Iterables & Loops p.IV
Complex Algorithms

Exercises

Rock, Paper, Scissors

Create a simple one player Rock, Paper, Scissors game in Python. The game is a best-of-three match. The player is playing against the computer. After each match, display the scores and allow players the option to play again.

Remember the basic rules:

- Rock crushes Scissors.
- Scissors cut Paper.
- Paper covers Rock.

Instructions

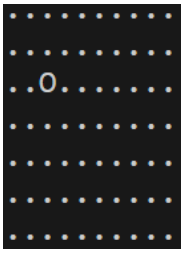
1. Initialize Variables
    a. Track scores for each player.
    b. Track the number of games.
    c. Define a list of allowed choices.
2. Obtain input for the player and the computer.
    a. Make sure that the player uses the correct move !
    b. For the computer, you can use the random library to generate a random choice:

```
import random # at the top of your python program
computer_choice = random.choice(["rock", "paper", "scissors"])
```

3. Use a loop to manage the flow of the game. For each turn in the loop:
    a. Capture each player's choice. Validate the input.
    b. Determine the round's winner.
        i. If there's a draw, restart the round.
        ii. If not, update and display the current scores.
    c. If a player reaches a score of 2, announce them as the match winner.
4. Once a match concludes, ask players if they'd like another match.
    a. If yes, reset scores and continue.
    b. If not, exit the game.
5. Display a closing message once the players decide not to continue.

Ball Movement

Create a simple console-based program in Python where a "ball" can be moved around the screen using keyboard commands.



Instructions

1. Initialize Variables
    a. Define your "scene" size. This could be a grid of 10x10, for example.
    b. Decide on a starting position for your ball, such as the center.
    c. Decide on a representation for the ball (e.g., "O") and empty spaces (e.g., ".").
2. Use lists to represent the scene's state. For each position in the grid, store either the ball's representation or an empty space.
3. Capture Player Input
    a. Use commands like 'W' for up, 'A' for left, 'S' for down, 'D' for right.
    b. Add a quit command like 'Q'.
    c. Make sure to validate the input, ensuring it's one of the allowed commands.
4. Interpret the user's command to calculate the new position for the ball.
5. Make sure the new position is within the grid's boundaries.
6. Before updating the scene on the console, you'll need to clear the previous state. You can use the os module to help with this:

```
import os  # at the top of your python program
os.system('cls' if os.name == 'nt' else 'clear')
```

7. Print out the current state of your grid, showing the ball and empty spaces.
8. Continually:
    a. Capture the player's command.
    b. Update the ball's position.
    c. Clear the console.
    d. Render the updated scene.
    e. Exit the loop when the player decides to quit.
9. Display a message to thank the player for playing when they decide to quit.