BSc. Artificial Intelligence & Data Science

Level 04

CM 1601

PROGRAMMING FUNDAMENTALS

**T20 Cricket Tournament**

**COURSEWORK-I REPORT**

NADUN SHAMIKA SENARATHNE

IIT ID: 20210488

RGU ID: 2117538

Table of Contents

**Generate random match Flowchart.................................................................3-4**

**Source Code ..................................................................................................5-35**

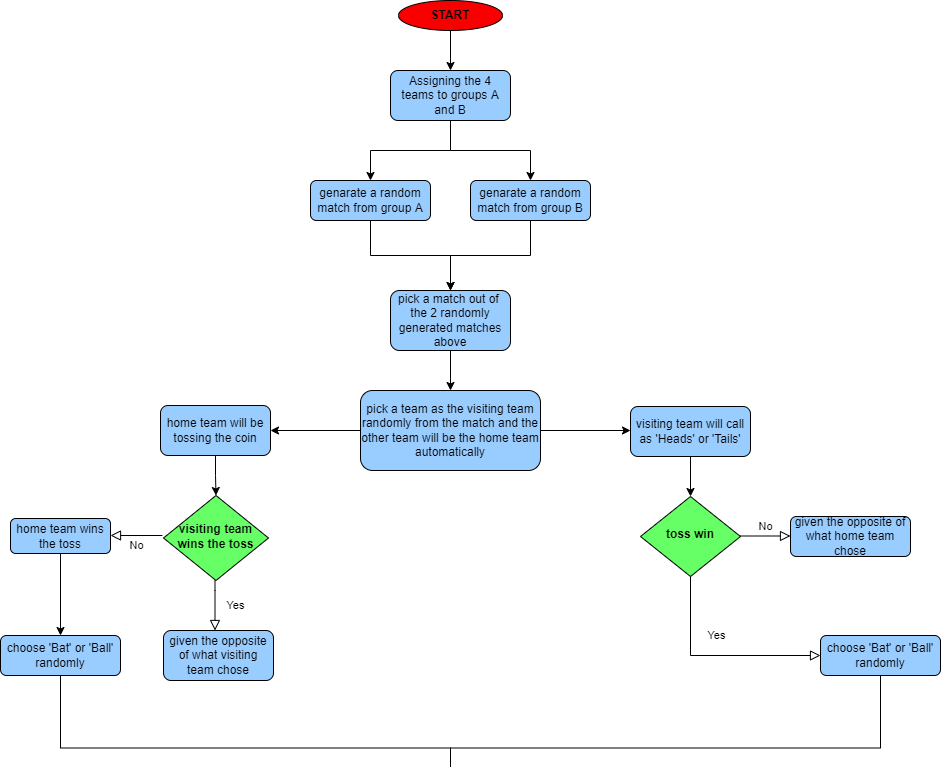
**Cricket.py…………..**...............................................................................................................6-33

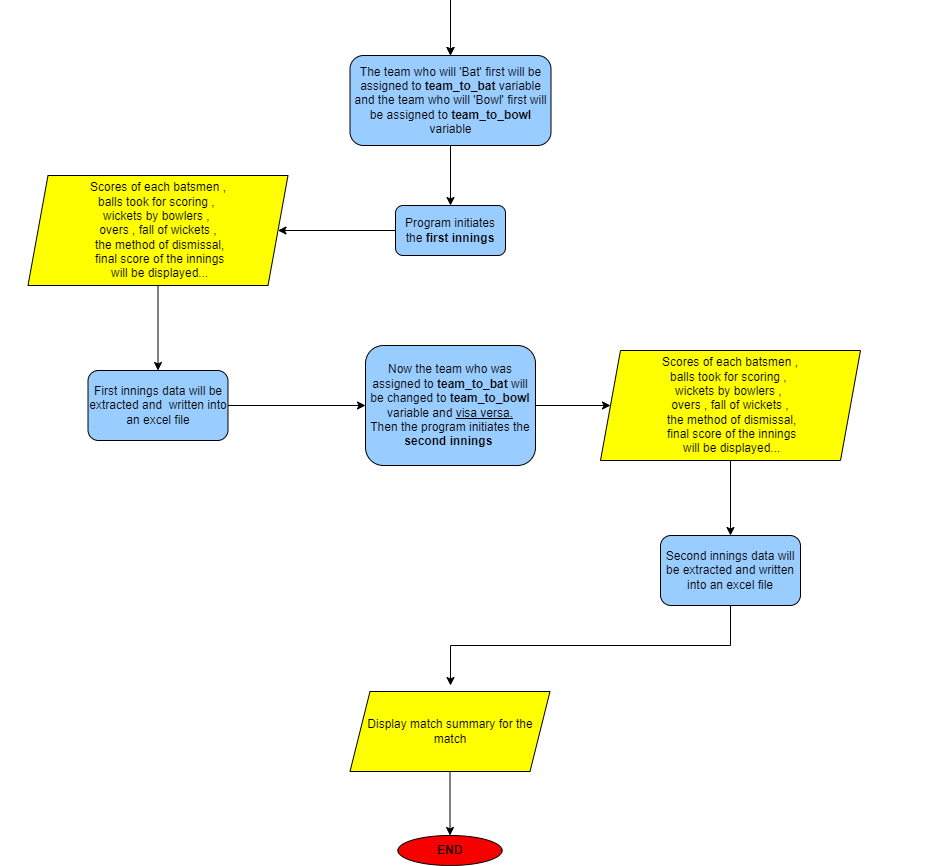
**Coursework.py............................................................................................................34-35**

**Output………………………………………………………………………………………………….……36-45**

**Test Plan .....................................................................................................46-52**

Generate random match Flowchart





Source Code

Assumptions

* User can only edit player names.
* There are no past records of players prior to this tournament.
* Only ways of getting out is by ‘Bowled’,’LBW’ or ‘Caught’ by the bowler himself.
* The hand cricket strategy is used for scoring and dismissals. (When the **batter\_score** is equal to the **bolwer\_score** the batsman is declared out otherwise the batsman will get runs)
* Extras are not considered as a method of scoring in this tournament.

