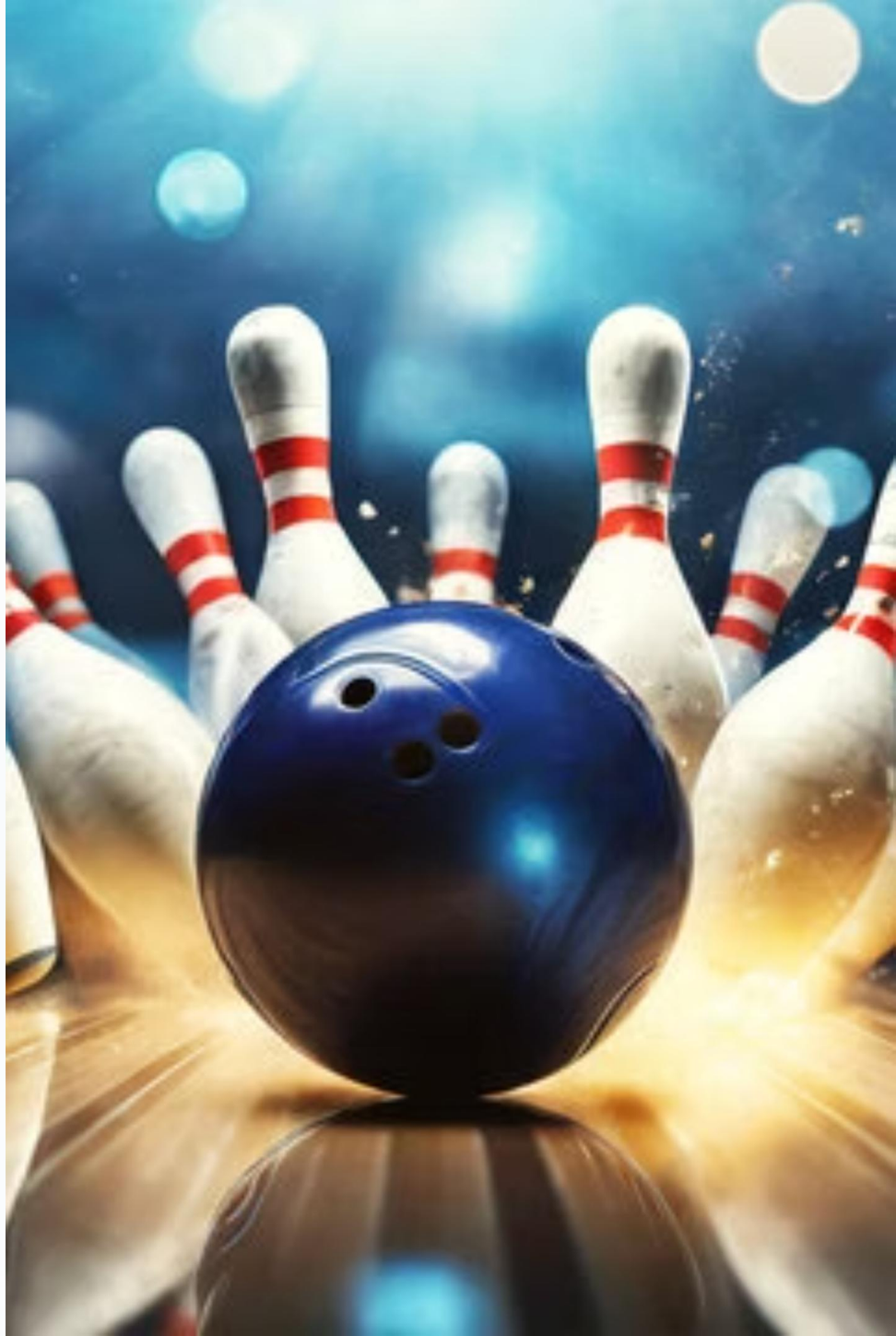
Python Module for Ten-Pin Bowling

Hridhya E K



The Challenge

Bowling Score Calculation

This Python module designed to accurately calculate the total score of a single ten-pin bowling game.

Complex Rules

Ten frames, strikes, spares, and open frames each have unique scoring implications.

Accurate Summation

The total game score is the sum of all 10 frame scores, requiring precise calculation.

Input Interpretation

Translating a string-based notation of rolls into numerical values for processing.

Scoring Rules Explained

The core of bowling score calculation lies in understanding how each frame type is scored.

