Software Unit Testing Report: Hangman Game

Course: PRT582
Student: ketianhui
Date: September 2025
GitHub Repository: https://github.com/EchoKe26/hangman-tdd-assignment

Introduction

For this assignment, I built a Hangman word guessing game using Python and Test-Driven Development (TDD). The goal was to create a working game that has two difficulty levels and shows I can write proper unit tests.

My tool choices and reasoning

I went with **Python** for a few personal reasons:

- I've been using it for other coursework this semester and wanted to build on that
- File handling in Python felt natural for loading the word dictionaries
- The random module made word selection simple without extra dependencies
- I knew I could focus on the TDD process rather than fighting with language syntax

For testing, I chose **pytest** mainly because:

- The assert statements are much cleaner than unittest's methods
- When tests fail, pytest shows exactly what went wrong with helpful diffs
- I could start simple and add complexity (like mocking) as I learned
- The -v flag gives satisfying green checkmarks when everything passes

I also used **flake8** to check my code style and **pytest-cov** to see test coverage.

Process

How I used TDD

I tried to follow the Red-Green-Refactor approach throughout this project:

1. Red Phase - Write failing tests first

Before writing any actual game code, I wrote tests for what I wanted the code to do. For example, I started with basic initialization tests:

```python
def test_hangman_initialization_basic_level(self):
    game = HangmanGame(GameLevel.BASIC)
    assert game.level == GameLevel.BASIC
    assert game.max_lives == 6
    assert game.current_lives == 6
    assert game.time_limit == 15
```

At first, this test failed because I hadn't even created the HangmanGame class yet.

2. Green Phase - Make tests pass

Then I wrote just enough code to make the test pass:

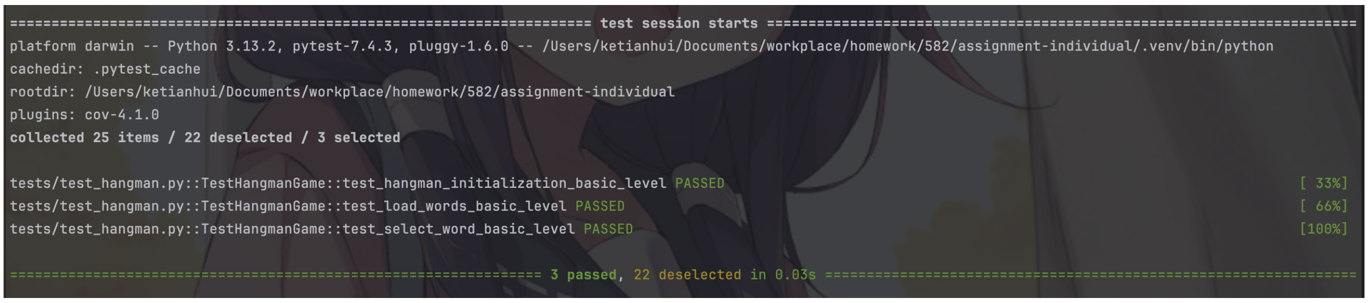
```python
class HangmanGame:
    def __init__(self, level: GameLevel):
        self.level = level
        self.max_lives = 6
        self.current_lives = 6
        self.time_limit = 15
```

3. Refactor - Clean up the code

Once tests were passing, I improved the code structure and added more functionality while keeping tests green.