Cricket.py

# ----------------------------------------------------------------------------------------------------Importing Modules------------------------------------------------------------------------------------------------

import openpyxl

from openpyxl import load\_workbook

from openpyxl.drawing.image import Image

import random

import pandas as pd

from operator import itemgetter

import os

import matplotlib.pyplot as plt

from PIL import Image

# ------------------------------------------------------------------------------------------Store information about teams and players.---------------------------------------------------------------------------------

Mumbai\_India = ['Mumbai\_India',

                r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Mumbai\_India\\Mumbai\_India.xlsx']

Chennai\_SouthAfrica = ['Chennai\_SouthAfrica',

                       r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Chennai\_SouthAfrica\\Chennai\_SouthAfrica.xlsx']

Delhi\_NewZealand = ['Delhi\_NewZealand',

                    r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Delhi\_NewZealand\\Delhi\_NewZealand.xlsx']

RoyalChallengers\_Bangladesh = ['RoyalChallengers\_Bangladesh',

                               r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\RoyalChallengers\_Bangladesh\\RoyalChallengers\_Bangladesh.xlsx']

Rajastan\_Australia = ['Rajastan\_Australia',

                      r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Rajastan\_Australia\\Rajastan\_Australia.xlsx']

Kolkata\_England = ['Kolkata\_England',

                   r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Kolkata\_England\\Kolkata\_England.xlsx']

Punjab\_Pakistan = ['Punjab\_Pakistan',

                   r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Punjab\_Pakistan\\Punjab\_Pakistan.xlsx']

Sunrisers\_SriLanka = ['Sunrisers\_SriLanka',

                      r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\Sunrisers\_SriLanka\\Sunrisers\_SriLanka.xlsx']

# ------------------------------------------------------------------------------------------------- Assigning teams to groups------------------------------------------------------------------------------------------

Group\_A = [Mumbai\_India, Chennai\_SouthAfrica,

           Delhi\_NewZealand, RoyalChallengers\_Bangladesh]

Group\_B = [Rajastan\_Australia, Kolkata\_England,

           Punjab\_Pakistan, Sunrisers\_SriLanka]

# ---------------------------------------------------------------------------------------Global variables which are used in the functions------------------------------------------------------------------------------

user\_input = ''

global\_exit = ''

TOTAL\_WICKETS = 10

TOTAL\_BALLS = 120

first\_ing\_total = 0

first\_ing\_wickets = 0

second\_ing\_total = 0

second\_ing\_wickets = 0

match\_between = []

team\_to\_bat = []

team\_to\_bowl = []

visiting\_team = []

home\_team = []

filename\_match = ''

selection = ''

toss = ''

choose = ''

df\_score\_card\_first\_ing\_without\_index = []

df\_bowler\_list\_first\_ing\_without\_index = []

df\_score\_card\_second\_ing\_without\_index = []

df\_bowler\_list\_second\_ing\_without\_index = []

winning\_team = []

losing\_team = []

graph\_first\_ing\_balls = []

graph\_first\_ing\_total = []

graph\_first\_ing\_fow\_balls = []

graph\_first\_ing\_fow\_total = []

graph\_second\_ing\_balls = []

graph\_second\_ing\_total = []

graph\_second\_ing\_fow\_balls = []

graph\_second\_ing\_fow\_total = []

# ---------------------------------------------------------------------------------------------------Defining the functions----------------------------------------------------------------------------------------------

def GetTeam(teamName):

    print(GetTeam)

    df = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\'+teamName+'\\'+teamName+'.xlsx')

    editable\_options = df.iloc[:, :1]

    print(editable\_options)

def EditTeam(teamName, row, col, edited\_name):

    df = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\'+teamName+'\\'+teamName+'.xlsx')

    df.at[row, col] = edited\_name

    print(df.iloc[:, 0])

    df.to\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\'+teamName+'\\'+teamName+'.xlsx', index=False)

def editPlayer(teamName):

    GetTeam(teamName)

    editTeam = input(

        f"Do you want to make any changes on team {teamName}? \n1 - yes \n0 - no ")

    if editTeam == '0':

        global global\_exit

        global\_exit = 'y'

    while editTeam == '1':

        row = int(input(

            "Which player do you want to edit? \nSelect the corresponding row number :  "))

        col = 'PLAYER NAME'

        val = input("What should be the change then? ")

        # --------------------------------------------------------------Update Player name in player\_standings when user edit name--------------------------------------------------------------------

        team = r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\team\_data\\'+teamName+'\\'+teamName+'.xlsx'

        player\_standings = pd.read\_excel(

            r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx')

        df\_player\_standings = pd.DataFrame(player\_standings)

        wb\_obj = openpyxl.load\_workbook(team)

        sheet\_obj = wb\_obj.active

        player\_name = sheet\_obj.cell(row=row+2, column=1).value

        find\_player\_standing\_index = df\_player\_standings.index[df\_player\_standings['PLAYER NAME'] == player\_name].tolist(

        )

        df\_player\_standings.at[find\_player\_standing\_index, 'PLAYER NAME'] = val

        writer = pd.ExcelWriter(

            r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx', engine='xlsxwriter')

        workbook = writer.book

        worksheet = workbook.add\_worksheet('Sheet1')

        writer.sheets['Sheet1'] = worksheet

        df\_player\_standings.to\_excel(

            writer, sheet\_name='Sheet1', startrow=0, startcol=0, index=False)

        writer.save()

        writer.close()

        EditTeam(teamName, row, col, val)

        print("your changes have been saved successfully !!!")

        editTeam = input(

            f"Do you want to make any more changes on team {teamName} again?\n1 - yes \n0 - no ")

        if editTeam == '0':

            global\_exit = 'y'

        else:

            global\_exit = ''

def team\_profile\_edit(getData):

    global global\_exit

    while (getData == '2') and (global\_exit != 'y'):

        getGroup = input(

            "Which group do you want to see? \n1 - group A \n2 - group B \nOr press 'x' to exit...     ")

        if getGroup == 'x':

            break

        elif getGroup == '1':

            getTeam = input("Which team do you want to see? \n1 - Mumbai India \n2 - Chennai SouthAfrica \n3 - Delhi NewZealand \n4 - RoyalChallengers Bangladesh \n Select a number from 1 to 4 \nOr press 'x' to exit...       ")

            if getTeam == 'x':

                break

            if getTeam == '1':

                editPlayer('Mumbai\_India')

            elif getTeam == '2':

                editPlayer('Chennai\_SouthAfrica')

            elif getTeam == '3':

                editPlayer('Delhi\_NewZealand')

            elif getTeam == '4':

                editPlayer('RoyalChallengers\_Bangladesh')

        elif getGroup == '2':

            getTeam = input(

                "Which team do you want to see? \n1 - Rajastan Australia \n2 - Kolkata England \n3 - Punjab Pakistan \n4 - Sunrisers SriLanka \n Select a number from 1 to 4")

            if getTeam == '1':

                editPlayer('Rajastan\_Australia')

            elif getTeam == '2':

                editPlayer('Kolkata\_England')

            elif getTeam == '3':

                editPlayer('Punjab\_Pakistan')

            elif getTeam == '4':

                editPlayer('Sunrisers\_SriLanka')

    else:

        global\_exit = ''

def generate\_random\_match():

    match\_list = os.listdir(

        r'E:\\IIT\\1st Year\\1st Trimester\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches')

    match\_list\_count = len(match\_list)

    match\_list\_count = match\_list\_count if match\_list\_count > 0 else 1

    while match\_list\_count != 12:

        match\_between\_A = random.sample(Group\_A, 2)

        match\_between\_B = random.sample(Group\_B, 2)

        chosen\_match = [match\_between\_A, match\_between\_B]

        global match\_between

        temp\_match\_between = random.choice(chosen\_match)

        temp1 = str(temp\_match\_between[0][0]) + '\_vs\_' + \

            str(temp\_match\_between[1][0]+'.xlsx')

        temp2 = str(temp\_match\_between[1][0]) + '\_vs\_' + \

            str(temp\_match\_between[0][0]+'.xlsx')

        if (temp1 not in match\_list) and (temp2 not in match\_list):

            match\_between = temp\_match\_between

            break

    else:

        match\_between = []

        raise IndexError('A very specific bad thing happened.')

