**CHARLES DARWIN UNIVERSITY SYDNEY CAMPUS**

Haymarket, NSW, Australia


# Software Unit Testing

S225 SOFTWARE ENGINEERING: PROCESS AND TOOLS

**PRT582**

Submitted By:

Nikesh Shrestha [S394589]

Submitted To:

Dr. Charles Yeo

Dr. Muhammad Rana

## I. Introduction:

The objective of this project was to design and implement a command-line Hangman game in Python. The game allows users to guess randomly selected words or phrases, with difficulty levels ranging from basic to intermediate. It includes visual feedback through ASCII art, input validation, and win/loss detection.

The Python 'unittest' module was used for automated testing. It is a built-in framework that supports test case creation, assertions, and test discovery. Its integration with TDD practices makes it ideal for validating individual components of the game logic.

## II. Process:

**Test-Driven Development (TDD):**

→ The development followed a TDD approach:

1. **Write tests first** – Before implementing a function, its expected behavior was defined through test cases.
2. **Implement functionality** – Code was written to pass the tests.
3. **Refactor and optimize** – Once tests passed, the code was cleaned up for readability and performance.

This cycle ensured that each feature was thoroughly validated and reduced the likelihood of bugs.

**Automated Testing Implementation:**

Tests were written for:

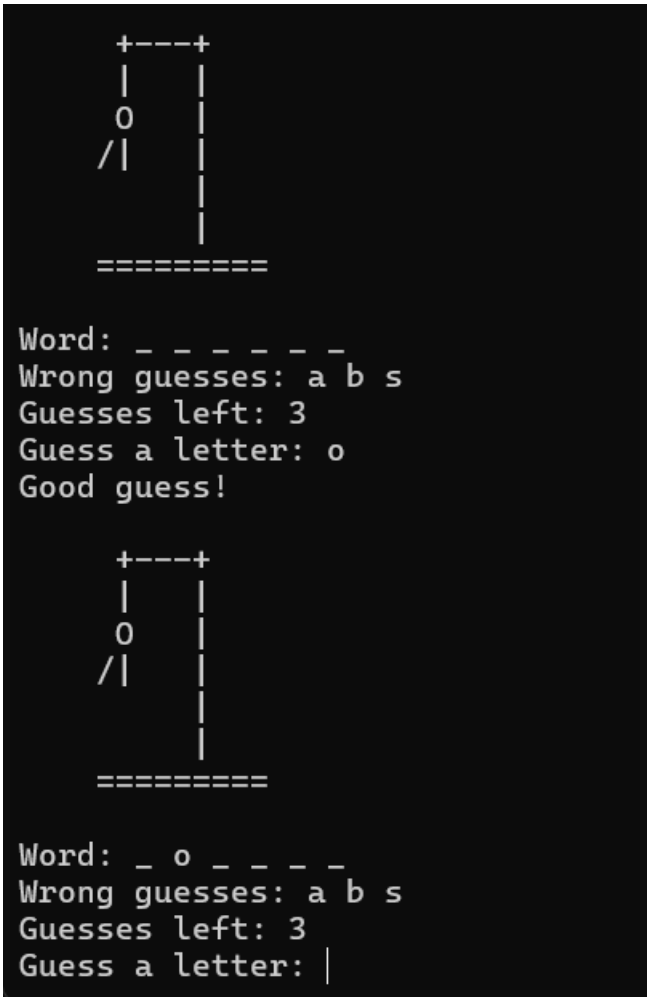
- random_basic() and random_intermediate() – Ensuring valid word/phrase selection.
- is_valid_word_or_phrase() – Verifying correct validation logic using both wordfreq and fallback heuristics.
- Game loop logic – Simulated guesses to test win/loss conditions and input handling.

❖ **Screenshots:**

```
# ASCII art stages for wrong guesses
HANGMAN_STAGES = [
    """
    +---+
    |   |
        |
        |
        |
        |
    =========
    """,
    """
    +---+
    |   |
    O   |
        |
        |
        |
    =========
    """,
    """
    +---+
    |   |
    O   |
    |   |
        |
        |
    =========
    """,
    """
    +---+
    |   |
    O   |
   /|   |
        |
```


+---+
|   |
O   |
/|\  |
|   |
|   |
=========
"""
'
"""
+---+
|   |
O   |
/|\  |
/   |
|   |
=========
"""
'
"""
+---+
|   |
O   |
/|\  |
/ \  |
|   |
=========
"""


C:\Users\newar\OneDrive\Desktop\hangman>python game.py
Choose difficulty-(B)asic or (I)ntermediate: b
+---+
|   |
|
|
|
|
=========
Word: _ _ _ _ _ _
Wrong guesses:
Guesses left: 6
Guess a letter: b
Wrong!
+---+
|   |
O   |
|
|
|
=========
Word: _ _ _ _ _ _
Wrong guesses: b
Guesses left: 5
Guess a letter:


+---+
|   |
O   |
/|   |
|
|
=========
Word: _ _ _ _ _ _
Wrong guesses: a b s
Guesses left: 3
Guess a letter: o
Good guess!
+---+
|   |
O   |
/|   |
|
|
=========
Word: _ o _ _ _ _
Wrong guesses: a b s
Guesses left: 3
Guess a letter:

## III. Conclusion:

**Lessons Learnt**

- **What went well:**

- TDD ensured reliable and maintainable code
- Python's simplicity accelerated development
- ASCII art added engaging visual feedback

- **Areas for improvement:**

- Could add GUI for better user experience
- Expand word pools for more variety
- Implement multiplayer or timed modes

## GitHub Repository link:

https://github.com/0Nikesh/Hangman