
* if the number guessed has a digit in the wrong place but correct number then it displays - X
Guess a 4 digit number, or type 'exit' to exit the game: 2345
HINTS: x x
Guess a 4 digit number, or type 'exit' to exit the game: exit
Original number was 3289.
Continue game? (y/n): y
Guess a 4 digit number, or type 'exit' to exit the game: 5567
HINTS: o x
Guess a 4 digit number, or type 'exit' to exit the game: jjjj
Wrong Entry
Guess a 4 digit number, or type 'exit' to exit the game: 6hg6
Wrong Entry
Guess a 4 digit number, or type 'exit' to exit the game:
```

Conclusion

In conclusion, the creation of the "Guess_it_Win_it" game combines creative functions with systematic testing procedures. The attraction of the game is its ability to entertain players while also exercising players' thinking skills through logic. Here I used the TDD method for development so that I planned before the creation. With the careful application of Flake8 and pylint, code quality is maintained in the background. Overall by using these best practises, the project achieves an ideal balance between creativity, reliability, and sustainability.

GitHub Repository Link: https://github.com/sharoncs888/PRT582_Sharon