def points\_table():

    points\_table = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\points\_table.xlsx')

    df\_points\_table = pd.DataFrame(points\_table)

    for team in match\_between:

        if(team in Group\_A):

            group = "Group A"

        else:

            group = "Group B"

        find\_team\_index = df\_points\_table.index[df\_points\_table[group] == team[0]].tolist(

        )

        if(group == "Group A"):

            current\_match\_count = df\_points\_table.at[find\_team\_index[0], 'Matches\_A']

            df\_points\_table.at[find\_team\_index,

                               'Matches\_A'] = current\_match\_count+1

        else:

            current\_match\_count = df\_points\_table.at[find\_team\_index[0], 'Matches\_B']

            df\_points\_table.at[find\_team\_index,

                               'Matches\_B'] = current\_match\_count+1

        print('\n\n')

    # ------------------------------------------------------------------------Write data to excel file by creating Excel Writer Object from Pandas---------------------------------------------------------------------

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\points\_table.xlsx', engine='xlsxwriter')

    workbook = writer.book

    worksheet = workbook.add\_worksheet('Match Summary')

    writer.sheets['Match Summary'] = worksheet

    df\_points\_table.to\_excel(writer, sheet\_name='Match Summary',

                             startrow=0, startcol=0, index=False)

    writer.save()

    writer.close()

def toss\_coin():

    coin = ["heads", "tails"]

    options = ['bat', 'bowl']

    global team\_to\_bat

    global team\_to\_bowl

    global visiting\_team

    global home\_team

    global selection

    global toss

    global choose

    visiting\_team = random.choice(match\_between)

    if visiting\_team in match\_between:

        match\_between.remove(visiting\_team)

    home\_team = match\_between[0]

    # This simulates the coin being tossed

    toss = random.choice(coin)

    # This simulates the visiting team choose head or tails

    selection = random.choice(coin)

    # This simulates the visiting team choose bat or bowl

    choose = random.choice(options)

    print('\n\n')

    print('Home Team - ', home\_team[0])

    print('Visiting Team - ', visiting\_team[0])

    print('\n\n')

    if selection == toss:

        print(visiting\_team[0], 'won the toss and chose to', choose)

        if choose == options[0]:

            team\_to\_bat = visiting\_team

            team\_to\_bowl = home\_team

        else:

            team\_to\_bat = home\_team

            team\_to\_bowl = visiting\_team

    else:

        print(home\_team[0], 'won the toss and chose to', choose)

        if choose == options[0]:

            team\_to\_bat = home\_team

            team\_to\_bowl = visiting\_team

        else:

            team\_to\_bat = visiting\_team

            team\_to\_bowl = home\_team

    print('\n\n')

    print('\n\nteam\_to\_bat', team\_to\_bat[0])

    print('team\_to\_bowl', team\_to\_bowl[0])

    print('\n\n')

def player\_standings(batting, bowling):

    player\_standings = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx')

    df\_player\_standings = pd.DataFrame(player\_standings)

    for player in batting:

        find\_player\_index = df\_player\_standings.index[df\_player\_standings['PLAYER NAME'] == player[0]].tolist(

        )

        if not(len(find\_player\_index) > 0):

            raise Exception('\nPLAYER NOT FOUND!!!!!!!!!')

        current\_player\_runs = df\_player\_standings.at[find\_player\_index[0], 'TOTAL RUNS']

        df\_player\_standings.at[find\_player\_index,

                               'TOTAL RUNS'] = current\_player\_runs+player[1]

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx', engine='xlsxwriter')

    workbook = writer.book

    worksheet = workbook.add\_worksheet('Sheet1')

    writer.sheets['Sheet1'] = worksheet

    df\_player\_standings.to\_excel(

        writer, sheet\_name='Sheet1', startrow=0, startcol=0, index=False)

    writer.save()

    writer.close()

    for player in bowling:

        find\_player\_index = df\_player\_standings.index[df\_player\_standings['PLAYER NAME'] == player[0]].tolist(

        )

        current\_player\_wickets = df\_player\_standings.at[find\_player\_index[0], 'WICKETS']

        df\_player\_standings.at[find\_player\_index,

                               'WICKETS'] = current\_player\_wickets+player[3]

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx', engine='xlsxwriter')

    workbook = writer.book

    worksheet = workbook.add\_worksheet('Sheet1')

    writer.sheets['Sheet1'] = worksheet

    df\_player\_standings.to\_excel(

        writer, sheet\_name='Sheet1', startrow=0, startcol=0, index=False)

    writer.save()

    writer.close()

def display\_player\_standings():

    player\_standings = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\player\_standings.xlsx')

    df\_player\_standings = pd.DataFrame(player\_standings)

    print('\n\nTop 5 run scores of the tournament')

    print(df\_player\_standings[['PLAYER NAME', 'TOTAL RUNS']].nlargest(

        5, 'TOTAL RUNS').to\_string(index=False))

    print('\n\nTop 5 wicket takers of the tournament')

    print(df\_player\_standings[['PLAYER NAME', 'WICKETS']].nlargest(

        5, 'WICKETS').to\_string(index=False))

def first\_innings():

    global filename\_match

    global first\_ing\_total

    global first\_ing\_wickets

    global df\_score\_card\_first\_ing\_without\_index

    global df\_bowler\_list\_first\_ing\_without\_index

    global graph\_first\_ing\_balls

    global graph\_first\_ing\_total

    global graph\_first\_ing\_fow\_balls

    global graph\_first\_ing\_fow\_total

    graph\_first\_ing\_balls = []

    graph\_first\_ing\_total = []

    graph\_first\_ing\_fow\_balls = []

    graph\_first\_ing\_fow\_total = []

    first\_ing\_total = 0

    first\_ing\_wickets = 0

    first\_ing\_balls = 1

    score\_card\_first\_ing = []

    batsman\_onstrike = [['name', 0, 0], True]

    batsman\_offstrike = [['name', 0, 0], False]

    bowler\_onstrike = []

    # batsman\_name , runs , balls\_faced , method of dismissal , bowler

    # importing batting team

    batting\_url = team\_to\_bat[1]

    batting\_team = pd.read\_excel(batting\_url)

    # converting excel to python list

    yet\_to\_bat = batting\_team.values.tolist()

    # bowlers\_name , first\_ing\_balls , runs , wickets , economy

    # importing bowling team

    bowling\_url = team\_to\_bowl[1]

    bowling\_team = pd.read\_excel(bowling\_url)

    # converting excel to python list

    bowling\_team\_list = bowling\_team.values.tolist()

    yet\_to\_bowl = []

    for i in reversed(range(len(bowling\_team\_list))):

        if len(yet\_to\_bowl) < 5:

            yet\_to\_bowl.append([bowling\_team\_list[i][0], 0, 0, 0, 0])

    dismissed\_batsmen = []

    batsman\_list = []

