

Project Report: Rock Paper Scissors CLI Game

Submitted by: Divyansh arora

Reg No:25BAI11338

Date: November 25, 2025

Subject: Introduction to Python Programming

1. Abstract

This project involves the development of a console-based (CLI) version of the classic hand game "Rock, Paper, Scissors" using the Python programming language. The primary objective was to create a robust, continuous game loop that features intelligent input handling and an efficient, mathematical approach to determining the winner, ensuring a seamless user experience without runtime errors.

2. Introduction

"Rock, Paper, Scissors" is a zero-sum game usually played between two people. In this computerized version, the user plays against an automated opponent (the Computer). The game relies on random number generation to ensure fairness for the computer's moves. This project demonstrates modular programming concepts, error handling, and algorithmic logic implementation in Python.

3. Problem Definition

The core requirement was to build a system that meets the following criteria:

- **Continuous Gameplay:** The program must not terminate after one round; it should continue until the user explicitly decides to quit.
- **Input Flexibility:** The system must handle various user inputs, including full words (e.g., "Rock") and abbreviations (e.g., "r"), while being case-insensitive.
- **Error Tolerance:** Invalid inputs (e.g., typos or unrelated text) must be caught and handled gracefully without crashing the program.
- **Game Logic:** The standard rules must apply: Rock beats Scissors, Scissors beats Paper, and Paper beats Rock.

4. System Design & Methodology

4.1. Input Handling Algorithm

To ensure a user-friendly experience, a "Smart Input" system was designed. Instead of requiring exact string matches, the system analyzes the starting character of the input using string manipulation methods (.startswith()).

- Input "r", "roc", "Rock" \rightarrow Detected as **Rock**.
- Input "s", "sci", "Scissors" \rightarrow Detected as **Scissors**.
- Input "p", "pap", "Paper" \rightarrow Detected as **Paper**.

4.2. Mathematical Win Logic (Integer Mapping)

Instead of using complex nested if-else statements comparing string values, the project utilizes integer mapping to represent the game states. This reduces computational complexity and improves code readability.

Mapping Table: | Move | Integer Value | :--- | :--- | | Rock | -1 | | Paper | 0 | | Scissors | 1 |

Logic Implementation: The winner is determined by comparing these integer values against specific win conditions:

1. **Tie:** User == Computer
2. **User Win Conditions:**
 - Rock (-1) vs Scissors (1)
 - Scissors (1) vs Paper (0)
 - Paper (0) vs Rock (-1)
3. **Computer Win:** Default else case if the above are not met.

5. Implementation Details

- **Programming Language:** Python 3.x
- **Key Libraries:** random (for generating computer moves).
- **Architecture:** Procedural/Modular.
 - inputHandler(inpt): Normalizes and validates user input.
 - computers_choice(): Randomly selects -1, 0, or 1.
 - win_dec(user, computer): Compares values to declare the winner.

- Main Loop: Orchestrates the game flow and handles the exit condition.

6. Testing and Results

The application was tested against various scenarios to ensure stability.

Test Case 1: Standard Gameplay

Input: rock **Computer:** scissors **Result:** User Wins! (Correct)

Test Case 2: Flexible Input

Input: p (for Paper) **Computer:** rock **Result:** User Wins! (Correct)

Test Case 3: Invalid Input

Input: banana **Result:** System prompts "Invalid input" and asks again. (Correct - No Crash)

Test Case 4: Exit

Input: 0 **Result:** Game loop terminates with a goodbye message.

7. Conclusion

The "Rock Paper Scissors" project successfully meets all functional requirements. By implementing modular functions and integer-based logic, the code is both efficient and easy to maintain. The addition of smart input handling significantly improves the user experience compared to strict string matching.

8. Future Scope

Future iterations of this project could include:

- **Score Tracking:** Implementing a counter to track wins/losses across the session.
- **GUI:** Developing a Graphical User Interface using Tkinter or PyQt.
- **Multiplayer:** Adding socket programming to allow two humans to play over a network.