Roulette Simulator Appendix

By: Jackson Gamel

Project Structure

```
roulette_simulator/
├── analysis.py              # Summary analysis of results
├── main.py                  # Entry point for the CLI menu and auto mode
├── roulette.py              # Core roulette game logic
├── strategies.py            # Betting strategy implementations
├── compare_results.csv      # (Generated) Spin-by-spin bankroll for compare mode
├── strategy_comparison.csv  # (Generated) Average bankrolls from auto trials
├── strategy_comparison.png  # (Generated) Strategy performance graph
└── README.md / APPENDIX.md  # Documentation
```

Running the Simulator

Requirements

- Python 3.7 or higher
- Install matplotlib (for graphing results):

For interactive mode : run this in the command prompt : python3 main.py

Then follow the menu prompts:

1. Choose a strategy (Martingale, Flat, etc.).
2. Enter starting bankroll.

3. Set your base bet.

4. Choose number of spins.

Detailed Strategy Explanations

1. Martingale Strategy

- Concept: Double your bet after every loss to recover previous losses and gain a profit equal to the original bet.

- Mechanism:

    1. Start with base bet (e.g., $1).

    2. If you win, bet $1 again.

    3. If you lose, double the bet to $2.

    4. Lose again? Bet $4, and so on.

    5. After winning, bet resets back to $1.

- Pros: High recovery potential if you can survive a losing streak.

- Cons: High risk of bankroll wipeout due to exponential bet growth.

2. Flat Betting Strategy

- Concept: Bet the same amount every spin.

- Mechanism:

1. Set base bet (e.g., $5).
2. Bet $5 every round regardless of win or loss.

- Pros: Simple, low-risk, easy to track.
- Cons: Limited potential for fast growth.

3. Reverse Martingale Strategy

- Concept: Increase your bet after a win, and reset after a loss.
- Mechanism:
    1. Start with base bet (e.g., $1).
    2. Win? Double your bet to $2.
    3. Win again? Bet $4, etc.
    4. As soon as you lose, reset to base bet size ($1).
- Pros: Capitalizes on winning streaks.
- Cons: One loss can erase a streak's gains.

4. D’Alembert Strategy

- Concept: Increase your bet by one unit after a loss, and decrease by one unit after a win.
- Mechanism:
    1. Start with base bet (e.g., $1).
    2. Lose? Bet $2 next.

3. Win? Drop back to $1.
4. Keeps increasing and decreasing by one unit.

- Pros: Slower bankroll swings than Martingale.
- Cons: Still risky in long losing streaks.

5. Fibonacci Strategy

- Concept: Use the Fibonacci sequence to determine bet size after losses.
- Mechanism:
    1. Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, ...
    2. After each loss, move one step forward in the sequence.
    3. After a win, move two steps back.
    4. Reset to start after big wins.
- Pros: Slower loss recovery than Martingale but less aggressive.
- Cons: Still requires a large bankroll for long loss streaks.

Mode 2: Compare All Strategies (Interactive)

From the main menu, choose:

markdown
CopyEdit
6. Compare Strategies

Then provide:

- Bankroll (e.g., 1000)
- Bet amount (e.g., 10)
- Spins (e.g., 1000)

📁 Files Created:

- compare_results.csv: Bankroll for each strategy per spin
- strategy_comparison.png: Graph of bankroll trends

Mode 3: Auto Mode (Batch Simulations)

Run directly from the terminal:

bash
CopyEdit
python3 main.py auto 100

Where 100 is the number of trials (sets of 100 spins) to run for each strategy.

- The simulator runs all 5 strategies over 100 spins, 100 times each.
- It averages the bankroll per spin across all trials.
- This simulates statistical behavior over the long term.

📁 Files Created:

- strategy_comparison.csv: Average bankrolls

- strategy_comparison.png: Graph comparing strategies

Interpreting the Graph

- X-axis: Number of spins (0 to N)
- Y-axis: Average bankroll
- Each line = One strategy
- Use this to observe long-term patterns:
    - Does Martingale crash?
    - Is Flat Betting more stable?
    - Which strategy ends with the highest bankroll?

Troubleshooting

Problem	Solution
Error: "matplotlib not found"	Run pip install matplotlib
Nothing happens on main.py	Check you're in the correct folder and using python3
Long runtime on auto mode	Try fewer trials (e.g., 50 instead of 1000)