    # method of dismissal

    method\_of\_dismissal = ['Bowled', 'Caught', 'LBW']

    bowler\_score = 0  # score counting variable for bowler

    batter\_score = 0  # score counting variable for batsman

    # opening batsmen coming to the field

    batsman\_onstrike[0] = yet\_to\_bat.pop(0)

    batsman\_offstrike[0] = yet\_to\_bat.pop(0)

    # opening bowler

    bowler\_onstrike = yet\_to\_bowl.pop(0)

    while (first\_ing\_balls < (TOTAL\_BALLS+1)):

        if first\_ing\_wickets == TOTAL\_WICKETS:

            break

        else:

            if ((first\_ing\_balls-1) > 0 and (first\_ing\_balls-1) % 6 == 0) and (len(yet\_to\_bowl)) > 0:

                yet\_to\_bowl.append(bowler\_onstrike)

                bowler\_onstrike = yet\_to\_bowl.pop(0)

            # get random scores for bowler and batsman

            bowler\_score = random.randint(1, 6)

            batter\_score = random.randint(0, 6)

            if bowler\_score == batter\_score:

                # adding wickets to bowler

                current\_bowler\_onstrike\_first\_ing\_wickets = 0

                current\_bowler\_onstrike\_first\_ing\_wickets = bowler\_onstrike[3]

                bowler\_onstrike[3] = current\_bowler\_onstrike\_first\_ing\_wickets + 1

                # adding first\_ing\_balls to batsman

                current\_batsman\_onstrike\_balls = batsman\_onstrike[0][2]

                batsman\_onstrike[0][2] = current\_batsman\_onstrike\_balls + 1

                # moving the dismissed\_batsmen to dismissed\_batsmen array

                dismissed\_batsmen.append(batsman\_onstrike[0])

                # adding method of dismissal to batsman

                current\_batsman\_method\_of\_dismissal = random.choices(

                    method\_of\_dismissal)

                batsman\_onstrike[0][3] = current\_batsman\_method\_of\_dismissal[0]

                # adding dismissed bowler name to batsman

                current\_batsman\_bowler\_dismissed = batsman\_onstrike[0][4]

                batsman\_onstrike[0][4] = current\_batsman\_bowler\_dismissed + \

                    str(bowler\_onstrike[0])

                # fall of wickets

                print('FOW at', first\_ing\_total, ' --> ', first\_ing\_wickets+1,

                      ' on over -', int(first\_ing\_balls/6), '.', (first\_ing\_balls) % 6, batsman\_onstrike[0][0])

                # appending the FOW data to the graph

                graph\_first\_ing\_fow\_balls.append(first\_ing\_balls)

                graph\_first\_ing\_fow\_total.append(first\_ing\_total)

                # bring new batsman to the crease (batsman\_onstrike)

                if len(yet\_to\_bat) > 0:

                    batsman\_onstrike[0] = []

                    batsman\_onstrike[0] = yet\_to\_bat.pop(0)

                # out - add wicket to wickets

                first\_ing\_wickets += 1

            else:

                # adding batter\_score to current\_batsman

                current\_batsman\_onstrike\_score = 0

                current\_batsman\_onstrike\_score = batsman\_onstrike[0][1]

                batsman\_onstrike[0][1] = current\_batsman\_onstrike\_score + batter\_score

                # adding first\_ing\_balls to current\_batsman

                current\_batsman\_onstrike\_balls = batsman\_onstrike[0][2]

                batsman\_onstrike[0][2] = current\_batsman\_onstrike\_balls + 1

                # adding batter\_score to current\_bowler

                current\_bowler\_onstrike\_runs = 0

                current\_bowler\_onstrike\_runs = bowler\_onstrike[2]

                bowler\_onstrike[2] = current\_bowler\_onstrike\_runs + \

                    batter\_score

                # swapping onstrike batsman when strike rotates

                if batter\_score == 1 or batter\_score == 3:

                    current\_batsman = batsman\_onstrike[0]

                    # swapping onstrike batsman

                    batsman\_onstrike[0] = batsman\_offstrike[0]

                    batsman\_offstrike[0] = current\_batsman

                else:

                    pass  # when batter\_score is not swapping

                # add batter score to first\_ing\_total

                first\_ing\_total += batter\_score

        # adding first\_ing\_balls to bowler

        current\_bowler\_onstrike\_balls = 0

        current\_bowler\_onstrike\_balls = bowler\_onstrike[1]

        bowler\_onstrike[1] = current\_bowler\_onstrike\_balls + 1

        # adding first\_ing\_balls to first\_ing ball count

        first\_ing\_balls += 1

        # adding first\_ing\_total to graph\_first\_ing\_total

        graph\_first\_ing\_total.append(first\_ing\_total)

    # assinging first innings balls to graph

    graph\_first\_ing\_balls = range(1, first\_ing\_balls)

    # last dismissed batsman

    last\_dismissal = dismissed\_batsmen[-1]

    # add dismissed\_batsmen to batsman\_list

    batsman\_list = dismissed\_batsmen

    # add each batsman in yet\_to\_bat to batsman\_list array for displaying purposes

    if len(yet\_to\_bat) > 0:

        for i in range(len(yet\_to\_bat)):

            batsman\_list.append(yet\_to\_bat[i])

    # add on and off strike batsmen to batsman\_list

    if first\_ing\_wickets != TOTAL\_WICKETS:

        batsman\_onstrike[0][3] = '\* NOT OUT'

        batsman\_list.append(batsman\_onstrike[0])

    batsman\_offstrike[0][3] = 'NOT OUT'

    batsman\_list.append(batsman\_offstrike[0])

    # add batsman\_list to score\_card\_first\_ing

    score\_card\_first\_ing = batsman\_list

    # add bowlers to bowler\_list\_first\_ing

    bowler\_list\_first\_ing = yet\_to\_bowl

    bowler\_list\_first\_ing.append(bowler\_onstrike)

    # sort score\_card\_first\_ing to the original batting order

    sorted\_list = sorted(score\_card\_first\_ing, key=itemgetter(5))

    # convert score\_card\_first\_ing to a data frame for displaying

    df\_score\_card\_first\_ing = pd.DataFrame(sorted\_list)

    # converting bowler first\_ing\_balls to overs

    for bowler\_overs\_first\_ing in bowler\_list\_first\_ing:

        bowler\_overs\_first\_ing[1] = str(

            int((bowler\_overs\_first\_ing[1])/6)) + '.' + str((bowler\_overs\_first\_ing[1]) % 6)

    # additing the economy for bowler

    for bowler\_economy\_first\_ing in bowler\_list\_first\_ing:

        bowler\_economy\_first\_ing[4] = round(

            bowler\_economy\_first\_ing[2]/float(bowler\_economy\_first\_ing[1]), 2)

