Fun With Python: Crafting Your Hangman Game

- .

What is Hangman?

Classic Word Game

Hangman is a word-guessing game where one player thinks of a word, and the other tries to guess it by suggesting letters. Incorrect guesses result in drawing parts of a hanged man.

Objective

The goal is to guess the word before the drawing is completed. It combines vocabulary and deductive reasoning skills.

Cultural Significance

Hangman is a popular pastime and an educational tool used in many English-speaking countries. It reinforces spelling and broadens vocabulary skills.

Agenda Overview

Project Setup

We'll set up the development environment and introduce basic Python concepts required for the project. Therefore, it is essential to have the right tools.

Core Logic

We will dive into the core code, including word selection, user input, and game state management to create an engaging user experience.

Enhancements

Enhance the game with features like difficulty levels, hints, and a graphical interface, making it more versatile.

Handling User Input

Prompt for Input

Prompt the user to enter a letter using the `input()` function. Provide clear instructions for the user to ensure correct input.

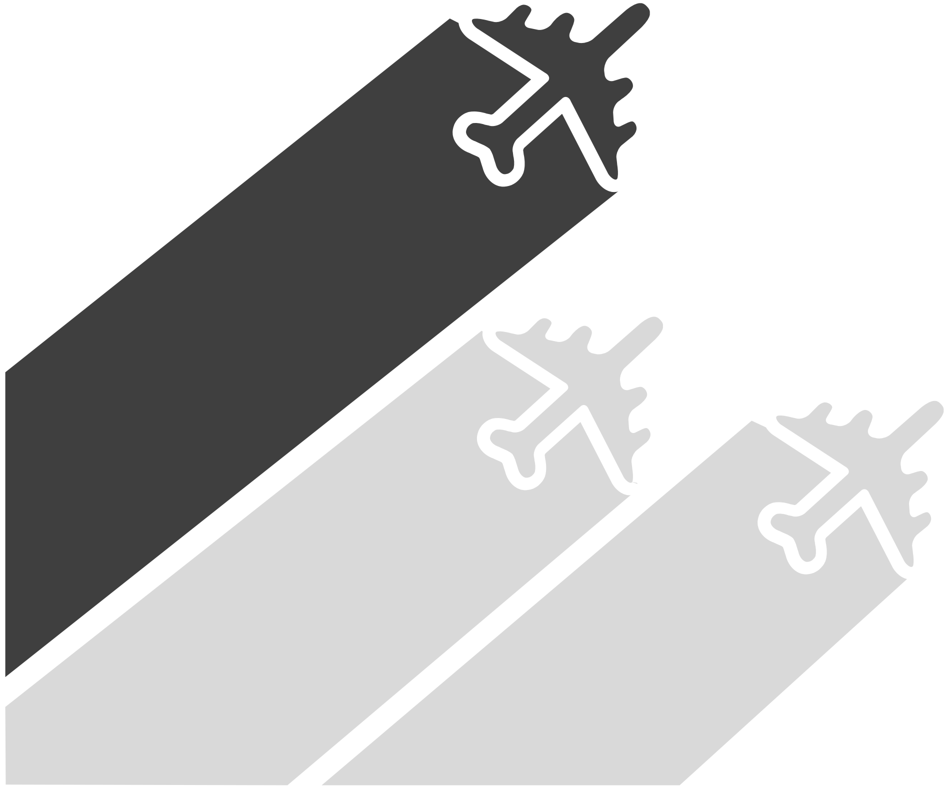
Input Validation

Validate the input to ensure it is a single letter. Reject any input with digits or other characters, maintaining input format.

Case Insensitivity

Convert the input to lowercase to make the game case-insensitive. This simplifies matching guessed letters with the secret word.

Game State Management



Displaying the Word

Show the user the current state of the word with correctly guessed letters and underscores for unguessed letters. It helps the user understand the game’s progress.

Checking the Guess

Check if the guessed letter is in the secret word. Use conditional statements to update shown word if the guess is correct, otherwise track incorrect ones.

Tracking Attempts

Keep track of the number of attempts the user has made. Display the remaining attempts to the user for better experience.

Word Selection

Selecting a word. Choosing a word randomly from a predefined list using the `random` module is the crucial first step. Keep various words to maintain user interest.

Game Display

Setting Up the Display. Setting up the game display involves showing underscores for each letter of the chosen word and an empty list for guessed letters. Display is critical for user engagement.

Setting Guesses

Setting the Number of Guesses. Set the number of allowed incorrect guesses, typically around 6-8, to give the user a reasonable chance to win. Adjust count for game balance.

Prompting User Guess

Prompt the user to guess a letter and validate the input. Handle cases where the input is not a single letter or has already been guessed.

Checking Letter Validity

Checking the Guess. Check if the guessed letter is in the word. If so, update the displayed word. Track the number of incorrect guesses to control the number of permissible tries.

