Rock Paper Scissors Game Documentation

Project Team

1. Name: K Laxmi Prasanna
   Roll No.: 2203A51423

2. Name: A Jasper Enoch
   Roll No.: 2203A51535

Institute Details

SR University

School of Computer Science & Artificial Intelligence

Training Information:

Training Company: L0gicWhi1e

Trainer: K V Srinivas Sir

Overview

This document provides comprehensive documentation for a command-line Rock Paper Scissors game implemented in Python. The game offers a complete CRUD (Create, Read, Update, Delete) functionality for managing game rounds and maintains player scores.

1. Class Structure

The game is implemented through a single class `RockPaperScissors` that maintains the game state and handles all game operations. The class uses the following attributes:

- `rounds`: List storing dictionaries of round information
- `p_score`: Integer tracking player's score
- `c_score`: Integer tracking computer's score

2. Game Features

- Play rounds against computer opponent
- View game history
- Track scores
- Update previous rounds
- Delete specific rounds
- Reset entire game
- Input validation and error handling
- Score management system

3. Methods Documentation

Constructor

def __init__(self):

Initializes a new game with empty rounds list and zero scores.

add_round()

def add_round(self, player_choice, computer_choice, result):

Creates a new round entry with provided choices and result.

Parameters:

- `player_choice`: String - Player's move choice
- `computer_choice`: String - Computer's move choice
- `result`: String - Outcome of the round

display_rounds()

def display_rounds(self):

Shows all played rounds with their details.

display_scores()

def display_scores(self):

Displays current scores for both player and computer.

determine_winner()

def determine_winner(self, player_choice, computer_choice):

Calculates the winner based on game rules.

Parameters:

- `player_choice`: String - Player's move
- `computer_choice`: String - Computer's move

Returns:

- String: "Player Wins", "Computer Wins", or "Tie"

update_round()

def update_round(self, round_number):

Updates an existing round with new choices and recalculates scores.

Parameters:

`round_number`: Integer - Round to update

delete_round()

def delete_round(self, round_number):

Removes a specific round and adjusts scores accordingly.

Parameters:

`round_number`: Integer - Round to delete

reset_game()

def reset_game(self):

Clears all rounds and resets scores to zero.

play_round()

def play_round(self):

Executes a single round of the game.

4. Game Flow

1. Game starts with an empty round history and zero scores
2. Player selects from menu options:
    - Play Round (1)
    - Display Rounds (2)
    - Display Scores (3)
    - Update Round (4)
    - Delete Round (5)
    - Reset Game (6)
    - Exit (7)
3. For each round:
    - Player makes choice (1-3)
    - Computer randomly selects
    - Winner is determined
    - Scores are updated
    - Round is recorded

5. Error Handling

The game implements error handling for:

- Invalid menu choices
- Invalid player moves
- Invalid round numbers for updates/deletions
- Type conversion errors
- Missing rounds during updates

6. Usage Examples

Starting a New Game

```
game = RockPaperScissors()
```

Playing a Round

```
# Select option 1 from menu
# Enter 1 for Rock, 2 for Paper, or 3 for Scissors
game.play_round()
```

Updating a Round

```
# Select option 4 from menu
# Enter round number
# Enter new choice
game.update_round(round_number)
```

Viewing Game History

```
# Select option 2 from menu
game.display_rounds()
```

Best Practices

1. Always validate round numbers before updates or deletions
2. Check scores after updating or deleting rounds
3. Use the reset function to start fresh rather than creating a new instance
4. Regularly display rounds to track game progress

Maintenance Notes

When modifying the code:

- Maintain score consistency after updates/deletions
- Preserve round numbering sequence
- Validate all user inputs
- Handle edge cases for empty game states
- Consider adding data persistence in future versions

Future Enhancements

Potential improvements could include:

1. Save/load game state
2. Multiple player support
3. Statistics tracking
4. GUI interface
5. Network play capability
6. Tournament mode
7. Achievement system