    # convert df\_bowler\_list\_first\_ing to a data frame for displaying

    df\_bowler\_list\_first\_ing = pd.DataFrame(bowler\_list\_first\_ing)

    print('\n\n-------------------------------------------------1st Innings Summary-------------------------------------------')

    print('\n')

    print('\nTotal-', first\_ing\_total, '\nwickets -', first\_ing\_wickets,

          '\novers -', int((first\_ing\_balls-1)/6), '.', (first\_ing\_balls-1) % 6, '\nballs', (first\_ing\_balls-1))

    # print('Extras - ',extras\_first\_ing)

    print('\nLast dismissal', last\_dismissal)

    print('\n\n------------------------------------------------1st Innings Scorecard-------------------------------------------')

    new\_headers = ['Batting', 'Runs', 'Balls Faced',

                   'MOD', 'Bowler', 'Batting No']

    df\_score\_card\_first\_ing.columns = new\_headers

    df\_score\_card\_first\_ing\_without\_index = df\_score\_card\_first\_ing.set\_index(

        'Batting')

    print('\n')

    print(df\_score\_card\_first\_ing\_without\_index)

    # convert match summary to a dataframe

    overs = str(int((first\_ing\_balls-1)/6)) + \

        '.' + str((first\_ing\_balls-1) % 6)

    first\_ing\_summary = [

        [first\_ing\_total, first\_ing\_wickets, overs, (first\_ing\_balls-1)]]

    df\_first\_ing\_summary = pd.DataFrame(first\_ing\_summary, columns=[

                                        'Total', 'Wickets', 'Overs', 'Balls'])

    print('\n\n---------------------------------------------1st Innings Bowling figures------------------------------------------')

    new\_headers = ['Bowling', 'Overs', 'Runs', 'Wickets', 'Economy']

    df\_bowler\_list\_first\_ing.columns = new\_headers

    df\_bowler\_list\_first\_ing\_without\_index = df\_bowler\_list\_first\_ing.set\_index(

        'Bowling')

    print('\n')

    print(df\_bowler\_list\_first\_ing\_without\_index)

    print('\n\n')

    # ----------------------------------------------------------------------Write data to excel file by creating Excel Writer Object from Pandas-------------------------------------------------------------------------

    filename\_match = str(visiting\_team[0]) + '\_vs\_' + str(home\_team[0])

    match\_file\_path = r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches\\' + filename\_match + '.xlsx'

    writer = pd.ExcelWriter(match\_file\_path, engine='xlsxwriter')

    workbook = writer.book

    worksheet = workbook.add\_worksheet('Match Summary')

    writer.sheets['Match Summary'] = worksheet

    df\_score\_card\_first\_ing.to\_excel(

        writer, sheet\_name='Match Summary', startrow=0, startcol=0, index=False)

    df\_first\_ing\_summary.to\_excel(

        writer, sheet\_name='Match Summary', startrow=14, startcol=0, index=False)

    df\_bowler\_list\_first\_ing.to\_excel(

        writer, sheet\_name='Match Summary', startrow=19, startcol=0, index=False)

    writer.save()

    writer.close()

    print('\n\n')

    # --------------------------------------------------------------------------------------------Update player standings------------------------------------------------------------------------------------------------

    player\_standings(score\_card\_first\_ing, bowler\_list\_first\_ing)

def second\_innings():

    global second\_ing\_total

    global second\_ing\_wickets

    global df\_score\_card\_second\_ing\_without\_index

    global df\_bowler\_list\_second\_ing\_without\_index

    global graph\_second\_ing\_balls

    global graph\_second\_ing\_total

    global graph\_second\_ing\_fow\_balls

    global graph\_second\_ing\_fow\_total

    graph\_second\_ing\_balls = []

    graph\_second\_ing\_total = []

    graph\_second\_ing\_fow\_balls = []

    graph\_second\_ing\_fow\_total = []

    second\_ing\_total = 0

    second\_ing\_wickets = 0

    second\_ing\_balls = 1

    score\_card\_second\_ing = []

    batsman\_onstrike = [['name', 0, 0], True]

    batsman\_offstrike = [['name', 0, 0], False]

    bowler\_onstrike = []

    # batsman\_name , runs , balls\_faced , method of dismissal , bowler

    # importing batting team

    batting\_url = team\_to\_bowl[1]

    batting\_team = pd.read\_excel(batting\_url)

    # converting excel to python list

    yet\_to\_bat = batting\_team.values.tolist()

    # bowlers\_name , second\_ing\_balls , runs , wickets

    # importing bowling team

    bowling\_url = team\_to\_bat[1]

    bowling\_team = pd.read\_excel(bowling\_url)

    # converting excel to python list

    bowling\_team\_list = bowling\_team.values.tolist()

    yet\_to\_bowl = []

    for i in reversed(range(len(bowling\_team\_list))):

        if len(yet\_to\_bowl) < 5:

            yet\_to\_bowl.append([bowling\_team\_list[i][0], 0, 0, 0, 0])

    dismissed\_batsmen = []

    batsman\_list = []

    # method of dismissal

    method\_of\_dismissal = ['Bowled', 'Caught', 'LBW']

    bowler\_score = 0  # score counting variable for bowler

    batter\_score = 0  # score counting variable for batsman

    # opening batsmen coming to the field

    batsman\_onstrike[0] = yet\_to\_bat.pop(0)

    batsman\_offstrike[0] = yet\_to\_bat.pop(0)

    # opening bowler

    bowler\_onstrike = yet\_to\_bowl.pop(0)

    while (second\_ing\_balls < (TOTAL\_BALLS+1)):

        if ((second\_ing\_wickets == TOTAL\_WICKETS) or (second\_ing\_total > first\_ing\_total)):

            break

        else:

            if ((second\_ing\_balls-1) > 0 and (second\_ing\_balls-1) % 6 == 0) and (len(yet\_to\_bowl)) > 0:

                yet\_to\_bowl.append(bowler\_onstrike)

                bowler\_onstrike = yet\_to\_bowl.pop(0)

            # get random scores for bowler and batsman

            bowler\_score = random.randint(1, 6)

            batter\_score = random.randint(0, 6)

            if bowler\_score == batter\_score:

                # adding wickets to bowler

                current\_bowler\_onstrike\_second\_ing\_wickets = 0

                current\_bowler\_onstrike\_second\_ing\_wickets = bowler\_onstrike[3]

                bowler\_onstrike[3] = current\_bowler\_onstrike\_second\_ing\_wickets + 1

                # adding second\_ing\_balls to batsman

                current\_batsman\_onstrike\_balls = batsman\_onstrike[0][2]

                batsman\_onstrike[0][2] = current\_batsman\_onstrike\_balls + 1

                # moving the dismissed\_batsmen to dismissed\_batsmen array

                dismissed\_batsmen.append(batsman\_onstrike[0])

                # adding method of dismissal to batsman

                current\_batsman\_method\_of\_dismissal = random.choices(

                    method\_of\_dismissal)

                batsman\_onstrike[0][3] = current\_batsman\_method\_of\_dismissal[0]

                # adding dismissed bowler name to batsman

                current\_batsman\_bowler\_dismissed = batsman\_onstrike[0][4]

                batsman\_onstrike[0][4] = current\_batsman\_bowler\_dismissed + \

                    str(bowler\_onstrike[0])