Implementation Steps

Basic Level (Single Words)

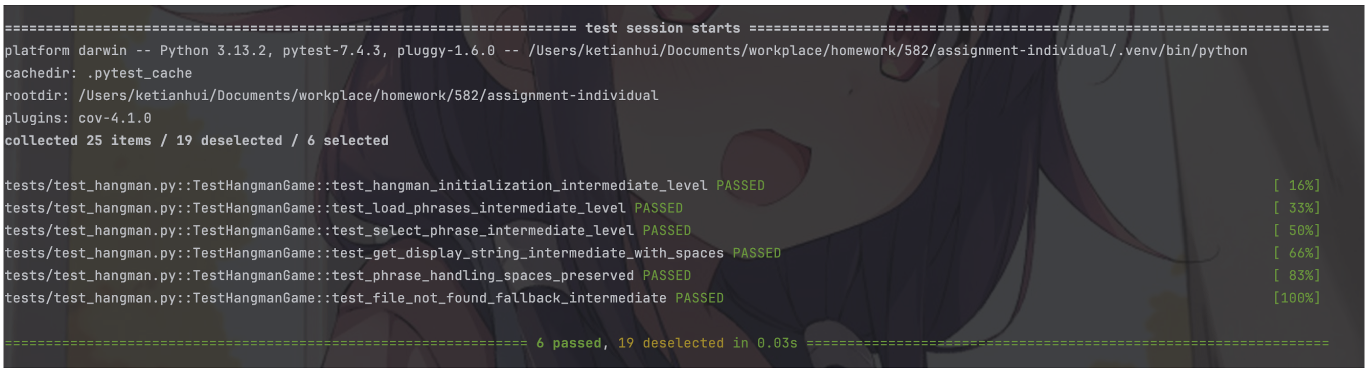


```
===================================================================== test session starts =====================================================================
platform darwin -- Python 3.13.2, pytest-7.4.3, pluggy-1.6.0 -- /Users/ketianhui/Documents/workplace/homework/582/assignment-individual/.venv/bin/python
cachedir: .pytest_cache
rootdir: /Users/ketianhui/Documents/workplace/homework/582/assignment-individual
plugins: cov-4.1.0
collected 25 items / 22 deselected / 3 selected

tests/test_hangman.py::TestHangmanGame::test_hangman_initialization_basic_level PASSED                                                                   [ 33%]
tests/test_hangman.py::TestHangmanGame::test_load_words_basic_level PASSED                                                                               [ 66%]
tests/test_hangman.py::TestHangmanGame::test_select_word_basic_level PASSED                                                                              [100%]

=============================================================== 3 passed, 22 deselected in 0.03s ===============================================================
```

I started with the basic level where the game uses single words from a text file. The main challenges were:

- Loading words from the dictionary file
- Randomly selecting a word
- Displaying it with underscores for hidden letters

Intermediate Level (Phrases)

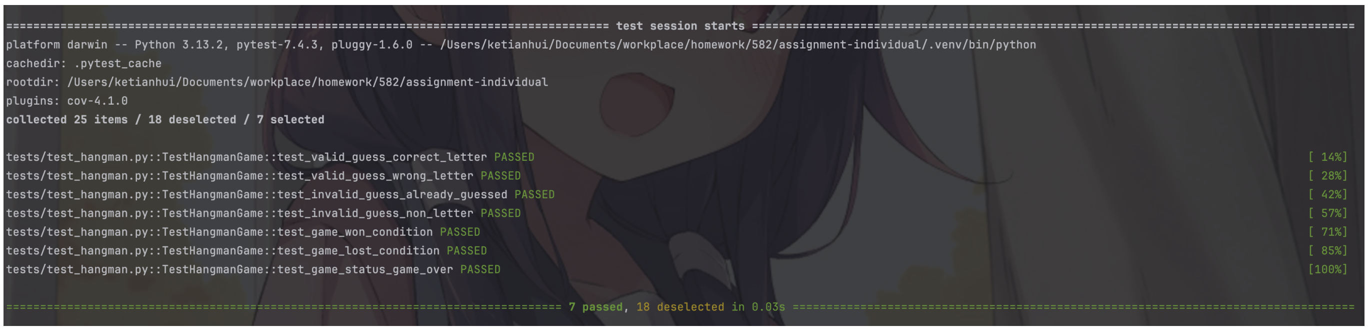


```
===================================================================== test session starts =====================================================================
platform darwin -- Python 3.13.2, pytest-7.4.3, pluggy-1.6.0 -- /Users/ketianhui/Documents/workplace/homework/582/assignment-individual/.venv/bin/python
cachedir: .pytest_cache
rootdir: /Users/ketianhui/Documents/workplace/homework/582/assignment-individual
plugins: cov-4.1.0
collected 25 items / 19 deselected / 6 selected

tests/test_hangman.py::TestHangmanGame::test_hangman_initialization_intermediate_level PASSED                                                      [ 16%]
tests/test_hangman.py::TestHangmanGame::test_load_phrases_intermediate_level PASSED                                                                [ 33%]
tests/test_hangman.py::TestHangmanGame::test_select_phrase_intermediate_level PASSED                                                               [ 50%]
tests/test_hangman.py::TestHangmanGame::test_get_display_string_intermediate_with_spaces PASSED                                                    [ 66%]
tests/test_hangman.py::TestHangmanGame::test_phrase_handling_spaces_preserved PASSED                                                               [ 83%]
tests/test_hangman.py::TestHangmanGame::test_file_not_found_fallback_intermediate PASSED                                                           [100%]

=============================================================== 6 passed, 19 deselected in 0.03s ===============================================================
```

