PROJECT TITLE: INTELLIGENT ROCK-PAPER-SCISSORS (FUZZY LOGIC EDITION)

Overview of the Project

This project revolves to creating a "Smart" Command Line Interface (CLI) game using Python, specifically designed to eliminate the rigidity of standard text-based games. Unlike traditional programs that require perfect spelling, this project utilizes automated fuzzy logic to interpret user intent without manual intervention for error correction. It is a logic-driven project based on the core programming concepts and algorithmic thinking learned in the VITYARTHI COURSE.

Problem Statement

In the world of beginner programming and CLI applications, user experience often takes a back seat. Standard games suffer from strict syntax requirements where a single typo (like "rck" instead of "rock") causes the program to reject the input or crash.

Major problems identified:

Rigid Input Sensitivity: Users are punished for typing too fast or making minor spelling errors.

Visual Boredom: Standard terminals are monochromatic, making it difficult to track the "momentum" of the game (winning vs. losing) at a glance.

Lack of Feedback: Users must manually calculate the score difference to know who is dominating the match.

Runtime Errors: Poorly handled inputs often lead to immediate program termination.

Solution (Project Goal)

The goal of this project is to build a resilient, "Intelligent" Rock-Paper-Scissors game that prioritizes user intent over syntactical precision. By implementing a probabilistic character-matching algorithm, the system "guesses" what the user meant to type. Furthermore, to solve the issue of visual boredom, the project implements a dynamic colour-grading engine that visually represents the game state—shifting the environment to "Cool Blue" when the user wins, and "Alarm Red" when the computer wins.

Features of the Project

Fuzzy Input Handler: A custom algorithm that deconstructs words into character lists to find the highest probability match, auto-correcting inputs like "ppr" to "Paper".

Dynamic Colour Grading: The interface uses colorama to shift the text colour spectrum based on the score differential (User Score minus Computer Score).

Robust Error Handling: The game catches completely invalid inputs gracefully without breaking the game loop.

Live Score Tracking: Global variables track the state continuously, providing real-time feedback.

Tools Used

Language: Python 3.x

Development Environment: VS Code

Libraries:

Colorama: For cross-platform terminal color rendering.

Random: To generate unpredictable computer moves.

Time: To create dramatic pauses and improve UI flow.

Logic: String manipulation, List comparisons, Exception handling (try/except).

Steps to Install and Run

Download the file: Save the attached RPS.py file to your local machine.

Environment Setup: Ensure you have Python installed. You can run this in a standard terminal or Jupyter Notebook.

Install Dependencies: This project uses an external library for colors. Open your terminal and run:

pip install colorama

Run the Game: Execute the file:

python RPS.py

Play: Enter your choice. Try typing variations like "roock" or "sissors" to test the auto-correction features.

Scope of the Project

Algorithm Design: This project demonstrates how to build probability-based logic rather than simple equality checks (if x == y), making it a strong case study for algorithm design.

User Experience (UX): It explores how visual cues (colour) can gamify a simple text interface, making it applicable to other CLI tools that need status indicators (e.g., server health monitors).

Error Management: It covers the scope of handling unexpected user behaviour without crashing, a critical skill in software development.

Target Users

VITYARTHI Course Students: To reinforce concepts of basic game logic, state management, and user input handling, serving as a practical application of the course syllabus.

Beginner Python Developers: To study clean function separation (e.g., separating the UI logic from the Game logic) and techniques for enhancing terminal output.

Individual Users: For quick, casual play against the computer in a low-friction environment that doesn't penalize typing mistakes.