OUP OF INSTITUTIONS

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

AUTONOMOUS INSTITUTE UNDER VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA, BELAGAVI 590018

DEPARTMENT OF ELECTRICAL & ELECTRONICS ENGINEERING

Seminar Report on

“Match Result Tracker”

For the course: Python

Names	usn no
Swathi HE	3BR23EC161
U Sahana	3BR23EC174
Harshitha G	3BR23EC055
Shravani VD	3BR23EC153
Kruthi Y	3BR23EC074

BALLARI INSTITUTE OF TECHNOLOGY & MANAGEMENT

NACC Accredited institution*
(Recognised by Govt.of karnataka,approved by AICTE,NewDelhi & Affiliated to Visvesvaraya Technological University, Belgavi)
“Jnana Gangotri” Campus,No.873/2,Ballari-Hospet Road,Allipur,Ballari-583104
Karnataka,India.
Ph: 08392-237100/23719, Fax:08392-237197

2023-2024

MATCH RESULT TRACKER

TEAM CODE CRUSHERS

TEAM MEMBERS:

SWATHI.H.E -3BR23EC161
SAHANA.U -3BR23EC174
SHRAVANI.V.D -3BR23EC153
HARSHITHA.G -3BR23EC055
KRUTHI.Y -3BR23EC074

Introduction

Purpose:

A match result tracker is designed to record and monitor the outcomes of sports matches or competitions. Functionality: Users input data such as team names, scores, and relevant details into the tracker.

Organization: The tracker organizes this data in a user-friendly format, allowing for easy retrieval and analysis.Audience: It caters to sports fans, coaches, analysts, and media professionals interested in tracking team and player performance.

Formats: Match result trackers can range from simple spreadsheets to more sophisticated software applications, depending on user needs.

Problem statement

- The current process is being executed manually
- More time consumption
- Need Human Interaction

Objectives

Accurate Tracking: Ensure accurate recording and tracking of match outcomes to provide reliable data for analysis.

Efficient Data Recording: Enable users to quickly input match results, including team names, scores, and relevant details.

User-Friendly Interface: Design an intuitive interface that makes it easy for users to navigate and input data without encountering unnecessary complexity.

Data Analysis Features: Provide tools or functionalities for users to analyze the recorded data, such as generating statistics, trends, or visual representations.

Data Security: Implement measures to safeguard user data, including backups and encryption, to maintain confidentiality and integrity.

Algorithm

Define Functions: Define functions for loading data , adding a match, displaying match results, displaying team performance and updating match data.

Add a Match: Allow users to add a new match by entering details such as team names and scores. This is done through the add_match function.

Delete a Team: Allow users to delete a team by entering details such as team name ,this is done through the delete_team function.

Update: Allow users to update the scores of the individual teams and results of the teams .

Display Match Results: Provide an option to display match results, including team names and scores. This is implemented in the display_results function.

Module split up

- **Create**:The program allows users to create new teams and record their matches.

- **Delete**:The program includes the ability to delete teams from the tracker.

- **Update**:The program automatically update the teams’records when matches are added.

Display:The program displays the current records of all teams

Conclusion

Performance Tracking:

The code facilitates the tracking of team performance over multiple matches by aggregating the total runs scored by each team. Teams with consistently higher total runs indicate stronger batting lineups or better overall performance. Conversely, teams with lower total runs may need to focus on improving their batting strategies or player performance. Consistent improvement or decline in total runs can provide insights into the overall trajectory of a team's performance.

