Tennis Tournament Simulation

Planned Data Structures and Justification

Name of data structure	Type of data structure
player	class object
maleLeaderBoard	list
femaleLeaderBoard	list
tournamentNames	list
doesNotAdvancedToNextRound	list
advancedToNextRound	list
winner	list

The data structures I have planned to use for the tennis tournament are found in the above table. I have planned use a combination of class object and list to store my data.

I decided to use a class object as this is the ideal way to store numerous different types of data of a player e.g. name - String, ranking- float and prizemoney – integer. By using a class object this will give me the ability to call individual instances of each data type. The class object also gives me the ability to call one of the methods inside that class. Another benefit of using a class object is the encapsulation offers durability and adaptability, as implementation of parts of the program to change without negatively affecting other parts. This enables the program to be easier to fix bugs or add new functionality with relatively local changes to a component.

I plan to use many lists within my tennis tournament as they can contain any variable type and contain as many variables as desired. The data I plan on storing in my list are the class object player this will be ideal as list offer simplicity when iterated over, this combination of list and object will give me the ability to iterate over any types of instance of the object. List also offer many inbuilt methods which will add to the ease of appending, removing, indexing and sorting.

Pseudocode – Ranking Points

Ranking function (Pass in arguments of the tournament and position)
placePoints – float
difficultly – float
totalPoints – float
Read ranking points CSV file
    For row in CSV in read file
        If finished equals row place then
            placePoints = row in CSV 'Tournament Ranking Points'
        if tournament equals row in CSV 'Tournament' then
            difficultly = row in CSV 'Difficulty'
totalPoints = (placePoints * difficultly)
return totalPoints

Pseudocode – View ranking Points

viewRankings()
    sort leaderBoard(player ranking, descending)
    print(ranking points table)
    for player in leaderBoard
        print(player name, player ranking)

Implementation

```
def rankings(finished, tournament):
    place = finished
    difficultly = 0.0
    placePoints = 0.0
    totalPoints = 0.0
    with open('RANKING POINTS.csv', newline='') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            if place == row['Place']:
                placePoints = row['Tournament Ranking  Points']
                placePoints = float(placePoints)
            if tournament == row['Tournament']:
                difficultly = row['Difficulty']
                difficultly = float(difficultly)
    totalPoints = (placePoints * difficultly)
    totalPoints = round(totalPoints, 2)
    return totalPoints
```

```
def womensRankings():
    """Prints and sorts the prize money"""
    positon = 1
    print("Womens Ranking table")
    femaleLeaderBoard.sort(key = lambda player: player._ranking, reverse=True)
    for player in femaleLeaderBoard:
        print(positon ,player._name, player._ranking)
        positon = positon + 1
    menu()
```

Implementation Solution for Ranking Players According to Points Earned

The python code shown above in combination the tennisRounds/manual input functions and view rankings, shows the implementation in of the rankings algorithm. The rankings function was close to my pseudocode but was extended in code as I have over looked a couple of issues.

The ranking function works by taking in the position of the player (calculated in tennisRound function or manual input) and the tournament (passed through from a menu). It then loops through the column 'Place' to see if the equals the position, if it does then it changes column to 'Tournament Ranking Points' column and assigns that to placePoints. This is a String of the ranking points which then gets parsed to a float. Next the function checks the tournament (which was passed through) equals a 'Tournament' from the tournament column in the CSV, if so it assigns that tournaments difficulty to the variable difficulty. placePoints is then multiplied by difficulty and assigned to totalpoints. Lastly totalpoints is rounded to two decimal places and then returned to tennisRound.

To sort the Leader boards in according to ranking points earned, the view rankings function, which was split into men's and women's rankings, I have used the inbuilt sort algorithm.

