import random

import time

# ------------------ Data ------------------

matches = {

"1": {"teams": ["India", "Australia"], "type": "ODI", "overs": 50},

"2": {"teams": ["CSK", "MI"], "type": "T20", "overs": 20},

"3": {"teams": ["England", "Pakistan"], "type": "Test", "overs": 90},

}

players = {

"India": ["Rohit Sharma", "Virat Kohli", "KL Rahul", "Shubman Gill", "Hardik Pandya",

"Ravindra Jadeja", "Axar Patel", "Kuldeep Yadav", "Jasprit Bumrah", "Mohammed Shami", "Mohammed Siraj"],

"Australia": ["David Warner", "Steve Smith", "Marnus Labuschagne", "Glenn Maxwell", "Pat Cummins",

"Cameron Green", "Mitchell Marsh", "Alex Carey", "Josh Hazlewood", "Nathan Lyon", "Mitchell Starc"],

"CSK": ["MS Dhoni", "Ruturaj Gaikwad", "Shivam Dube", "Moeen Ali", "Ravindra Jadeja",

"Ben Stokes", "Deepak Chahar", "Maheesh Theekshana", "Matheesha Pathirana", "Tushar Deshpande", "Devon Conway"],

"MI": ["Rohit Sharma", "Ishan Kishan", "Suryakumar Yadav", "Cameron Green", "Tilak Varma",

"Tim David", "Jofra Archer", "Piyush Chawla", "Jason Behrendorff", "Riley Meredith", "Jasprit Bumrah"],

"England": ["Joe Root", "Ben Stokes", "Jonny Bairstow", "Jos Buttler", "Harry Brook",

"Moeen Ali", "Chris Woakes", "Mark Wood", "Jofra Archer", "James Anderson", "Stuart Broad"],

"Pakistan": ["Babar Azam", "Mohammad Rizwan", "Fakhar Zaman", "Imam-ul-Haq", "Shadab Khan",

"Iftikhar Ahmed", "Mohammad Nawaz", "Haris Rauf", "Shaheen Afridi", "Hasan Ali", "Naseem Shah"],

}

team\_roles = {

"India": {"captain": "Virat Kohli", "wicketkeeper": "KL Rahul"},

"Australia": {"captain": "Steve Smith", "wicketkeeper": "Alex Carey"},

"CSK": {"captain": "MS Dhoni", "wicketkeeper": "MS Dhoni"},

"MI": {"captain": "Rohit Sharma", "wicketkeeper": "Ishan Kishan"},

"England": {"captain": "Joe Root", "wicketkeeper": "Jos Buttler"},

"Pakistan": {"captain": "Babar Azam", "wicketkeeper": "Mohammad Rizwan"},

}

# ------------------ Utility Functions ------------------

def show\_matches():

print("\nAvailable Matches:")

for key, match in matches.items():

print(f"{key}. {match['teams'][0]} vs {match['teams'][1]} - {match['type']} Match")

def toss(team1, team2):

winner = random.choice([team1, team2])

decision = random.choice(["bat", "bowl"])

print(f"\nToss Winner: {winner}")

print(f"{winner} has chosen to {decision} first.")

return winner, decision

def show\_playing\_xi(team):

print(f"\nPlaying XI for {team}:")

print(f" Captain: {team\_roles[team]['captain']}")

print(f" Wicketkeeper: {team\_roles[team]['wicketkeeper']}")

for player in players[team]:

if player not in (team\_roles[team]['captain'], team\_roles[team]['wicketkeeper']):

print(f" - {player}")

# ------------------ Innings Simulation ------------------

def simulate\_innings(team, overs\_limit, match\_type, bowling\_team, target=None):

"""

Simulates an innings over-by-over.

If 'target' is provided, the innings ends as soon as total\_score reaches or exceeds it.

Returns:

total\_score, wickets, batting\_scorecard, over\_analysis, bowling\_scorecard

"""

print(f"\n🏏 {team} Batting Innings (Live) 🏏")

print(f"{team} is batting and {bowling\_team} is bowling.")

total\_score = 0

wickets = 0

batsman\_index = 0

batting\_scorecard = {}

current\_batsman = players[team][batsman\_index]

batting\_scorecard[current\_batsman] = {"Runs": 0, "Balls": 0, "Fours": 0, "Sixes": 0}

over\_analysis = {}

# Select bowlers – use last 5 players from the bowling team

bowlers = players[bowling\_team][-5:]

bowling\_scorecard = {bowler: {"Overs": 0, "Runs": 0, "Wickets": 0, "Maidens": 0} for bowler in bowlers}