                # fall of wickets

                print('FOW', second\_ing\_total, ' --> ', second\_ing\_wickets+1,

                      ' on over -', int(second\_ing\_balls/6), '.', (second\_ing\_balls) % 6, batsman\_onstrike[0][0])

                # appendind the FOW data to the graph

                graph\_second\_ing\_fow\_balls.append(second\_ing\_balls)

                graph\_second\_ing\_fow\_total.append(second\_ing\_total)

                # bring new batsman to the crease (batsman\_onstrike)

                if len(yet\_to\_bat) > 0:

                    batsman\_onstrike[0] = []

                    batsman\_onstrike[0] = yet\_to\_bat.pop(0)

                # out - add wicket to wickets

                second\_ing\_wickets += 1

            else:

                # adding batter\_score to current\_batsman

                current\_batsman\_onstrike\_score = 0

                current\_batsman\_onstrike\_score = batsman\_onstrike[0][1]

                batsman\_onstrike[0][1] = current\_batsman\_onstrike\_score + batter\_score

                # adding second\_ing\_balls to current\_batsman

                current\_batsman\_onstrike\_balls = batsman\_onstrike[0][2]

                batsman\_onstrike[0][2] = current\_batsman\_onstrike\_balls + 1

                # adding batter\_score to current\_bowler

                current\_bowler\_onstrike\_runs = 0

                current\_bowler\_onstrike\_runs = bowler\_onstrike[2]

                bowler\_onstrike[2] = current\_bowler\_onstrike\_runs + \

                    batter\_score

                # swapping onstrike batsman when strike rotates

                if batter\_score == 1 or batter\_score == 3:

                    current\_batsman = batsman\_onstrike[0]

                    # swapping onstrike batsman

                    batsman\_onstrike[0] = batsman\_offstrike[0]

                    batsman\_offstrike[0] = current\_batsman

                else:

                    pass  # when batter\_score is not swapping

                # add batter score to second\_ing\_total

                second\_ing\_total += batter\_score

        # adding second\_ing\_balls to bowler

        current\_bowler\_onstrike\_balls = 0

        current\_bowler\_onstrike\_balls = bowler\_onstrike[1]

        bowler\_onstrike[1] = current\_bowler\_onstrike\_balls + 1

        second\_ing\_balls += 1

        # adding second\_ing\_total to graph\_second\_ing\_total

        graph\_second\_ing\_total.append(second\_ing\_total)

    # assinging second innings balls to graph

    graph\_second\_ing\_balls = range(1, second\_ing\_balls)

    # plotting balls and total graph

    plt.plot(graph\_first\_ing\_balls, graph\_first\_ing\_total)

    plt.plot(graph\_second\_ing\_balls, graph\_second\_ing\_total)

    # plotting fow graph

    plt.plot(graph\_first\_ing\_fow\_balls, graph\_first\_ing\_fow\_total, linestyle='', linewidth=3,

             marker='o', markerfacecolor='green', markersize=6)

    plt.plot(graph\_second\_ing\_fow\_balls, graph\_second\_ing\_fow\_total, linestyle='', linewidth=3,

             marker='o', markerfacecolor='red', markersize=6)

    # add legend

    plt.legend([team\_to\_bat[0], team\_to\_bowl[0],

               team\_to\_bat[0]+' Wickets', team\_to\_bowl[0]+' Wickets'])

    # giving a title to the graph

    plt.title('--Innings Progression Graph--')

    # naming the x axis

    plt.xlabel('Balls')

    # naming the y axis

    plt.ylabel('Runs')

    # saving the graph into an image

    plt.savefig(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\myplot.png', format='png')

    # clearing the plt image to avoid overwriting

    plt.clf()

    # last dismissed batsman

    last\_dismissal = dismissed\_batsmen[-1]

    # add dismissed\_batsmen to batsman\_list

    batsman\_list = dismissed\_batsmen

    # add each batsman in yet\_to\_bat to batsman\_list array for displaying purposes

    if len(yet\_to\_bat) > 0:

        for i in range(len(yet\_to\_bat)):

            batsman\_list.append(yet\_to\_bat[i])

    # add on and off strike batsmen to batsman\_list

    if second\_ing\_wickets != TOTAL\_WICKETS:

        batsman\_onstrike[0][3] = '\* NOT OUT'

        batsman\_list.append(batsman\_onstrike[0])

    batsman\_offstrike[0][3] = 'NOT OUT'

    batsman\_list.append(batsman\_offstrike[0])

    # add batsman\_list to score\_card\_second\_ing

    score\_card\_second\_ing = batsman\_list

    # sort score\_card\_second\_ing to the original batting order

    sorted\_list = sorted(score\_card\_second\_ing, key=itemgetter(5))

    # add bowlers to bowler\_list\_second\_ing

    bowler\_list\_second\_ing = yet\_to\_bowl

    bowler\_list\_second\_ing.append(bowler\_onstrike)

    # convert df\_score\_card\_second\_ing to a data frame for displaying

    df\_score\_card\_second\_ing = pd.DataFrame(sorted\_list)

    # converting bowler second\_ing\_balls to overs

    for bowler\_overs\_second\_ing in bowler\_list\_second\_ing:

        bowler\_overs\_second\_ing[1] = str(

            int((bowler\_overs\_second\_ing[1])/6)) + '.' + str((bowler\_overs\_second\_ing[1]) % 6)

    # adding the economy for bowler

    for bowler\_economy\_second\_ing in bowler\_list\_second\_ing:

        if not(float(bowler\_economy\_second\_ing[1]) > 0):

            raise Exception('BOWLER ECONOMY NOT FOUND!!!!!!!!!!!')

        bowler\_economy\_second\_ing[4] = round(

            bowler\_economy\_second\_ing[2]/float(bowler\_economy\_second\_ing[1]), 2)

    # convert df\_bowler\_list\_second\_ing to a data frame for displaying

    df\_bowler\_list\_second\_ing = pd.DataFrame(bowler\_list\_second\_ing)

    print('\n\n------------------------------------------------------2nd Innings Summary---------------------------------')

    print('\nTotal-', second\_ing\_total, '\nwickets -', second\_ing\_wickets,

          '\novers -', int((second\_ing\_balls-1)/6), '.', (second\_ing\_balls-1) % 6, '\nballs', (second\_ing\_balls-1))

    # print('Extras',extras\_second\_ing)

    print('\nLast dismissal', last\_dismissal)

    print('\n\n-----------------------------------------------------2nd Innings Scorecard---------------------------------')

    new\_headers = ['Batting', 'Runs', 'Balls Faced',

                   'MOD', 'Bowler', 'Batting No']

    df\_score\_card\_second\_ing.columns = new\_headers

    df\_score\_card\_second\_ing\_without\_index = df\_score\_card\_second\_ing.set\_index(

        'Batting')

    print('\n')

    print(df\_score\_card\_second\_ing\_without\_index)