In evaluation of the ranking players by their points earned I feel my algorithm is more than efficient for the task in hand. As tennis tournaments never get larger than a few thousand, at most, it is more than acceptable to search through the CSV file or list of players sequentially. Had the data set been larger I believe that a binary search would be appropriate. In terms of my sorting algorithm which is in fact the inbuilt python sort, I have researched that the time complexity is O(n log n) which efficient enough for sorting a list of tennis players.

Pseudocode – Prize Money

prizeMoneyFromTournament(Pass in arguments of the tournament and position):
  prizemoney - integer
  Read prize money CSV file
    For row in CSV in read file
      If position equals row in CSV 'Place' AND tournament equals row in CSV 'Tournament'
      prizemoney = row in CSV 'Prize Money ($)'
  return prizemoney

Pseudocode – View Prize Money

viewPrizeMoney()
    sort leaderBoard(player prize money, descending)
    print(prize money table)
    for player in leaderBoard
      print(player name, player prize money)

Implementation of Prize Money

```
def prizeMoneyFromTournament(finished, tournament):
    place = finished
    prizemoney = 0
    with open('PRIZE MONEY.csv', newline='') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            if place == row['Place'] and tournament == row['Tournament']:
                prizemoney = row['Prize Money ($)']
                prizemoney = int(prizemoney)
    return prizemoney


def womensPrizeMoneyRankings():
    """Prints and sorts the prize money"""
    positon = 1
    print("womens Prize Money Rankings table")
    femaleLeaderBoard.sort(key = lambda player: player._prizeMoney, reverse=True)
    for player in femaleLeaderBoard:
        print(positon ,player._name, player._ranking)
        positon = positon + 1
    menu()
```

Round Function Used in conjunction with ranking function and prizemoney function

```
roundCSV function ()
doesNotAdvancedToNextRound empty list
advancedToNextRound empty list
winners empty list
position – integer
roundPoints – float
prizemoney - int
Read ranking points CSV file
        For row in CSV in file
                Print row in CSV Player a , Score Player A , Player B, Score Player B
                Increment position
                If Score Player A < Score Player B then
                Add Player A to doesNotAdvancedToNextRound
                Add Player B to advancedToNextRound
                Else
                Add Player B to doesNotAdvancedToNextRound
                Add Player A to advancedToNextRound
        For i in  doesNotAdvancedToNextRound
                roundpoints = call function ranking (pass in tournament and position)
                prizeMoney = call function  prizeMoney(pass in tournament and position)
                if male tournament then
                        for player in maleLaderboard
                                if i equals player.name then
                                player.ranking += roundpoints
                                player.prizeMoney += prizeMoney
                if female tournament then
                        for player in femaleLaderboard
                                if i equals player.name then
                                player.ranking += roundpoints
                                player.prizeMoney += prizeMoney
```