Future Enhancement

- Future enhancement for the live tracker of cricket match results and displaying the live score boards.

```
class Team:
    def _init_(self, name):
        self.name = name
        self.matches_played = 0
        self.wins = 0
        self.losses = 0
        self.ties = 0

    def record_win(self):
        self.matches_played += 1
        self.wins += 1

    def record_loss(self):
        self.matches_played += 1
        self.losses += 1

    def record_tie(self):
        self.matches_played += 1
        self.ties += 1

    def _str_(self):
        return f"{self.name} - Played: {self.matches_played},
Wins: {self.wins}, Losses: {self.losses}, Ties: {self.ties}"

# Define a class for the cricket match
class Match:
    def _init_(self, team1, team2, score1, score2):
        self.team1 = team1
        self.team2 = team2
        self.score1 = score1
        self.score2 = score2

    def result(self):
        if self.score1 > self.score2:
            self.team1.record_win()
            self.team2.record_loss()
            return f"{self.team1.name} won the match by
{self.score1 - self.score2} runs."
        elif self.score2 > self.score1:
            self.team2.record_win()
            self.team1.record_loss()
            return f"{self.team2.name} won the match by
{self.score2 - self.score1} runs."
        else:
            self.team1.record_tie()
            self.team2.record_tie()
            return "The match was a tie."

# Define a class to track the cricket matches
class CricketTracker:
    def _init(self):  # Corrected __init_ method
        self.teams = {}

    def add_team(self, team_name):
        if team_name not in self.teams:
            self.teams[team_name] = Team(team_name)
        return self.teams[team_name]

    def add_match(self, team1_name, team2_name, score1,
score2):
        team1 = self.add_team(team1_name)
        team2 = self.add_team(team2_name)
        match = Match(team1, team2, score1, score2)
        print(match.result())

    def delete_team(self,team_name):
        if team_name in self.teams:
            del self.teams[team_name]
            print(f"Team '{team_name}'has been deleted.")
        else:
            print(f"Team '{team_name}' does not exist.")

    def display_teams(self):
        print("\nTeams' Records:")
        for team in self.teams.values():
            print(team)

# Usage example:
if _name_ == "_main_":
    tracker = CricketTracker()

    # Adding matches and their results
    tracker.add_match("RCB", "CSK", 250, 230)
    tracker.add_match("MI", "RCB", 300, 300)
    tracker.add_match("CSK", "MI", 270, 260)

    # Display teams and their records
    tracker.display_teams()
    tracker.delete_team("CSK")
    tracker.display_teams()
```

```
RCB won the match by 20 runs.
The match was a tie.
CSK won the match by 10 runs.

Teams' Records:
RCB - Played: 2, Wins: 1, Losses: 0, Ties: 1
CSK - Played: 2, Wins: 1, Losses: 1, Ties: 0
MI - Played: 2, Wins: 0, Losses: 1, Ties: 1
Team 'CSK'has been deleted.

Teams' Records:
RCB - Played: 2, Wins: 1, Losses: 0, Ties: 1
MI - Played: 2, Wins: 0, Losses: 1, Ties: 1
```

Team` Class

This class represents a cricket team and tracks various statistics.

Attributes

name`: Name of the team.
- `matches_played: Number of matches played by the team
`wins: Number of matches won.
`losses`: Number of matches lost.
- `ties: Number of tied matches.

- **Methods**:
  - `__init__(self, name)`: Initializes the team with its name and zero stats.
  - `record_win(self)`: Increments the `matches_played` and `wins` count.
  - `record_loss(self)`: Increments the `matches_played` and `losses` count.
  - `record_tie(self)`: Increments the `matches_played` and `ties` count.
  - `__str__(self)`: Returns a formatted string with the team’s stats.

Match Class:

This class represents a cricket match between two teams and handles the result computation based on scores.

Attributes:
- `team1`: First team playing the match.
- `team2`: Second team playing the match.
- `score1`: Score of `team1`.

- `score2`: Score of `team2`.

-Methods

- `__init__(self, team1, team2, score1, score2)`: Initializes the match with the teams and their respective scores.
- `result(self)`: Determines and records the result of the match based on the scores:
  - If `team1` has a higher score, `team1` wins.
  - If `team2` has a higher score, `team2` wins.
  - If both scores are equal, the match is a tie.

`CricketTracker` Class:

This class manages the teams, matches, and results, and allows saving and loading of team statistics.

Attributes:

- `teams`: A dictionary storing `Team` objects, with team names as keys.

Methods:

- `__init__(self)`: Initializes an empty `teams` dictionary.
- `add_team(self, team_name)`: Adds a new team to the tracker if it doesn't already exist.
- `add_match(self, team1_name, team2_name, score1, score2)`: Records the result of a match between two teams.
- `display_teams(self)`: Displays all teams and their stats.
-

`delete_team()`:

- The method checks if the given `team_name` exists in the `teams` dictionary.
- If it exists, it deletes the team and prints a confirmation

message.
- If the team is not found, it informs the user that the team does not exist.