Open Frame

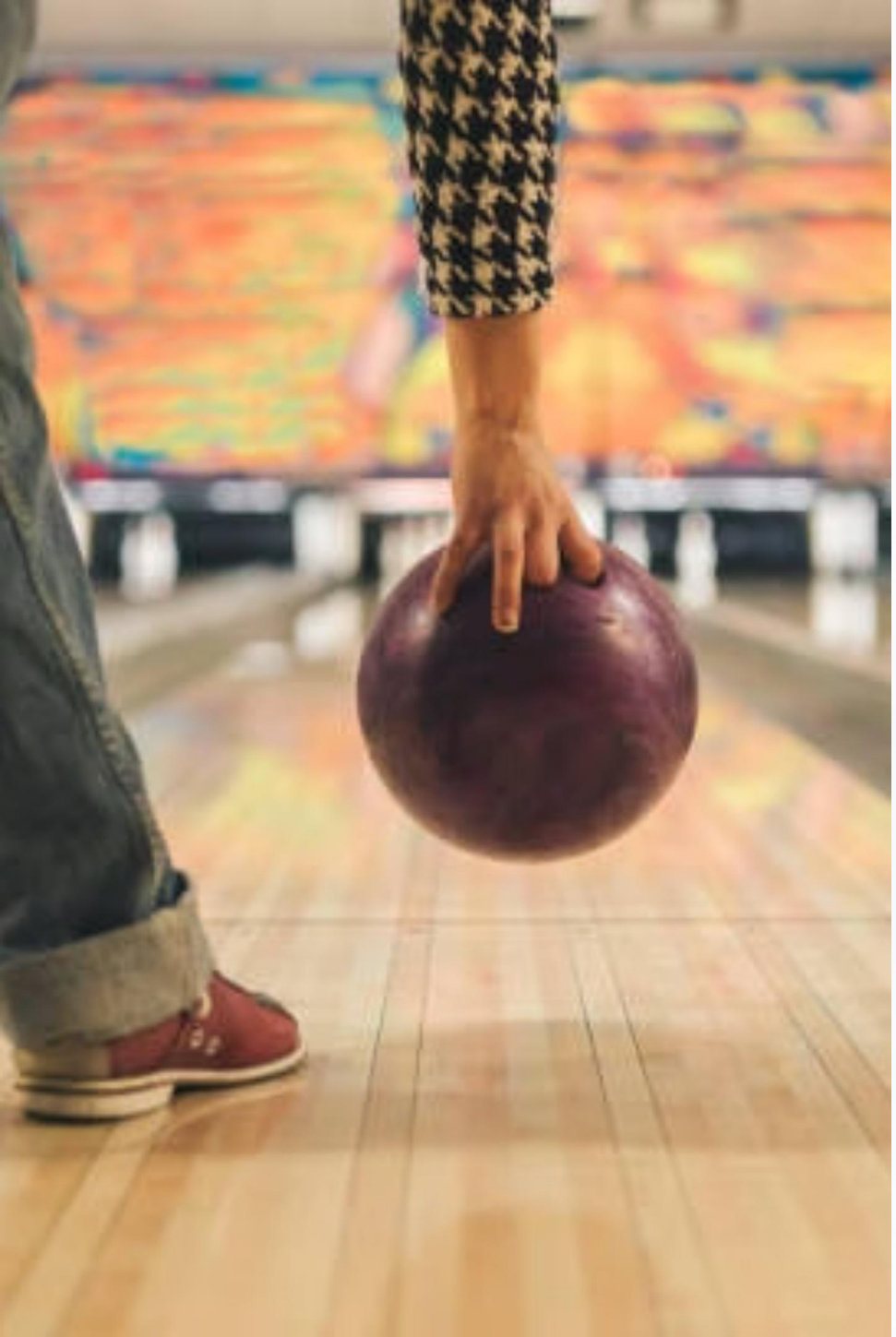
Less than 10 pins in two rolls. Frame score is simply the sum of pins knocked down in that frame.

Spare (/)

Exactly 10 pins in two rolls. Frame score is 10 plus the pins knocked down in the next single roll.

Strike (X)

10 pins in the first roll of a frame. Frame score is 10 plus the pins knocked down in the next two rolls.



Input Format: Standardized Notation

Our module uses a string format, inspired by traditional bowling score sheets, to represent game rolls.

Digits 0-9

Represents the number of pins knocked down in a roll.

X: Strike

Indicates 10 pins knocked down on the first roll.

/: Spare

Indicates 10 pins knocked down over two rolls in a frame.

-: Miss

Represents 0 pins knocked down in a roll.

Input Validation: Ensuring Accuracy

Before scoring, the `validate_input` function performs critical checks on the game string to ensure its structural correctness. This foundational step is vital for preventing errors and guaranteeing accurate, reliable calculations.

```
def validate_input(game: str):
    """
    This function checks whether the input is correct or not.
    """
    # ... (code for validation logic).......
```

This validation specifically confirms the game string adheres to the structural rules of a complete bowling game (e.g., number of frames, rolls per frame). It ensures the individual characters representing pins knocked down (0-9, X, /) are syntactically valid .

Core Logic: Calculating the Score

At the heart of the system, the `calculate_score` function takes a list of validated roll scores and applies the complex scoring rules of bowling to accurately determine the total game score. Below is a detailed breakdown of its logic:

- **Frame Processing:** The function iterates through the 10 frames, accumulating the score progressively.
- **Strike Handling:** If a strike (10 pins on first roll) occurs, 10 points plus the scores of the next two rolls are added to the total.
- **Spare Handling:** If a spare (10 pins over two rolls) occurs, 10 points plus the score of the next single roll are added to the total.
- **Open Frame Handling:** For open frames (neither strike nor spare), only the scores of the two rolls within that frame are added.

```
def calculate_score(rolls:list) -> int:
 """
 This function calculate the total score of a single ten-pin bowling game.
 """
 # ..... remining score calculaltion logic......
```

Illustrative Examples

Let's look at how the module handles different game scenarios, demonstrating its accuracy across various roll combinations.

"9-9-9-9-9-9-9-9-9-9-"

All open frames, each scoring 9 pins. Total: 90.

"XXXXXXXXXXXX"

A perfect game with 12 strikes. Total: 300.

"5/5/5/5/5/5/5/5/5/5/5"

All spares, with each first roll knocking down 5 pins. Total: 150.

Conclusion

Robust & Reliable

This Python module provides a precise and dependable way to calculate ten-pin bowling scores. It accurately handles the game's complex rules, ensuring correct scoring for all roll combinations.

Ultimately, this module delivers a flawless and indispensable scoring solution, mastering the intricate rules of ten-pin bowling with unparalleled precision and reliability.