    # convert match summary to a dataframe

    overs = str(int((second\_ing\_balls-1)/6)) + \

        '.' + str((second\_ing\_balls-1) % 6)

    second\_ing\_summary = [

        [second\_ing\_total, second\_ing\_wickets, overs, (second\_ing\_balls-1)]]

    df\_second\_ing\_summary = pd.DataFrame(second\_ing\_summary, columns=[

        'Total', 'Wickets', 'Overs', 'Balls'])

    print('\n\n--------------------------------------------------2nd Innings Bowling figures------------------------------')

    new\_headers = ['Bowling', 'Overs', 'Runs', 'Wickets', 'Economy']

    df\_bowler\_list\_second\_ing.columns = new\_headers

    df\_bowler\_list\_second\_ing\_without\_index = df\_bowler\_list\_second\_ing.set\_index(

        'Bowling')

    print('\n')

    print(df\_bowler\_list\_second\_ing\_without\_index)

    print('\n\n')

    # -----------------------------------------------------------------------Write data to excel file by creating Excel Writer Object from Pandas-----------------------------------------------------------------------

    book = load\_workbook(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches\\' + filename\_match + '.xlsx')

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches\\' + filename\_match + '.xlsx', engine='openpyxl')

    writer.book = book

    writer.sheets = dict((ws.title, ws) for ws in book.worksheets)

    df\_score\_card\_second\_ing.to\_excel(

        writer, sheet\_name='Match Summary', startrow=0, startcol=9, index=False)

    df\_second\_ing\_summary.to\_excel(

        writer, sheet\_name='Match Summary', startrow=14, startcol=9, index=False)

    df\_bowler\_list\_second\_ing.to\_excel(

        writer, sheet\_name='Match Summary', startrow=19, startcol=9, index=False)

    worksheet = book.worksheets[0]

    img = openpyxl.drawing.image.Image(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\myplot.png')

    img.anchor = 'D30'

    worksheet.add\_image(img)

    writer.save()

    writer.close()

    # --------------------------------------------------------------------------------------------Update player standings--------------------------------------------------------------------------------------------

    player\_standings(score\_card\_second\_ing, bowler\_list\_second\_ing)

def update\_points\_table(winning\_team, losing\_team, is\_drawn):

    update\_points\_table = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\points\_table.xlsx')

    df\_update\_points\_table = pd.DataFrame(update\_points\_table)

    if is\_drawn == 0:

        if(winning\_team in Group\_A):

            group = "Group A"

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            current\_wins\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Won']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Won'] = current\_wins\_count+1

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == losing\_team[0]].tolist(

            )

            current\_loss\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Lost']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Lost'] = current\_loss\_count+1

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            current\_points\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Points']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Points'] = current\_points\_count+2

        elif(winning\_team in Group\_B):

            group = "Group B"

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            current\_wins\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Won\_B']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Won\_B'] = current\_wins\_count+1

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == losing\_team[0]].tolist(

            )

            current\_loss\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Lost\_B']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Lost\_B'] = current\_loss\_count+1

            find\_points\_table\_index = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            current\_points\_count = df\_update\_points\_table.at[find\_points\_table\_index[0], 'Points\_B']

            df\_update\_points\_table.at[find\_points\_table\_index,

                                      'Points\_B'] = current\_points\_count+2

    elif is\_drawn == 1:

        if(winning\_team in Group\_A):

            group = "Group A"

            find\_points\_table\_index\_1 = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            find\_points\_table\_index\_2 = df\_update\_points\_table.index[df\_update\_points\_table[group] == losing\_team[0]].tolist(

            )

            current\_points\_count\_1 = df\_update\_points\_table.at[find\_points\_table\_index\_1[0], 'Points']

            df\_update\_points\_table.at[find\_points\_table\_index\_1,

                                      'Points'] = current\_points\_count\_1+1

            current\_points\_count\_2 = df\_update\_points\_table.at[find\_points\_table\_index\_2[0], 'Points']

            df\_update\_points\_table.at[find\_points\_table\_index\_2,

                                      'Points'] = current\_points\_count\_2+1

        elif(winning\_team in Group\_B):

            group = "Group B"

            find\_points\_table\_index\_1 = df\_update\_points\_table.index[df\_update\_points\_table[group] == winning\_team[0]].tolist(

            )

            find\_points\_table\_index\_2 = df\_update\_points\_table.index[df\_update\_points\_table[group] == losing\_team[0]].tolist(

            )

            current\_points\_count\_1 = df\_update\_points\_table.at[find\_points\_table\_index\_1[0], 'Points']

            df\_update\_points\_table.at[find\_points\_table\_index\_1,

                                      'Points'] = current\_points\_count\_1+1

            current\_points\_count\_2 = df\_update\_points\_table.at[find\_points\_table\_index\_2[0], 'Points']

            df\_update\_points\_table.at[find\_points\_table\_index\_2,

                                      'Points'] = current\_points\_count\_2+1

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\points\_table.xlsx', engine='xlsxwriter')

    workbook = writer.book

    worksheet = workbook.add\_worksheet('Sheet1')

    writer.sheets['Sheet1'] = worksheet

    df\_update\_points\_table.to\_excel(

        writer, sheet\_name='Sheet1', startrow=0, startcol=0, index=False)

    writer.save()

    writer.close()

def display\_points\_table():

    update\_points\_table = pd.read\_excel(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\points\_table.xlsx')

    df\_update\_points\_table = pd.DataFrame(update\_points\_table)

    print('\n')

    print(df\_update\_points\_table.to\_string(index=False))

def match\_summary():

    # Toss

    print('\n\n---------------------------------------------------------Match Summary------------------------------------------\n')

    match\_toss = ''

    match\_result = ''

    if selection == toss:

        match\_toss = str(visiting\_team[0].replace(

            '\_', ' ')+' Won the toss and chose to ' + choose)

        print(match\_toss)

        print('\n')

    else:

        match\_toss = str(home\_team[0].replace(

            '\_', ' ')+' Won the toss and chose to ' + choose)

        print(match\_toss)

        print('\n')

    print('------------------------------------------------------First Innings Top Performers---------------------------------\n\n')

    print(team\_to\_bat[0].replace('\_', ' '))

    print(df\_score\_card\_first\_ing\_without\_index.nlargest(4, 'Runs'))

    print('\n', team\_to\_bowl[0].replace('\_', ' '))

    print(df\_bowler\_list\_first\_ing\_without\_index.nlargest(3, 'Wickets'))

    print('\n\nTotal', first\_ing\_total, '/', first\_ing\_wickets)

    print('\n')

    print('------------------------------------------------------Second Innings Top Performers---------------------------------\n\n')

    print(team\_to\_bowl[0].replace('\_', ' '))

    print(df\_score\_card\_second\_ing\_without\_index.nlargest(4, 'Runs'))

    print('\n', team\_to\_bat[0].replace('\_', ' '))

    print(df\_bowler\_list\_second\_ing\_without\_index.nlargest(3, 'Wickets'))

    print('\n\nTarget', first\_ing\_total+1)

    print('Total', second\_ing\_total, '/', second\_ing\_wickets)

    print('\n\n')