Implementation solution for tennisRound Function conjunction with ranking function and prizemoney function

```
def tennisRounds(tournament,mensOrWomens):
    winner = []
    TournamentRound = 1
    TournamentRound = str(TournamentRound)
    sizeOfroundsForLoop = sizeOfTournament(mensOrWomens)
    numberOfplayers = 0
    doesNotAdvancedToNextRound = []
    for x in range(0, sizeOfroundsForLoop):
        print("--------------------------------------------------------------------------------")
        print(tournament + " ROUND " + str(TournamentRound) )
        print("--------------------------------------------------------------------------------")
        roundpoints = 0
        prizeMoney = 0
        finished = 0
        TournamentRound = str(TournamentRound)
        with open(tournament+' ROUND '+TournamentRound+' '+mensOrWomens+'.csv', newline='') as csvfile:
            TournamentRound = int(TournamentRound)
            TournamentRound = TournamentRound + 1
            reader = csv.DictReader(csvfile)
            doesNotAdvancedToNextRound = []
            for row in reader:
                print(row['Player A'], row['Score Player A'], row['Player B'], row['Score Player B'])
                finished = finished + 1
                if TournamentRound == 6:
                    if (row['Score Player A'] > row['Score Player B']):
                        winner.append(row['Player A'])
                    else:
                        winner.append(row['Player B'])
                if (row['Score Player A'] < row['Score Player B']):
                    doesNotAdvancedToNextRound.append(row['Player A'])
                else:
                    doesNotAdvancedToNextRound.append(row['Player B'])
            print("\n")
            for i in doesNotAdvancedToNextRound: # i stands for player A or B
                finished = int(finished)
                finished = finished + 1
                finished = str(finished)
                roundpoints =  rankings(finished, tournament)
                prizeMoney = prizeMoneyFromTournament(finished, tournament)
                if mensOrWomens == 'MEN':
                    for player in maleLeaderBoard:
                        if i == player._name:
                            player._ranking += roundpoints
                            player._prizeMoney += prizeMoney
                if mensOrWomens == 'WOMEN':
                   for player in femaleLeaderBoard:
                       if i == player._name:
                            player._ranking += roundpoints
                            player._prizeMoney += prizeMoney
                for i in winner: # i stands for player player A or B
                    finished = int(finished)
                    finished = finished - 1
                    finished = str(finished)
                    roundpoints =  rankings(finished,tournament)
                    prizeMoney = prizeMoneyFromTournament(finished, tournament)
                    if mensOrWomens == 'MEN':
                        for player in maleLeaderBoard:
                            if i == player._name:
                                player._ranking += roundpoints
                                player._prizeMoney += prizeMoney
                    if mensOrWomens == 'WOMEN':
                        for player in femaleLeaderBoard:
                            if i == player._name:
                                    player._ranking += roundpoints
                                    player. prizeMoney += prizeMoney
```

CSV and manual input

To enter matches for the tennis tournaments I have approached two methods.

The first method I implemented was reading in a CSV file and processing matches. I tackled this by creating a function for each round, in each tournament, for each gender. This left me with a large amount of functions that all did the same process. I then improved upon this by making a function handle all rounds, for both genders. This significantly decreased the amount of functions I had used, to one. This was done by passing the tournament name and gender into the method and then using this I could fill in the string that made up the CSV file name. I then need to calculate the number of rounds a tournament would have. This was done by a function which I called sizeOfTournament which took the number of players, men's or women's, then divide this by 2, incrementing number of rounds till number of players equalled one. This was then put into a for loop with the range of 0 to sizeOfTournament. This gave my tournament the ability to scaled up, if it fit the same tournament layout e.g. $\mathbf{2}^{n}$

The second method I implemented was manual input to process matches. I used the experience that I had from creating the CSV input to assist my construction of this method. I used the same menu system to pass the tournament name and gender and the same sizeOfTournament function. I then needed to figure out how to create the matches for the user to fill in. I used a for loop to calculate half the size of the players in that round, this was then used to split the players into two lists, leftsideOfMatch and rightsideOfMatch. I then used a for loop with a zip to loop through both simultaneously. I then had to take input from the console for both players and check that the scores were integers, correct number of sets and not a draw. This was achieved by a while loop with several conditional statements inside. If the if statements were true the user would have to enter the scores in again, if none were triggered the else statement would break the while loop, meaning that the input was correct.

In justification in how the program handle the task and in the input methods I believe my tennis tournament application has achieved the specification. It allows both manual and CSV input, checks all inputs by user, stores and sorts both ranking points and prize money and offers the ability to save ranking points and prize money leader boards to continue at a later date. In evaluation of the program, I do believe that certain aspects could have been improved upon that would either; speed up searches through data structures or improve user interaction. In terms of speeding up searches I have used a lot of nested loops which increase exponentially, I have also used for loops to search through CSV rows or list sequentially this could have been vastly improved upon if the list had been sorted first then a binary search tree to quickly grab the correct data. In terms of improving user interaction I could have used a GUI, which would vastly improve the usability. I also could have made an option to save the tournament after each round, both CSV and manual input. This not only would make it easier for the user to input the scores more freely but also give the program a better real world application.