Ending the Game

Win Condition

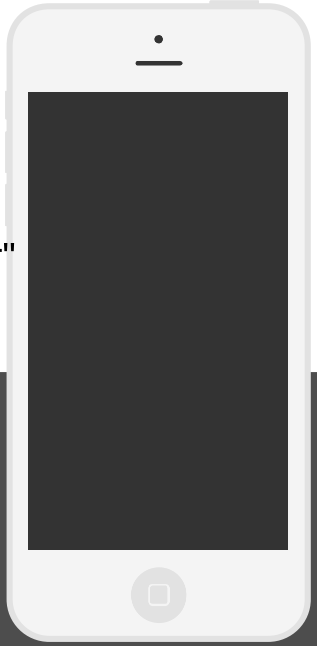
The game ends if the user correctly guesses the word.

Display a congratulatory message to the user.

Lose Condition

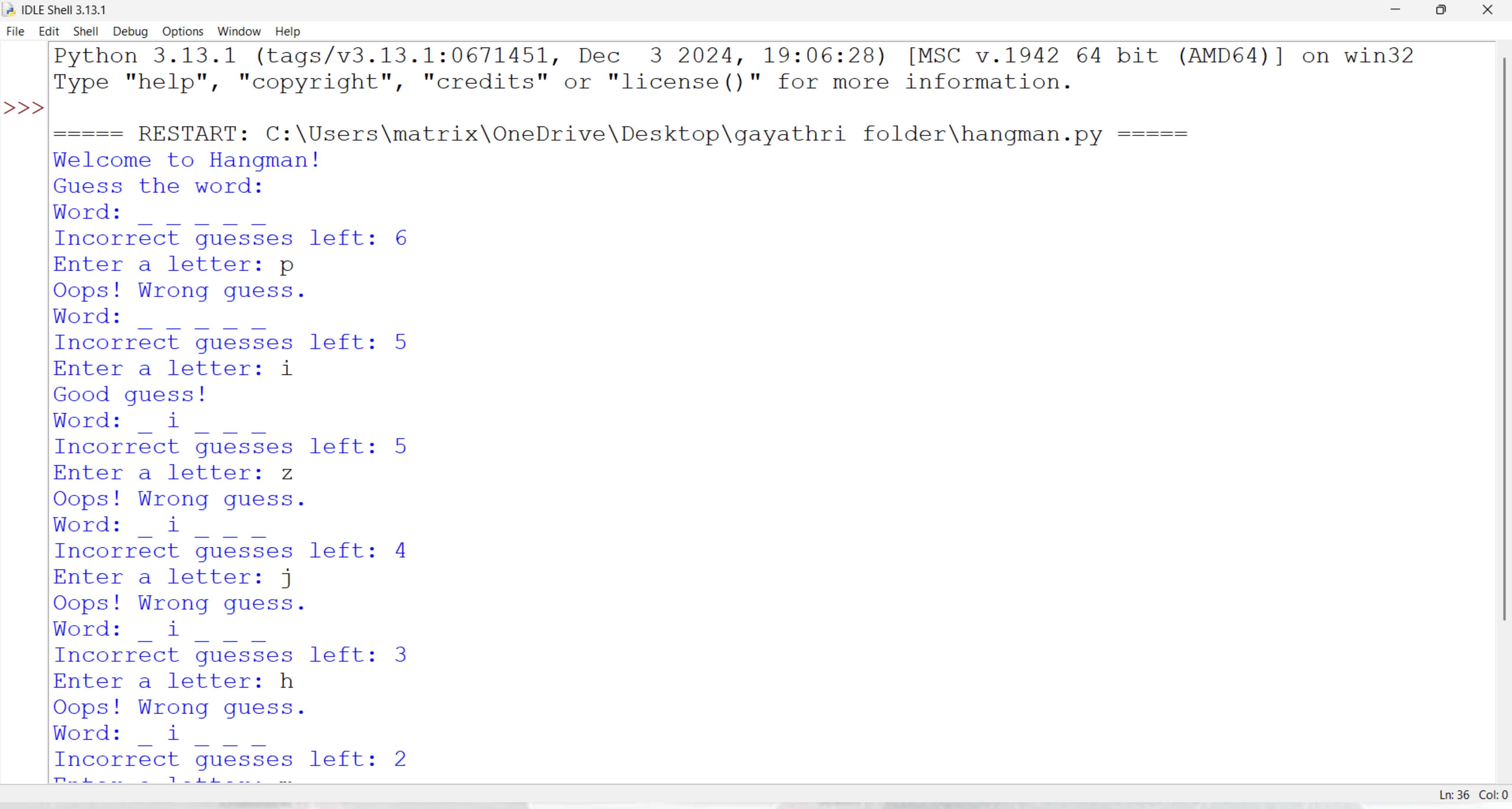
The game ends if the user runs out of guesses.

Reveal the correct word and provide a "game over" message.