# Set maximum overs per bowler for limited overs formats

if match\_type == "T20":

max\_bowler\_overs = 4

elif match\_type == "ODI":

max\_bowler\_overs = 10

else:

max\_bowler\_overs = overs\_limit # For Test, no strict limit

outcomes = ["0", "1", "2", "3", "4", "6", "W"]

outcome\_weights = [0.25, 0.20, 0.10, 0.05, 0.20, 0.10, 0.10]

for over in range(1, overs\_limit + 1):

if wickets >= 10 or (target is not None and total\_score >= target):

break

available\_bowlers = [b for b in bowlers if bowling\_scorecard[b]["Overs"] < max\_bowler\_overs]

if available\_bowlers:

bowler = random.choice(available\_bowlers)

else:

bowler = random.choice(bowlers)

bowling\_scorecard[bowler]["Overs"] += 1

runs\_this\_over = 0

maiden = True

for ball in range(6):

if wickets >= 10 or (target is not None and total\_score >= target):

break

outcome = random.choices(outcomes, weights=outcome\_weights, k=1)[0]

batting\_scorecard[current\_batsman]["Balls"] += 1

if outcome == "W":

wickets += 1

bowling\_scorecard[bowler]["Wickets"] += 1

print(f"Over {over}, Ball {ball+1}: {current\_batsman} is OUT! Score: {total\_score}/{wickets}")

maiden = False

batsman\_index += 1

if batsman\_index < len(players[team]):

current\_batsman = players[team][batsman\_index]

batting\_scorecard[current\_batsman] = {"Runs": 0, "Balls": 0, "Fours": 0, "Sixes": 0}

else:

break

else:

runs = int(outcome)

total\_score += runs

runs\_this\_over += runs

batting\_scorecard[current\_batsman]["Runs"] += runs

if runs == 4:

batting\_scorecard[current\_batsman]["Fours"] += 1

elif runs == 6:

batting\_scorecard[current\_batsman]["Sixes"] += 1

if runs > 0:

maiden = False

print(f"Over {over}, Ball {ball+1}: {current\_batsman} scores {runs} run(s) | Total: {total\_score}/{wickets}")

time.sleep(0.2)

# If target reached exactly, break out immediately.

if target is not None and total\_score >= target:

break

over\_analysis[over] = runs\_this\_over

bowling\_scorecard[bowler]["Runs"] += runs\_this\_over

if maiden:

bowling\_scorecard[bowler]["Maidens"] += 1

print(f"End of Over {over}: {runs\_this\_over} runs | Score: {total\_score}/{wickets}")

# If target reached exactly, no need for extra balls/overs.

if target is not None and total\_score >= target:

break

print(f"\n{team} Innings Ended: {total\_score}/{wickets} in {over} overs")

# Batting Scorecard Display

print("\nBatting Scorecard:")

print(f"{'Batsman':<20} {'Runs':<5} {'Balls':<5} {'4s':<3} {'6s':<3} {'SR':<6}")

for batsman, stats in batting\_scorecard.items():

sr = round((stats["Runs"] / stats["Balls"]) \* 100, 2) if stats["Balls"] > 0 else 0.0

print(f"{batsman:<20} {stats['Runs']:<5} {stats['Balls']:<5} {stats['Fours']:<3} {stats['Sixes']:<3} {sr:<6}")

# Over-to-Over Analysis Display

print("\nOver-to-Over Analysis:")

for ov, runs in over\_analysis.items():

print(f"Over {ov}: {runs} runs")

# Bowling Scorecard Display

print("\nBowling Scorecard:")

print(f"{'Bowler':<20} {'Overs':<5} {'Runs':<5} {'Wkts':<5} {'Maidens':<7}")

for bowler, stats in bowling\_scorecard.items():

print(f"{bowler:<20} {stats['Overs']:<5} {stats['Runs']:<5} {stats['Wickets']:<5} {stats['Maidens']:<7}")

return total\_score, wickets, batting\_scorecard, over\_analysis, bowling\_scorecard

# ------------------ Match Simulation ------------------

def standard\_match\_simulation(team1, team2, match\_type, overs\_limit):

print(f"\n🏏 {team1} vs {team2} - {match\_type} Match Start! 🏏")

toss\_winner, decision = toss(team1, team2)

if decision == "bat":

first\_batting = toss\_winner

second\_batting = team2 if toss\_winner == team1 else team1

else:

first\_batting = team2 if toss\_winner == team1 else team1

second\_batting = toss\_winner

show\_playing\_xi(team1)

show\_playing\_xi(team2)

# First Innings (no target)

print(f"\n--- First Innings: {first\_batting} batting ---")

first\_score, first\_wkts, bat1, over1, bowl1 = simulate\_innings(first\_batting, overs\_limit, match\_type, second\_batting)

print("\n--- Innings Break ---\n")

time.sleep(2)

# Second Innings (target is first\_score + 1)

target = first\_score + 1

print(f"\n--- Second Innings: {second\_batting} batting ---")

print(f"Target for {second\_batting} is {target} runs.")

second\_score, second\_wkts, bat2, over2, bowl2 = simulate\_innings(second\_batting, overs\_limit, match\_type, first\_batting, target=target)

print("\n🏆 MATCH RESULT 🏆")

if second\_score >= target:

margin = 10 - second\_wkts

print(f"🎉 {second\_batting} wins by {margin} wickets!")

else:

margin = first\_score - second\_score

print(f"🎉 {first\_batting} wins by {margin} runs!")

combined = {}

combined.update(bat1)

for batsman, stats in bat2.items():

if batsman in combined:

combined[batsman]["Runs"] += stats["Runs"]

else:

combined[batsman] = stats

potm = max(combined.items(), key=lambda x: x[1]["Runs"])

print(f"\n⭐ PLAYER OF THE MATCH: {potm[0]} with {potm[1]['Runs']} runs ⭐")

def test\_match\_simulation(team1, team2, overs\_limit):

"""

Simulates a Test match with four innings.