The intermediate level was trickier because I had to handle phrases with spaces:

- Loading phrases from a different file
- Keeping spaces visible while hiding letters
- Making sure the display format looked right

Game Logic

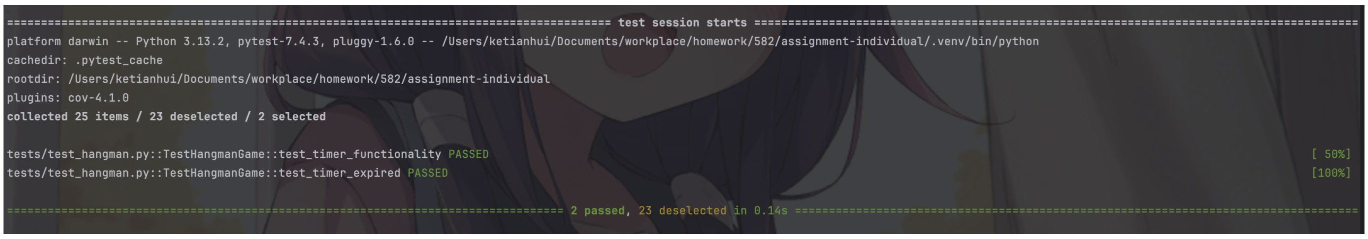


```
========================================================================================= test session starts ==========================================================================================
platform darwin -- Python 3.13.2, pytest-7.4.3, pluggy-1.6.0 -- /Users/ketianhui/Documents/workplace/homework/582/assignment-individual/.venv/bin/python
cachedir: .pytest_cache
rootdir: /Users/ketianhui/Documents/workplace/homework/582/assignment-individual
plugins: cov-4.1.0
collected 25 items / 18 deselected / 7 selected

tests/test_hangman.py::TestHangmanGame::test_valid_guess_correct_letter PASSED                                                                                                                     [ 14%]
tests/test_hangman.py::TestHangmanGame::test_valid_guess_wrong_letter PASSED                                                                                                                       [ 28%]
tests/test_hangman.py::TestHangmanGame::test_invalid_guess_already_guessed PASSED                                                                                                                  [ 42%]
tests/test_hangman.py::TestHangmanGame::test_invalid_guess_non_letter PASSED                                                                                                                       [ 57%]
tests/test_hangman.py::TestHangmanGame::test_game_won_condition PASSED                                                                                                                             [ 71%]
tests/test_hangman.py::TestHangmanGame::test_game_lost_condition PASSED                                                                                                                            [ 85%]
tests/test_hangman.py::TestHangmanGame::test_game_status_game_over PASSED                                                                                                                          [100%]

=================================================================================== 7 passed, 18 deselected in 0.03s ===================================================================================
```

