Shooter Target Challenge

Challenge Preamble

This challenge is designed to help you practice Python fundamentals through a fun shooter-style simulation. You'll work with file handling, loops, conditionals, and basic geometry to calculate scores based on how close shots land to their targets. Each target has three scoring rings (bullseye, inner, outer), and every shot earns points depending on which ring it hits.

Part 1 – Target Detection and Scoring: Theory

In this program, you will process two text files: one containing the target information (**`rings.txt`**) and the other containing shot coordinates (**`shots.txt`**). Each line of **`rings.txt`** defines a target with its center coordinates and three radii (bullseye, inner, and outer). Each line of **`shots.txt`** provides the **`(x, y)`** coordinates of a fired shot.

The main task is to determine if each shot hits the **bullseye (10pts)**, **inner ring (5pts)**, **outer ring (3 pts)**, or **misses** completely (**0pts**). You'll achieve this by calculating the distance between the shot's position and the target's center using the **Pythagorean theorem (math.hypot)**. The smaller the distance, the better the shot!

Key Concepts:

- Reading structured text files.
- Parsing strings into numeric values for mathematical calculations.
- Using math to calculate distance.
- Employing match-case statements for score classification.
- Using dictionaries to count hit types and compute totals.

Part 2 – Code Walkthrough

Let's go over the important components of the program step-by-step:

1. Function: **`get_shoot_score(target: list[str], shot: list[str]) -> int:`**

This function receives two parameters: target (the target's data) and shot (the shot's coordinates). It extracts the target's position and ring radii, then computes the distance between the shot and the target center using the **`math.hypot()`** function. Depending on this distance, it returns a score: 10 for bullseye, 5 for inner, 3 for outer, and 0 for a miss.

2. Function: **`main()`**

The main function handles file reading and scoring logic. It reads both files, converts the data into lists, and iterates through every shot to determine the corresponding score using **`get_shoot_score()`**. It also maintains counters for each hit category (Bullseye, Inner, Outer) and prints logs at the end.

Step-by-step explanation:

1. Open the files **`rings.txt`** and **`shots.txt`** using **`open()`** within a context manager.
2. Split and convert each line into numeric values (**`floats`** or **`int`**) for calculations.
3. Initialize score counter.
4. Use nested loops: for each shot, check all targets to compute scores.
5. Use match-case to classify the score and increment the appropriate category.
6. Sum all scores and display the Final score.
7. If you succeed your final score would be **269 pts**

Part 3 – Bonus Challenge: `PyGame` Visualization

In the extended version, the program uses the **`PyGame`** library to visualize each shot and its corresponding target. Every second, a new target and shot appear on the screen, allowing you to watch the scoring sequence in real time.

`PyGame` is a multimedia library for Python that provides tools to build games and visual simulations. Key features used in this challenge include:

- **`pygame.display.set_mode()`** – Creates a display window.
- **`pygame.draw.circle()`** – Draws each target ring and shot on the screen.
- **`pygame.time.get_ticks()`** and **`clock.tick()`** – Control timing and frame rate.
- Event handling (**`pygame.event.get()`**) – Allows quitting or pausing the replay.
- A continuous game loop – Keeps updating and redrawing the screen until the user exits.

The **`PyGame`** version enhances your understanding of event loops, timing control, and coordinate mapping. Try modifying the update delay, ring colors, or add interactivity for extra fun!