Implements follow-on if first innings lead is 200+ runs.

Displays lead/trail after each innings.

"""

print(f"\n🏏 {team1} vs {team2} - Test Match Start! 🏏")

toss\_winner, decision = toss(team1, team2)

if decision == "bat":

first\_batting = toss\_winner

second\_batting = team2 if toss\_winner == team1 else team1

else:

first\_batting = team2 if toss\_winner == team1 else team1

second\_batting = toss\_winner

show\_playing\_xi(team1)

show\_playing\_xi(team2)

# 1st Innings

print(f"\n--- 1st Innings: {first\_batting} batting ---")

score1, wkts1, bat1, over1, bowl1 = simulate\_innings(first\_batting, overs\_limit, "Test", second\_batting)

# 1st Innings for second side

print(f"\n--- 1st Innings: {second\_batting} batting ---")

score2, wkts2, bat2, over2, bowl2 = simulate\_innings(second\_batting, overs\_limit, "Test", first\_batting)

lead\_after\_1st = score1 - score2

if lead\_after\_1st > 0:

print(f"\nAfter 1st innings, {first\_batting} leads by {lead\_after\_1st} runs.")

else:

print(f"\nAfter 1st innings, {second\_batting} leads by {abs(lead\_after\_1st)} runs.")

follow\_on = False

if lead\_after\_1st >= 200:

print(f"\n{second\_batting} is asked to follow on!")

follow\_on = True

if follow\_on:

# 2nd Innings (Follow-on for second side)

print(f"\n--- 2nd Innings (Follow-On): {second\_batting} batting ---")

score3, wkts3, bat3, over3, bowl3 = simulate\_innings(second\_batting, overs\_limit, "Test", first\_batting)

new\_lead = score1 - score3

print(f"\nAfter follow-on, {first\_batting} leads by {new\_lead} runs.")

# 4th Innings: Final chase by first side

print(f"\n--- 2nd Innings: {first\_batting} batting ---")

target4 = new\_lead + 1

print(f"Target for {first\_batting} is {target4} runs.")

score4, wkts4, bat4, over4, bowl4 = simulate\_innings(first\_batting, overs\_limit, "Test", second\_batting, target=target4)

if score4 >= target4:

margin = 10 - wkts4

print(f"\n🎉 {first\_batting} wins by {margin} wickets!")

else:

margin = target4 - score4

print(f"\n🎉 {second\_batting} wins by {margin} runs!")

else:

# Normal sequence without follow-on: 2nd Innings for first side

print(f"\n--- 2nd Innings: {first\_batting} batting ---")

score3, wkts3, bat3, over3, bowl3 = simulate\_innings(first\_batting, overs\_limit, "Test", second\_batting)

trail = score3 - score2

if trail > 0:

print(f"\nAfter 2nd innings, {second\_batting} trails by {trail} runs.")

else:

print(f"\nAfter 2nd innings, {second\_batting} leads by {abs(trail)} runs.")

# 4th Innings for second side

print(f"\n--- 2nd Innings: {second\_batting} batting ---")

target4 = score3 - score2 + 1

print(f"Target for {second\_batting} is {target4} runs.")

score4, wkts4, bat4, over4, bowl4 = simulate\_innings(second\_batting, overs\_limit, "Test", first\_batting, target=target4)

if score4 >= target4:

margin = 10 - wkts4

print(f"\n🎉 {second\_batting} wins by {margin} wickets!")

else:

margin = target4 - score4

print(f"\n🎉 {first\_batting} wins by {margin} runs!")

combined = {}

for innings in [bat1, bat2]:

for player, stats in innings.items():

combined[player] = combined.get(player, 0) + stats["Runs"]

potm = max(combined.items(), key=lambda x: x[1])

print(f"\n⭐ PLAYER OF THE MATCH: {potm[0]} with {potm[1]} runs ⭐")

# ------------------ Main Simulation ------------------

def match\_simulation(team1, team2, match\_type, overs\_limit):

if match\_type == "Test":

test\_match\_simulation(team1, team2, overs\_limit)

else:

standard\_match\_simulation(team1, team2, match\_type, overs\_limit)

def main():

show\_matches()

choice = input("\nEnter the match number you want to watch: ")

if choice in matches:

match = matches[choice]

team1, team2 = match["teams"]

match\_type, overs\_limit = match["type"], match["overs"]

if match\_type == "ODI":

overs\_limit = 50

elif match\_type == "T20":

overs\_limit = 20

elif match\_type == "Test":

overs\_limit = 90

match\_simulation(team1, team2, match\_type, overs\_limit)

else:

print("\nInvalid selection. Please try again.")

if \_\_name\_\_ == "\_\_main\_\_":

main()