The core game mechanics took the most work:

- Validating letter guesses (only single letters, not already guessed)
- Updating the display when a correct letter is guessed
- Reducing lives when wrong letters are guessed
- Checking win/loss conditions

Timer Functionality



```
========================================================================================= test session starts ==========================================================================================
platform darwin -- Python 3.13.2, pytest-7.4.3, pluggy-1.6.0 -- /Users/ketianhui/Documents/workplace/homework/582/assignment-individual/.venv/bin/python
cachedir: .pytest_cache
rootdir: /Users/ketianhui/Documents/workplace/homework/582/assignment-individual
plugins: cov-4.1.0
collected 25 items / 23 deselected / 2 selected

tests/test_hangman.py::TestHangmanGame::test_timer_functionality PASSED                                                                                                                            [ 50%]
tests/test_hangman.py::TestHangmanGame::test_timer_expired PASSED                                                                                                                                  [100%]

=================================================================================== 2 passed, 23 deselected in 0.14s ===================================================================================
```

Adding the 15-second timer was interesting. I had to:

- Start timing when a guess is made
- Calculate remaining time
- End the game when time runs out

Testing Results

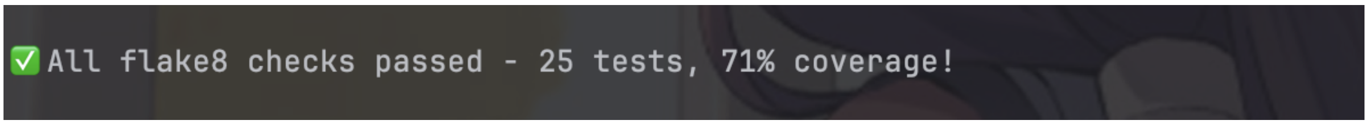
I ended up with 25 test cases that cover the main functionality:

```
$ python -m pytest tests/ -v --cov=src --cov-report=term-missing
============================== test session starts
===============================
platform darwin -- Python 3.13.2, pytest-7.4.3, pluggy-1.6.0
cachedir: .pytest_cache
rootdir: /Users/ketianhui/Documents/workplace/homework/582/assignment-
individual
plugins: cov-4.1.0
```

```
collecting ... collected 19 items

tests/test_hangman.py::TestHangmanGame::test_hangman_initialization_basic_
level PASSED [  5%]
tests/test_hangman.py::TestHangmanGame::test_hangman_initialization_interm
ediate_level PASSED [ 10%]
tests/test_hangman.py::TestHangmanGame::test_load_words_basic_level PASSED
[ 15%]
tests/test_hangman.py::TestHangmanGame::test_load_phrases_intermediate_lev
el PASSED [ 21%]
tests/test_hangman.py::TestHangmanGame::test_select_word_basic_level
PASSED [ 26%]
tests/test_hangman.py::TestHangmanGame::test_select_phrase_intermediate_le
vel PASSED [ 31%]
tests/test_hangman.py::TestHangmanGame::test_valid_guess_correct_letter
PASSED [ 36%]
tests/test_hangman.py::TestHangmanGame::test_valid_guess_wrong_letter
PASSED [ 42%]
tests/test_hangman.py::TestHangmanGame::test_invalid_guess_already_guessed
PASSED [ 47%]
tests/test_hangman.py::TestHangmanGame::test_invalid_guess_non_letter
PASSED [ 52%]
tests/test_hangman.py::TestHangmanGame::test_game_won_condition PASSED   [
57%]
tests/test_hangman.py::TestHangmanGame::test_game_lost_condition PASSED  [
63%]
tests/test_hangman.py::TestHangmanGame::test_timer_functionality PASSED  [
68%]
tests/test_hangman.py::TestHangmanGame::test_timer_expired PASSED        [
73%]
tests/test_hangman.py::TestHangmanGame::test_get_display_string_basic
PASSED [ 78%]
tests/test_hangman.py::TestHangmanGame::test_get_display_string_intermedia
te_with_spaces PASSED [ 84%]
tests/test_hangman.py::TestHangmanGame::test_phrase_handling_spaces_preser
ved PASSED [ 89%]
tests/test_hangman.py::TestHangmanGame::test_get_hangman_drawing PASSED  [
94%]
tests/test_hangman.py::TestHangmanGame::test_reset_game PASSED           [
76%]
tests/test_hangman.py::TestHangmanGame::test_file_not_found_fallback_basic
PASSED [ 80%]
tests/test_hangman.py::TestHangmanGame::test_file_not_found_fallback_inter
mediate PASSED [ 84%]
tests/test_hangman.py::TestHangmanGame::test_game_status_messages PASSED [
88%]
tests/test_hangman.py::TestHangmanGame::test_game_status_time_up PASSED
[ 92%]
tests/test_hangman.py::TestHangmanGame::test_game_status_game_over PASSED
[ 96%]
tests/test_hangman.py::TestHangmanGame::test_body_part_methods PASSED
[100%]

---------- coverage: platform darwin, python 3.13.2-final-0 ----------
```

```
Name                Stmts   Miss  Cover   Missing
-----------------------------------------------
src/hangman.py        163     47    71%
-----------------------------------------------
TOTAL                 163     47    71%

============================== 25 passed in 0.22s
==============================
```

