VITYARTHI COURSE PYTHON PROGRAMMING

Intelligent Rock-Paper-Scissors

Fuzzy Logic & Dynamic UI

Govind Parashuram Jadhav

Registration Number: 25BAI11219

Introduction

This project reimagine the classic CLI game by addressing two major flaws in standard implementations:

- **Rigid User Input**
- **Lack of Visual Engagement**

> **Smart Input:** Uses fuzzy logic to understand typos (e.g., "rck" = Rock).
>
> **Atmospheric UI:** The terminal changes color based on the game state.

Tech Stack: Python 3.x, Colorama, Random.

```
        # A smarter way to play def game_loop():
while True: input = get_smart_input()
update_color_grade()
```

The Problem

1. Rigid Input Sensitivity

In standard games, typing "scissor" instead of "scissors" or accidentally entering "rock" crashes the program or forces a restart. This breaks the user's immersion and flow.

2. Visual Boredom

Standard CLI games are monochromatic. It is difficult to track who is winning at a glance without reading every line of text mentally.

The Solution

+ Fuzzy Matching

The system deconstructs input into character lists and calculates a "match score" against target words.

! Winning Spectrum

A cool **Blue/Magenta** spectrum indicates user dominance and winning streaks.

! Losing Spectrum

A hot **Red/Yellow** spectrum indicates the computer is winning.

Functional Modules

⌨ Input Processor

Deconstructs raw strings. Scores character matches. Auto-corrects typos based on highest probability.

⚙ Game Engine

Randomizes computer moves. Determines win/loss logic. Updates global score counters.

🎨 UI Renderer

Calculates score differential (U-C). Applies colorama styles based on the "Blue" or "Red" zone.

Architecture: Fuzzy Input Handler

The core innovation is the `inputHandler(inpt)` function. It doesn't use simple equality checks.

Conceptual Logic Flow

`User Input: "pap"` `Target List: ["rock", "paper", "scissors"]`

1. Deconstruct "pap" → `['p', 'a', 'p']`
2. Match against "paper" → Found 'p', 'a', 'p' (High Score)
3. Match against "rock" → No match (Zero Score)
4. Result → Returns "paper"

Architecture: Dynamic Color Grading

The Formula

Diff = User_Score - Comp_Score

The visual atmosphere is strictly controlled by the `col()` function, which reads the global variables.

Diff > 0 (Winning)
Magenta (+1) ← Blue (+3)

Diff = 0: White (Neutral)

Diff < 0 (Losing)
Yellow (-1) ← Red (-3)

Code: Input Logic

```
    def inputHandler(inpt): inptList = list(inpt) passOutput = "rock paper scissors" r,p,s = passOutput.split() rval, pval, sval = 0, 0, 0 # Iterate through target words to find matches for k in [list(r), list(p), list(s)]: for l in inptList: if l in k: if k == list(r): rval += 1 elif k == list(p): pval += 1 elif k == list(s): sval += 1 w = max(rval, pval, sval) # Return word with highest match score
```

Results: Execution Snapshots

Dynamic Color Grading (Losing)

```
Computer Wins!

Enter '0' to exit

Computer: 9

User: 7


Your choice: ro

Computer's choice: scissors
```

Input Handling

```
Enter '0' to exit

Computer: 5

User: 5


Your choice: 5

Invalid input

Enter '0' to exit
```

Conclusion

Project Success

This project successfully demonstrates how basic Python logic can be elevated with user-centric design. By adding fuzzy input handling and dynamic color feedback, a simple game transforms into an engaging, responsive application.

Thank You

Govind Parashuram Jadhav

25BAI11219