ATP Tournament Simulation Documentation

Overview:

The purpose of this program is to simulate shifts in rankings of the top 128 ranked ATP men's tennis players, starting with the players and their rankings accurate as of November 8th, 2020 from the ATP Official Website. This program allows users to manually change players points (which will adjust their ranking), input a single match result, simulate a Grand Slam Style Tournament, and view the rankings as a website with HTML.

There are two primary data structures used in this file: a dictionary and a list. The dictionary atp_data contains the data imported from the ATP Player CSV. The keys for the dictionary are the names of the players and the values are a list of the form [ranking, points], where ranking and points are both integers. A list called atp_players also exists and is primarily used to iterate through the dictionary.

To simulate the Grand Slam Style Tournament, the brackets are set up as follows: the top ranked player plays the 128th ranked player, the second ranked player plays the 127th ranked player, and so forth. Each player is assigned a "win weight," with win weights ranging from 1-16, with 16 being most likely to win. Players ranked in the top 8 will receive weight 16, players ranked 9-16 will receive weight 15, and so forth. To determine the winner of a matchup, the player's probability of winning is equivalent to their player weight divided by the sum of both players' player weights. For instance, in the matchup between the first seed and the 128th seed, the probability the first seed would win would be 16/17. A random number generator is used to determine the winner based on those probabilities. This process is repeated, simulating round after round, until a winner has been selected.

This project is organized into two different code files: the main.py file and functions.py file. The functions.py file is where the functions that carry out processes to make the simulation work are. Examples of functions in this file are an updateRanking function (used to make sure the ranking values change when point values change), a tourneysim function which simply simulates a single round of a grand slam, and so forth. The main file is where all of the simulations are run. The CSV of player data is imported in the main file as well as the functions that carry out the processes mentioned above. The functions in the main file call functions from the functions file with the parameters already imputed, making it easier for the user to run simulations. For example, the function simluateTournament() in main.py calls functions.roundWinners(atp_data, atp_players) (a function in functions.py). The user has no chance of inputting the wrong parameters when calling simulateTournament() and ensures the right parameters will be used every time. In other words, the

functions defined in main.py are designed to make it easier for users to run the simulation by generally avoiding manual parameter entering.
To use this program, open the file named main.py in the zip file. If you would like to reset the data after simulations, simply restart the shell.

List of Functions:

Functions in main.py (listed alphabetically): (Called by the user)

- `editPlayerData():`
  - Purpose: To edit the points of a player manually and then have the rankings update following the manual change.
  - Parameters: None
  - How it Works: Asks for two inputs: the name of the player and the change in points (enter integers, positive for an increase in points, negative for a decrease in points). After receiving the inputs, it accesses the dictionary storing that player's points and adds the value to the current points. Then, it calls the updateRankings function from functions.py (see below for more information) to update the rankings.
- `getPlayerData():`
  - Purpose: To print the ranking and points of a specific player.
  - Parameters: None
  - How it Works: Asks the user to input the name of the player. Then it uses that input as a key in the atp_data dictionary to access and print the player's current ranking and points.
- `matchResult(points=10):`
  - Purpose: To manually enter a match result between two players and then update their points and rankings accordingly. Calls the matchUpdate function from functions.py (see below) with parameters correctly entered.
  - Parameters:
    - Points: (OPTIONAL): Change the number of points the winner of the match receives; the default point gain for the winner is 10 points.
  - How it Works: Asks the user for three inputs: the first two are the names of the players playing the match and the third is the winner of the match. Then it uses the player names as keys to update the points of each player in the dictionary and then calls the updateRankings function from functions.py to update the rankings.
- `printRankings():`

    - Purpose: To output the rankings (with player names and points) to an HTML file which opens in your default web browser.
    - Parameters: None
    - How it Works: It creates a variable that is a long string; the string contains the HTML code. Then, iteration and string concatenation are used to output the rankings, player names, and points.
- `simulateTournament():`
    - Purpose: Simulate a Grand Slam Style ATP Tournament.
    - Parameters: None
    - How it Works: It calls the roundWinners function from functions.py (see below for more information on that function). To summarize, this function carries out the different rounds of the tournament calling another function tourneysim from functions.py where tourneysim determines all the players that win each round, and roundWinners keeps track of the winners of each round until there is only one player left. That winner is then printed, and the rankings/points of each player are updated.

Functions in functions.py (listed alphabetically): (Called by other functions)

- `assignProbability(dic,lis):`
    - Purpose: To generate weights for each player based on their ranking. These weights are used in the tournament simulation to determine the probability of a player winning a matchup against another player. The player weights are assigned as follows: Rankings 1-8 get a weight of 16, 9-16 a weight of 15, all the way to 121-128 get a weight of 1.
    - Parameters:
        - dic: The dictionary containing the player data (atp_data should be used).
        - lis: The list containing the names of the players (atp_players should be used).
    - How it Works: Using the player list, it iterates through the player dictionary. Using conditional statements, it assigns the player name as a key to a new dictionary with a value being the associate weight mentioned above. It then returns the dictionary.
- `editplayerdata(player, points_change,dic5,lis5):`
    - Purpose: To edit a player's points without entering a match result or simulating a tournament.
    - Parameters:
        - player: The name of the player to edit (entered as a string).
        - points_change: The number of points add to the player's point total
        - dic5: The dictionary containing all player data (atp_data should be used)