    # Match result

    global winning\_team

    global losing\_team

    is\_drawn = 0

    print('\n------------------------------------------------------------Match Result----------------------------------------------\n')

    if (second\_ing\_total > first\_ing\_total):

        match\_result = str(team\_to\_bowl[0].replace('\_', ' ')+' Won by ' +

                           str(TOTAL\_WICKETS-second\_ing\_wickets)+' wickets')

        print(match\_result)

        winning\_team = team\_to\_bowl

        losing\_team = team\_to\_bat

    elif (second\_ing\_total < first\_ing\_total):

        match\_result = str(team\_to\_bat[0].replace(

            '\_', ' ')+' Won by ' + str(first\_ing\_total-second\_ing\_total)+' runs')

        print(match\_result)

        winning\_team = team\_to\_bat

        losing\_team = team\_to\_bowl

    elif (second\_ing\_total == first\_ing\_total):

        winning\_team = team\_to\_bat

        losing\_team = team\_to\_bowl

        is\_drawn = 1

        print('\n\n-------------------------------------------------------Match drawn-----------------------------------------------\n')

    book = load\_workbook(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches\\' + filename\_match + '.xlsx')

    writer = pd.ExcelWriter(

        r'E:\\IIT\\1st Year\\1st Trimester\\CM1601 [PRO]  Programming Fundamentals\\Course Work\\tournament\\matches\\' + filename\_match + '.xlsx', engine='openpyxl')

    writer.book = book

    worksheet = book.worksheets[0]

    worksheet['F27'] = match\_toss

    worksheet['F28'] = match\_result

    writer.save()

    writer.close()

    print('\n---------------------------------------------------------------------------------------------------------------------\n')

    # --------------------------------------------------------------------------------------------------Update Points table---------------------------------------------------------------------------------------------

    update\_points\_table(winning\_team, losing\_team, is\_drawn)

Coursework.py

# ----------------------------------------------------------------------------------------------------Importing Modules-----------------------------------------------------------------------------------------------

import cricket

from openpyxl import load\_workbook

import random

import pandas as pd

from operator import itemgetter

# ---------------------------------------------------------------------------------Global variables which are used when accessing the functions------------------------------------------------------------------------

user\_input = ''

global\_exit = ''

winning\_team = []

losing\_team = []

print("\n\n-------------------------------------------------------------------------------Welcome to IIT Cricket Premier League 2021-----------------------------------------------------------------------------------")

try:

    while user\_input != 'x':

        user\_input = input(

            "\n\nPress the desired number for your action... \n\nPlay a new match - 1 \nView/edit team/player profile - 2 \nView Player Standings - 3 \nView Tournament Standings - 4 \nPress 'x' to exit...  ")

        if user\_input == '1':

            cricket.generate\_random\_match()

            cricket.points\_table()

            cricket.toss\_coin()

            cricket.first\_innings()

            cricket.second\_innings()

            cricket.match\_summary()

        elif user\_input == '2':

            cricket.team\_profile\_edit(user\_input)

        elif user\_input == '3':

            cricket.display\_player\_standings()

        elif user\_input == '4':

            cricket.display\_points\_table()

        elif user\_input == 'x':

            break

        else:

            print('Input value incorrect \nTry again!!!\n')

except IndexError as e:

    print('\n-------------------------------------------------------------------------------------------TOURNAMENT OVER-------------------------------------------------------------------------------------------\n')

except Exception as e:

    print(e)

print("\n\n-----------------------------------------------------------------------------------------Thank you for playing!!!----------------------------------------------------------------------------------------")

Output

Main Menu

Text

Description automatically generated

Editing menu

Text

Description automatically generated

Text

Description automatically generated



Text

Description automatically generated

Text

Description automatically generatedViewing Tournament Standing before the tournament begins

Playing a match

Text

Description automatically generated

First Innings

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Second Innings

Text

Description automatically generated

Text

Description automatically generated

Match Summary

A picture containing text

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

Viewing Tournament Standing after playing a match

Text

Description automatically generated

Viewing Player Standing after playing a match

Text

Description automatically generated

Exiting from the main menu

A black screen with white text

Description automatically generated with low confidence

Match Summary example

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Chart, line chart

Description automatically generated

Test Plan

01)Generating two teams from the two groups randomly and picking one match

* Input :

match\_between\_A = random.sample(Group\_A, 2)

match\_between\_B = random.sample(Group\_B, 2)

chosen\_match = [match\_between\_A,match\_between\_B]

match\_between = random.choice(chosen\_match)

print(match\_between)

* Expected output:

Ex: Punjab Pakistan vs Kolkata England

* Actual output:

Text

Description automatically generated

02)Assigning the two picked teams from the match as Home team and Visiting team

* Input :

coin = ["heads", "tails"]

options = ['bat','bowl']

visiting\_team = random.choice(match\_between)

if visiting\_team in match\_between:

    match\_between.remove(visiting\_team)

home\_team = match\_between[0]

print('Home Team - ', home\_team)

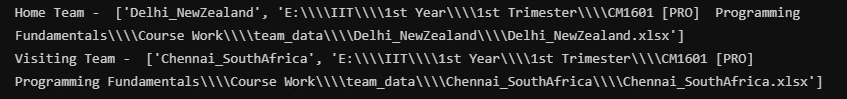
print('Visiting Team - ', visiting\_team)

* Expected output:

Home team – Delhi New Zealand

Visiting team – Chennai South Africa

* Actual output:



03)Displaying batting card without index column

* Input :

#add batsman\_list to score\_card\_first\_ing

score\_card\_first\_ing = batsman\_list

# sort score\_card\_first\_ing to the original batting order

sorted\_list = sorted(score\_card\_first\_ing, key=itemgetter(5))

# convert score\_card\_first\_ing to a data frame for displaying

df\_score\_card\_first\_ing = pd.DataFrame(sorted\_list)

df\_score\_card\_first\_

* Expected output:

A picture containing text, monitor, screen, cellphone

Description automatically generated

* Actual output:

A picture containing text, road, cellphone, monitor

Description automatically generated

04)Showing a summary of an innings’ top performers

* Input :

print(team\_to\_bat[0].replace('\_', ' '))

print(df\_score\_card\_first\_ing\_without\_index.nlargest(11, 'Runs'))

print('\n',team\_to\_bowl[0].replace('\_', ' '))

print(df\_bowler\_list\_first\_ing\_without\_index.nlargest(5, 'Wickets'))

print('\nTotal',first\_ing\_total,'/',first\_ing\_wickets)

* Expected output:

Graphical user interface, text

Description automatically generated

* Actual output:

Text

Description automatically generated

05)Converting the random match generation code into a function

* Input :

def generate\_random\_match():

    match\_between\_A = random.sample(Group\_A, 2)

    match\_between\_B = random.sample(Group\_B, 2)

    chosen\_match = [match\_between\_A, match\_between\_B]

    match\_between = random.choice(chosen\_match)

print(match\_between)

* Expected output:

Ex: Punjab Pakistan vs Kolkata England

* Actual output:

