

---

# Fifteen Years of Women's Tennis

The final assignment asks you to use the computational thinking and programming skills you learned in the course to answer an empirical social science question. You are expected to apply the best practices and theoretical concepts we covered in the course to produce a program that not only returns the correct output but is also legible, modular, and reasonably optimized. The assignment assumes mastery of loops, conditionals, and functions, as well as awareness of issues related to runtime performance.

In honor of Emma Raducanu's historical achievements this year, we will study the results of womens' tennis matches over the period 2007-2021. Your objectives are to parse the data, reconstruct tournament brackets, identify the top players, and implement an algorithm to provide an alternative ranking for the players.

**NOTE: You are only allowed to use fundamental Python data types (lists, tuples, dictionaries, numpy.ndarray, etc.) to complete this assignment.** You are not allowed to use advanced data querying and data analysis packages such as pandas, sqlite, networkx, or similar. We impose this restriction in order to test your grasp of fundamental programming concepts, not your scripting experience with Python libraries you acquired from prior work or other courses.

## Hints

Although this assignment is quite streamlined, imagine that the tasks here are part of a larger project. How would you structure your program if in the future you may need to use a different dataset with similar structure, manipulate the data somewhat differently, add additional analyses, or modify the focus of the current analysis?

Keep different data manipulations in separate functions/methods and group related functions/classes in separate .py files. Name your modules in an informative way.

Using an object-oriented approach to solve the problems is entirely optional and you will not obtain bonus points for this. If you are not confident in your programming skills, we recommend developing your solution with functions only.

## Data

Any time when you refer to the data in your code, please use a relative path such as `'../assignment-final-data/filename.csv'` instead of an absolute path - This way, we will be able to test your submission with our own copy of the data without having to modify your code.

The repository contains fifteen .csv files with match results, one file for each year. Each file contains the following variables:

- Tournament – the name of the tournament that the match was part of.

- Start date – the date when the tournament starts.
- End date – the date when the tournament ends.
- Best of – 3 means that first player to win 2 sets wins match (all WTA matches are best of 3 sets).
- Player 1, Player 2 – names of the players in the match.
- Rank 1, Rank 2 – WTA ranks of Player 1 and Player 2 before the start of the tournament. Not all players will have a ranking.
- Set 1-3 – result for each set played where the score is shown as: number of games won by Player 1 - number of games won by Player 2. The player that wins the most games in a set wins that set.
- Comment
  - Completed means match was played.
  - Player retired means that the named player withdrew and the other player won by default.

## Hints

When writing your code, test it on a small "toy dataset", instead of the entire data. This way, you won't need to wait for minutes/hours just to find out that you have a syntax error!

You should not modify the original data in any way. If your code creates new data files, you should save them in the directory where this file resides.

You should consider whether and which estimates to save on disk to speed up queries, instead of calculating them again and again from the raw data. If you decide to do so, please write your code to save any such files with processed data in the directory where this file resides. This way, we can run your code without having to alter it. Think about what is the most efficient way, both in terms of time and space, to save the data.

## 1. Reconstructing the tournaments

Tournaments in tennis are typically in knockout format. In each round there are several pairs of fixtures, the winners of each match advance to the next round and the losers are eliminated. The winners then proceed to the next round and the process continues until two players contest the final. Typically, the rounds in the competition go as follows: [First Round, Second Round, ...,] Quarterfinals, Semifinals, Final.

In some cases, tournaments have a Round Robin (also known as all-play-all) group stage, meaning that each player in a group plays against each other player in turn. There are usually two parallel groups with 4 players in each. The top player(s) (i.e. those who won the most matches) in each group advance to a short knockout stage (typically just Semifinals and Final). These tournaments are:

- Sony Ericsson Championships 2007-2015
- Commonwealth Bank Tournament of Champions 2009
- Qatar Airways Tournament of Champions Sofia 2012
- Garanti Koza WTA Tournament of Champions 2013-2014
- BNP Paribas WTA Finals 2016-2018

- WTA Elite Trophy 2015-2019
- WTA Finals 2019, 2021

Very occasionally, tournaments also include a Third Place match too.

Your task is to identify the winner in each match and the round in which the match was played. To check your work, please call the procedures you have written to print the answers to the following questions:

- Who won the final of the 2021 Women's US Open?
- Who played against whom in the 4th Round of the 2018 French Open?
- In which round was Venus Williams eliminated in the 2011 Australian Open?
- How many finals has Naomi Osaka played in until now?
- How many times have Venus and Serena Williams played against each other and how many of these matches each won?

## 2. Winners win

One simple and naive way to rank players is to count how many matches they won each year. Write a procedure that estimates this. Then print the three top ranked players for the year 2021 and for the period 2007-2021, together with the total number of matches they won. Higher scores are better, so ranking is the reversed order of most matches won.

## 3. Winners don't lose

A more sophisticated ranking algorithm will account for the fact that some players may play fewer games (e.g., due to an injury) and that wins in later stages of a tournament (e.g., in the final and semi-final compared to earlier rounds of the competition) are more important. Write another procedure that estimates a player's rank by adding  $r$  points for every win and subtracting  $1/r$  for every loss, where  $r = 1$  for the lowest elimination round of the tournament,  $r = 2$  for the next round of the tournament, and so on. In other words,  $r$  starts at 1 and increases for every next elimination round. This way, winning larger competitions brings more points (they have more elimination rounds), wins in later rounds improve one's rank more, and losses in earlier rounds diminish one's rank more. For round-robin stage matches, assume that  $r = 1$ .

Use this measure to print the three top ranked players for the year 2021 and for the period 2007-2021, together with their scores.

## 4. Winners beat other winners (WbW)

Another idea for ranking players is that winning over better players should count more. However, how we measure a good player depends on whether they beat other good players, so we get into

a recursive situation. Do not worry, you will not have to write a recursive procedure here as every recursive solution can be rewritten as iteration! This is what we will do:

1. First, count all the players in the given data and assign each a score of  $1/n$ , where  $n$  is the number of unique players.
2. Then, repeatedly do the following sequence of steps:
  - A. Each player divides its current score equally among all the matches they have lost and passes these shares onto the players they lost to. If the player never lost, then they pass their current score to themselves. If the player lost two times to a specific individual, then they pass two shares to that individual.
  - B. Each player updates their score to be the sum of the shares they receive.
  - C. Rescale the score of each player by multiplying it by 0.85 and adding  $0.15/n$ .
3. Repeat the procedure until adjustments are too small to matter. You may need to come up with modifications of the algorithm or the data if the algorithm cannot converge or produces non-sensical results in specific situations.

This algorithm essentially starts with a world in which everyone is equally important and then starts to "pass importance" iteratively to the winners until an equilibrium is reached.

Use this measure, which we will call WbW, to print the three top ranked players for the year 2021 and for the period 2007-2021, together with their scores.

## 5. Compare your WbW ranking measure

In fact, the data already contain the players' official WTA rank in the variables Rank1 and Rank2, which follows its own [complex procedure](#). How well does your WbW ranking correlate with the WTA ranking over time?

Use the data from 2007 to estimate the players' WbW ranking at the end of this year. This will give you sufficient data to initialize your estimate.

Then, starting with 2008, update your ranking before the start of each tournament, based only on results from tournaments completed in the previous 52 weeks. Use a scatter plot from matplotlib to plot the players' WTA ranking on the x-axis against your WbW ranking calculated before the start of the same tournament on the y-axis. If an individual's WTA ranking changes mid-tournament (rare), take their first listed ranking within that tournament. Each point on the plot should be a player's ranking before the start of each tournament for each tournament they participated in from 2008 until now. So the number of points to plot will be the number of unique individuals that took part in each tournament for all tournaments during that period.

Write a couple of sentences to note what you observe. You will not be marked for your observations but some reflection is important as it could help you identify problems with your code, for example.

## Import and run your code here

Keep your code in separate .py files and then import it in the code cell below. Then call the functions/methods you need to conduct the analysis described above and print the requested

outputs. We should be able to run the cell below to calculate again the results and get the requested output, without having to modify your code in any way.