        - lis5: The list containing all players (atp_players should be used)
    - How it Works: This function takes the player's name and uses it as a key in the player data dictionary to modify the points. Following the modification, the updateRanking function is used to make sure the rankings reflect the new point totals.
- `getplayerdata(dic1):`
    - Purpose: To print the data from a specific player.
    - Parameters:
        - dic1: The dictionary containing the player data (atp_data should be used).
    - How it Works: It asks for an input of a player name, finds the ranking and points of that player, and then prints them out.
- `htmlRankings(dic1,lis1):`
    - Purpose: To convert atp_data information into an HTML file.
    - Parameters:
        - dic1: The dictionary containing the player data (atp_data should be used).
        - lis1: The list that contains the names of all the players (atp_players should be used).
    - How it Works: It creates a string called htmlRankings that contains the HTML code. It uses iteration to import the data from the dictionary and list into the HTML code.
- `matchUpdate(pointAmt,dic1,lis1):`
    - Purpose: To update player rankings and points after playing a hypothetical match.
    - Parameters:
        - pointAmt: The amount of points the winner has added to their total.
        - dic1: The dictionary containing the player data (atp_data should be used).
        - lis1: The list that contains the names of all the players (atp_players should be used).
    - How it Works: It asks for three inputs: the two players playing the match and the winner. It then adds the points to the winner's points before playing the match and then updates the rankings using the updateRanking function.
- `printResultsHTML(r128,r64,r32,r16,rQF,rSF,rF):`
    - Purpose: Output the string results for each round of a simulated tournament to an HTML file opened by the default browser.
    - Parameters:
        - r128: The list of string results from the first round.
        - r64: The list of string results from the second round.
        - r32: The list of string results from the third round.
        - r16: The list of string results from the fourth round.

        - rQF: The list of string results from the quarterfinals.
        - rSF: The list of string results from the semifinals.
        - rF: The string result from the final (states the winner).
    - How it Works: Sets up HTML code in a string called results. Then, iteration is used to convert the string results to HTML code for each round. Since each round required slightly different formatting, these different iterations are carried out seven times. Finally, the function opens the HTML file in the computer's default browser.
- `roundWinners(dic1,lis1):`
    - Purpose: To simulate the seven rounds of a Grand Slam Tournament, output the results, and update player points and rankings accordingly.
    - Parameters:
        - dic1: The dictionary containing all the player data (atp_data should be used).
        - lis1: The list containing all the player names (atp_player should be used).
    - How it Works: First it creates a copy of the dic1 that will not change when rankings are updated as it is a required parameter of tourneysim. Then, it creates a copy of lis1 so that lis1 is not changed but players can be "popped" from the copy of the list. Then the assignProbabilities function is called, assigning the returned dictionary to a variable named player_probabilities. Then, the tourneysim function is called seven times, with it returning the list of next round players and list of results to two variables, respectively. After tourneysim has been called seven times and a winner has been declared, the function printResultsHTML is called and given the string results lists as parameters.
- `tourneysim(dic2,lis2,points,dic3,lis3,dic4):`
    - Purpose: To simulate a single round of a tournament; meant to be called by the roundWinners function for each round of the tournament until there is a champion.
    - Parameters:
        - dic2: This is for the dictionary containing the player weights (use the dictionary returned from the assignProbability function).
        - lis2: The list of players playing in the round (for the first round it is a copy of the atp_players list; for subsequent rounds it is a list of the winners).
        - points: The number of points each round is worth; the winners will have this number of points added to their points total and their ranking will be updated.
        - dic3: The player dictionary with the ATP player data (atp_data should be used).
        - lis3: The list with all the players (atp_players should be used).

        - dic4: A copy of dic3 that will not be changed during the simulation when rankings and points are updated in dic3.
    - How it Works: Initial matchups are determined by having the 1st seed play the 128th seed, 2nd seed play the 127th seed, and so forth. Therefore, the program pops the player at index zero and player at index -1 in each round for the matchup. From there, the winner is determined by calculating winning probabilities using the player weights (player1's odds of winning are equivalent to player1_weight / (player1_weight + player2_weight)). A random number generator is used to determine the winner. The winner is appended to the nextround list, and a string is completed with the result and appended to the htmlResults list. The function returns both the nextround list and htmlResults list.
- `updateRanking(playername, dic,lis):`
    - Purpose: To update the ranking value of a player after their points change.
    - Parameters:
        - playername: This takes the name of the player to update the ranking for.
        - dic: This takes the dictionary that contains the ranking/points data for the players (atp_data should be used).
        - lis: This takes the list that contains the names of all the players (atp_players should be used).
    - How it Works: First, it saves the old ranking of the player in a variable and then sets a variable for the new ranking equivalent to 1. Then, using the list of players, it iterates through the dictionary to see at what point the player has greater points than the player found in the dictionary. Whenever their points are less, the new ranking increases by 1. Once the player's points are greater than whichever case is being iterated, the new ranking reflects their new numerical ranking. Afterward, their old ranking is used to compare which player's rankings need to move up or down 1 based on the shift in this player. The atp_players list is then updated with this person moving in the list so that the players are organized in order of ranking.