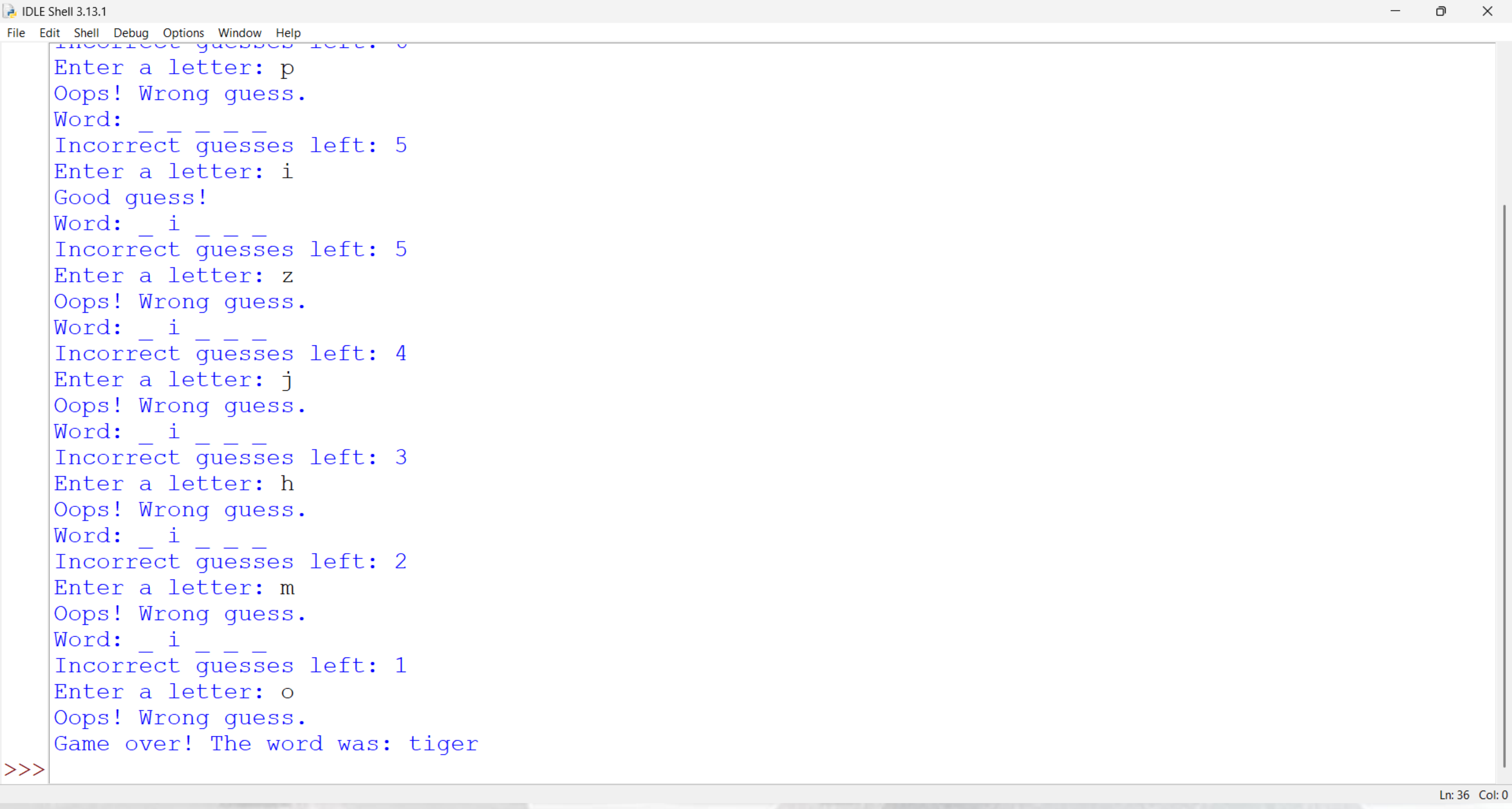


Playing Again

Ask the user if they want to play again. Restart the game if the user selects yes, which allows for an unlimited gaming experience.



```
Python 3.13.1 (tags/v3.13.1:0671451, Dec  3 2024, 19:06:28) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\matrix\OneDrive\Desktop\gayathri folder\hangman.py =====
Welcome to Hangman!
Guess the word:
Word: _ _ _ _ _
Incorrect guesses left: 6
Enter a letter: p
Oops! Wrong guess.
Word: _ _ _ _ _
Incorrect guesses left: 5
Enter a letter: i
Good guess!
Word: _ i _ _ _
Incorrect guesses left: 5
Enter a letter: z
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 4
Enter a letter: j
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 3
Enter a letter: h
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 2
```



```
Incorrect guesses left: 6
Enter a letter: p
Oops! Wrong guess.
Word: _ _ _ _ _
Incorrect guesses left: 5
Enter a letter: i
Good guess!
Word: _ i _ _ _
Incorrect guesses left: 5
Enter a letter: z
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 4
Enter a letter: j
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 3
Enter a letter: h
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 2
Enter a letter: m
Oops! Wrong guess.
Word: _ i _ _ _
Incorrect guesses left: 1
Enter a letter: o
Oops! Wrong guess.
Game over! The word was: tiger
>>>
```

Thank you

- .