# Introduction

## Objectives and Requirements

The goal of this project was to create a Python program that calculates the Scrabble score of a word. The program includes the following features:

- It calculates the score of a word based on the value of each letter given in the assignment.
- The program considers both upper and lower-case letters to be the same.
- The player has to enter a word with a randomly chosen length.
- Players have 15 seconds to enter the word, and they get extra points for entering it faster.
- The word must be valid and checked using an online dictionary.
- The game runs for 10 rounds or until the player quits, and the total score is shown at the end.

## Unit Testing Tool

I used Python unittest tool to automatically test our code. The unittest is a testing tool that checks if the code works as expected. I used it to write tests first, then wrote the code to pass those tests. This method is called **Test Driven Development (TDD)**.

---

# Process

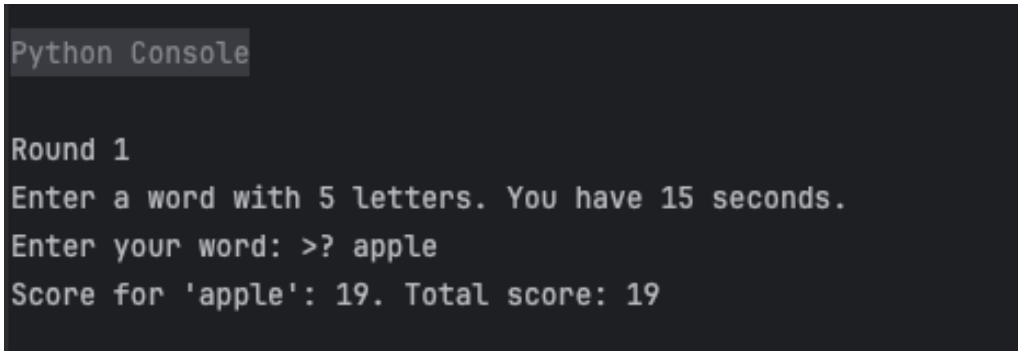
## How TDD and Unit Testing Were Used

1. **Writing Tests First**: I started by writing tests to check if the program calculates the Scrabble score correctly, validates the words, and handles user input.
2. **Test Scenarios**:
    - Checked if the program correctly calculates the score of words like "Assignment", "quiz",  “cabbage”, “apple” and “XylophOne”
    - Tested if the program can correctly identify real words and reject fake words.
    - Made sure the program gives feedback when the user enters something wrong, like numbers or a word of the wrong length.
3. **Writing Code After Tests**: Once the tests were ready, I wrote the program to pass all the tests. I kept running the tests to check that the code worked.

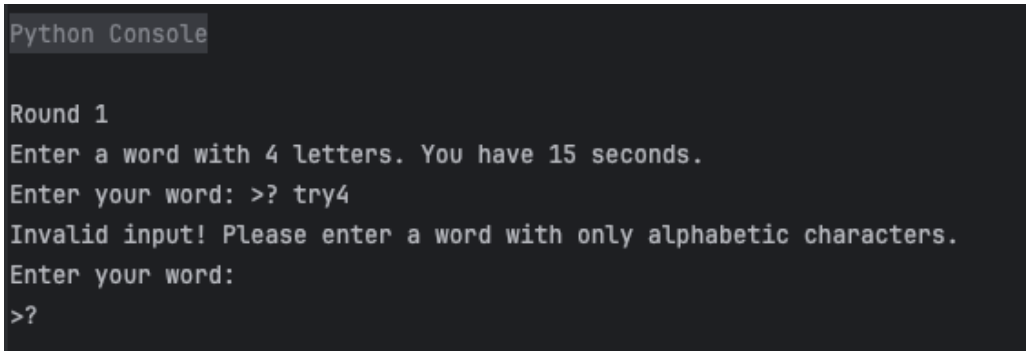
## Screenshots

Here's what some useful screenshots would show:

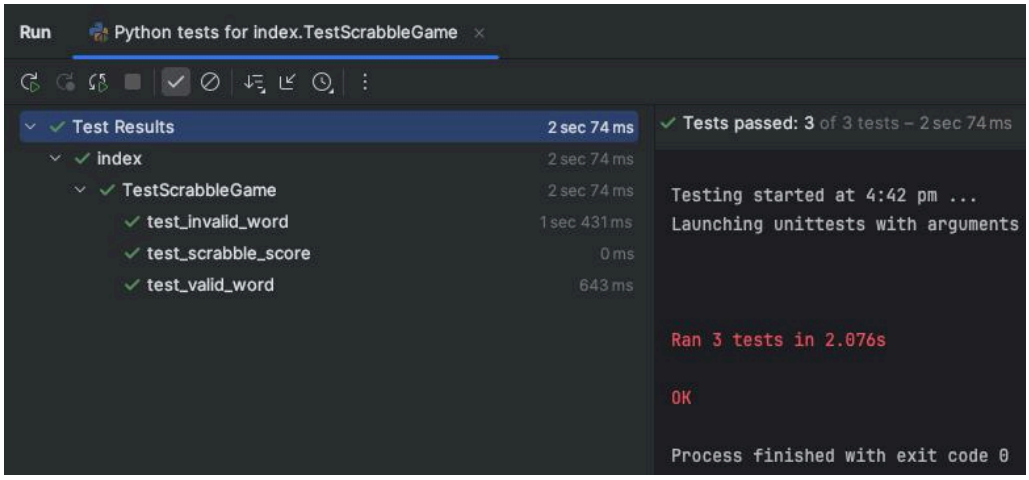
- **Screenshot 1**: The program correctly calculates and displays the score for a word like "apple".



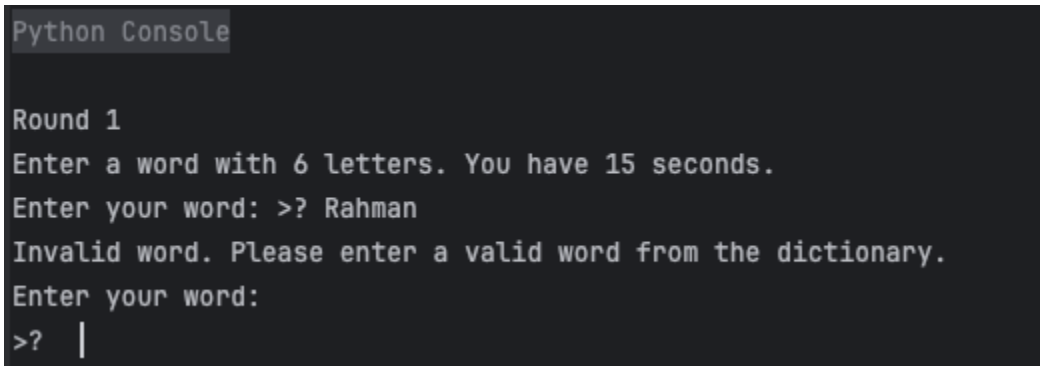

- **Screenshot 2**: The program gives feedback when the user enters invalid input, like a word with numbers.




- **Screenshot 3**: All unit tests passing, showing "OK" when the tests are run.



- **Screenshot 4**: The program rejects an invalid word not found in the dictionary.




---

# Conclusion

## What We Learned

- **What Worked Well**:
  - Writing tests first (TDD) made sure the code worked as planned from the start.
  - The unittest tool helped ensure the program could handle different situations, like invalid input and calculating scores correctly.
  - The program gave clear feedback to the user when they entered something wrong.
- **What Could Be Better**:
  - **Dictionary API Errors**: If there's a problem with the dictionary API (like no internet connection), the program doesn't handle it well. Adding error handling for this would improve the user experience.
  - **Timer System**: The current timer works, but it could be improved to stop exactly when the player finishes typing.

## Improvements

- **Faster Testing with API Mocking**: Right now, the tests rely on live dictionary API calls. This can slow things down. We could use a method called "mocking" to simulate API responses and make tests faster.
- **User Experience**: Adding a graphical interface (like buttons and text boxes) could make the game more fun and easier to play.

## Git Repository

https://github.com/Rahman421-star/ScrabbleScore