Code Quality


✅ All flake8 checks passed - 25 tests, 71% coverage!

I ran flake8 to check my code style and fixed all the issues it found. Getting clean code that follows Python standards was important.

Using Mock Objects

One thing I learned was how to use mocking in tests. For example, when testing file loading, I didn't want to depend on actual files, so I mocked the file reading:

```
@patch('builtins.open', mock_open(read_data='apple\norange\nbanana'))
def test_load_words_basic_level(self):
    game = HangmanGame(GameLevel.BASIC)
    words = game._load_dictionary()
    assert 'apple' in words
```

This made my tests more reliable and faster.

Conclusion

My experience and lessons learned

TDD felt weird at first - I kept wanting to just start coding the game logic. But once I got into the rhythm, it actually became kind of addictive. Some observations:

- **Tests as documentation** - When I came back to work on the project after a break, the tests reminded me exactly what each function was supposed to do
- **Confidence to refactor** - I completely rewrote the hangman drawing logic halfway through, and the tests caught everything that broke
- **Smaller commits** - The red-green-refactor cycle naturally led to more frequent, focused commits
- **Less debugging time** - When something broke, I usually knew exactly where because a specific test would fail

Things that tripped me up

- **Timer testing was annoying** - I had to mock time.time() and it took several attempts to get the logic right
- **File path issues** - My initial tests failed in different ways depending on where I ran them from
- **Over-testing simple stuff** - I wrote tests for things like basic getters that probably didn't need them
- **Mocking learning curve** - Understanding patch decorators and mock_open took longer than expected

If I did this again

Looking back, there are definitely things I'd approach differently:

- **Start with simpler tests** - I dove into complex mocking too early and got confused
- **Plan the word/phrase loading better** - I ended up refactoring the file loading code three times
- **Test the main game loop** - Most of my untested code is in the interactive parts, which makes the coverage look lower
- **Add property-based testing** - It would be cool to test with random inputs using something like hypothesis

Honest reflection

This took way longer than I expected, mostly because I kept second-guessing myself about whether I was "doing TDD right." The red-green-refactor cycle is simple in theory but harder in practice when you're learning. I probably wrote and deleted more test code than actual game code.

That said, when I finally ran the complete test suite and saw all 19 tests pass, it really felt like a huge accomplishment. I'm definitely more confident about the code quality than I would be if I'd just written it normally and tested manually.

The GitHub repository with all my code and tests is at: **https://github.com/EchoKe26/hangman-tdd-assignment**