MYSTERY GAME

Overview

The Mystery Solving Game is an interactive text-based adventure game built using Python. The game immerses players in a suspenseful journey through a deserted mansion where they must solve puzzles, uncover hidden clues, and ultimately find the missing artifact. This project demonstrates my skills in Python programming, problem-solving, and creating engaging user experiences.

Project Structure

The game is structured into various functions, each representing different parts of the mansion and the player's interactions. Here's a breakdown of the main components:

1. **Introduction**:
   - `intro()`: Welcomes the player and sets the stage for the adventure.
2. **Exploration**:
   - `explore_mansion()`: Allows the player to choose between different rooms (library, dining hall, conservatory) to explore.
3. **Library**:
   - `library()`: Players can investigate a desk or bookshelf to find clues.
   - `puzzle()`: A riddle-solving challenge where players must arrange symbols in the correct order to progress.
4. **Dining Hall**:
   - `dining_hall()`: Players interact with a butler or examine a suspicious portrait.
   - `numeric_lock()`: A numeric code puzzle to unlock a safe.
5. **Conservatory**:
   - `conservatory()`: Players must find a way to escape a gas-filled room by inspecting plants, statues, or windows.
6. **Secret Passage**:
   - `secret_passage()`: Leads to the final discovery of the missing artifact.
7. **Victory and Game Over**:
   - `victory()`: Celebrates the player's success in solving the mystery.
   - `game_over()`: Provides an option to restart the game if the player fails.

Key Features

- **Text-Based Interaction**: The game uses a function, `print_slow()`, to print text slowly, enhancing the immersive experience.
- **Decision Making**: Players make choices that affect the outcome of the game, providing a dynamic gameplay experience.
- **Puzzle Solving**: The game includes various puzzles, such as a riddle and a numeric lock, which require logical thinking and creativity.

- **Replayability**: If players fail, they can choose to play again, making the game replayable and engaging.

Code Highlights

- **Functionality**: The game is modular, with each part of the mansion and each interaction implemented in separate functions, promoting code readability and maintainability.
- **Random Elements**: The numeric lock puzzle uses a randomly generated code, adding an element of unpredictability to the game.

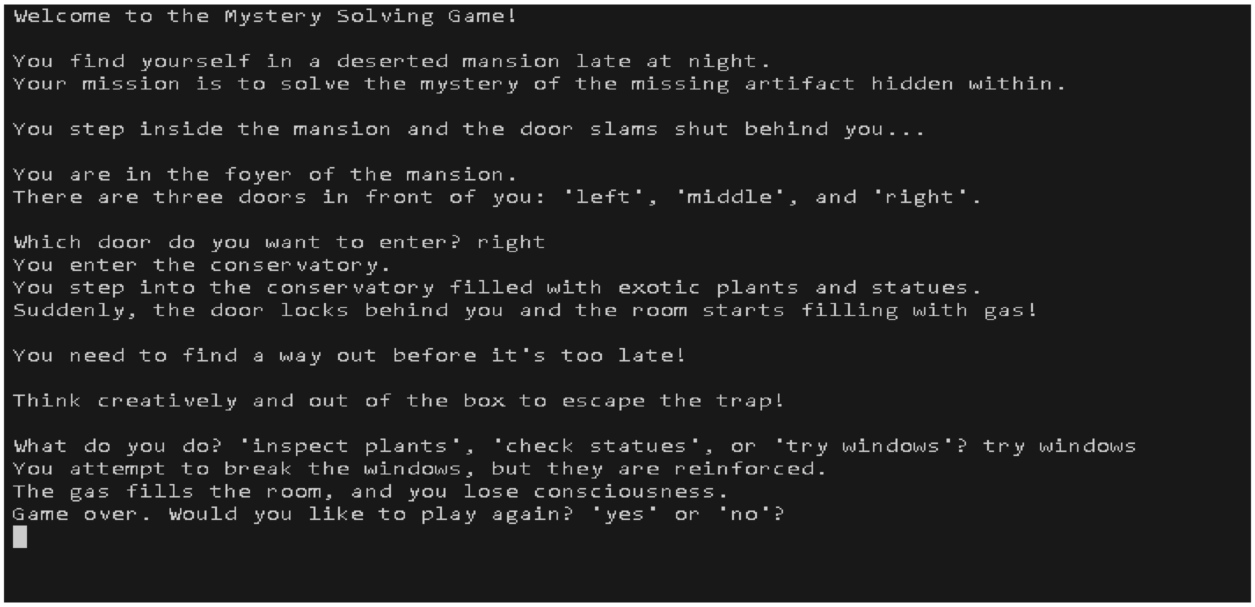
Learning Outcomes

- **Python Programming**: Developed a deeper understanding of Python functions, control structures, and user input handling.
- **Problem-Solving**: Enhanced my ability to design and implement logical puzzles and interactive scenarios.
- **User Experience**: Focused on creating an engaging and immersive experience for players through thoughtful text presentation and decision-making elements.

Conclusion

The Mystery Solving Game project showcases my ability to create interactive and engaging applications using Python. By combining storytelling, puzzle-solving, and user interaction, I have developed a project that not only highlights my technical skills but also my creativity and attention to detail.

Implementation



```
Welcome to the Mystery Solving Game!

You find yourself in a deserted mansion late at night.
Your mission is to solve the mystery of the missing artifact hidden within.

You step inside the mansion and the door slams shut behind you...

You are in the foyer of the mansion.
There are three doors in front of you: 'left', 'middle', and 'right'.

Which door do you want to enter? right
You enter the conservatory.
You step into the conservatory filled with exotic plants and statues.
Suddenly, the door locks behind you and the room starts filling with gas!

You need to find a way out before it's too late!

Think creatively and out of the box to escape the trap!

What do you do? 'inspect plants', 'check statues', or 'try windows'? try windows
You attempt to break the windows, but they are reinforced.
The gas fills the room, and you lose consciousness.
Game over. Would you like to play again? 'yes' or 'no'?
```