Homework #3 - pandas

Re-submit Assignment

Due Oct 18, 2019 by 11:59pm **Points** 10 **Submitting** a file upload
File Types html **Available** after Oct 3, 2019 at 10pm

Homework #3 - pandas

Complete these exercises and submit a single Jupyter Notebook file (in .html format, not .ipynb) that contains your responses by **Midnight (11:59PM) on Wednesday 10/16**. Late assignments will be penalized up to 2 points per day (20%), unless prior arrangements have been made to submit the assignment after the deadline.

The notebook should be well organized. Each section should be **clearly labeled with the exercise (and part) that it addresses** (e.g., Exercise #1a, #1b, #2) in a Markdown cell block. Use (clear and concise) comments as needed to help describe each step of your process. All notebook cells that contain essential steps should be executed and the output should be visible, so as to demonstrate your successful completion of the exercise. If you cannot complete an exercise in its entirety, you should make an effort to demonstrate your intermediate progress in order to maximize partial credit, and move forward as best as possible. You may submit any written answers to the exercises in the notebook as text cells.

Academic Integrity: Each student is expected to submit his or her own original work. You may collaborate with your classmates on the concepts of the homework assignment, but you should not submit the same documentation for any part of the assignment. Submissions that contain significant similarities will be reported directly to the Office of Student Conduct.

Background

After you develop your code to play Addition War and perform descriptive analysis using NumPy functionality (as in Homework #2), you realize that it's a bit inefficient to store all of the results in separate arrays. Let's utilize pandas to store and analyze the results of our simulated gameplay.

Exercise #1 (4 points)

Before analyzing the results, let's begin with some updates to your **play** function (corrected, or taken from the Homework #2 solutions, once provided; you will also need to make use of your **deal** function, but no changes will be needed):

Step 1.1: (1 point) Add the capability to track the number of ties for each game. In this total, you should include not only the ties that occur at the beginning of a round, but any additional ties that occur during a tie breaker.

Step 1.2: (1 point) Add the capability to track the largest total won during a round in a given game, which could occur in a standard round (with no ties) or a tiebreaker round (with one or more ties). You do not need to distinguish whether the total was won in a standard or tiebreaker round.

Step 1.3: (1 point) Add the capability to play multiple games within the function, using a new argument *games* (equals 1 by default). The function should track the following statistics for each game, stored in a dictionary (ideally as separate entries, which will facilitate next steps):

- Initial totals (i.e., sum of card values for initially dealt hands) for player 1 and player 2, respectively
- Number of ties during gameplay
- Maximum round total won (i.e., maximum sum of card values won during a round, potentially including tiebreakers)
- Final totals (i.e., sum of card values after completing gameplay) for player 1 and player 2, respectively

Note that you do not need to explicitly track the outcome of the game (i.e., who wins or loses, or whether there was a tie). This will be determined at a later step.

Step 1.4: Demonstrate your function for 1-game (*games* = 1) and 5-game samples (*games* = 5), and display the resultant dictionaries for each case (1 point).

Exercise #2 (1 point)

Simulate 1000 games and store your results in a pandas DataFrame. Preview the first and last 10 rows of your results as proof of a successful run.

Exercise #3 (3 points)

Given your DataFrame of simulated game results, let's determine the outcome of each game, based on the final player totals.

Step 3.1: (1 point) Add a new column to your DataFrame ('outcome') that contains the value 'Player 1 wins' if Player 1's final total is higher, 'Player 2 wins' if Player 2's final total is higher, or 'Tie game' if the two player's final totals are equal.

Step 3.2: (1 point) Add new columns for the winning total ('winning_total') and losing total ('losing_total'), which contain the final totals for the winning and losing players, respectively, and missing values (NumPy.nan) for tie games.

Step 3.3: Preview the first 10 rows of your modified DataFrame. Then, summarize the frequency of the three outcomes and display the rows in your DataFrame that contain the 'Tie game' outcomes.

Step 3.4: Produce descriptive statistics (e.g., minimum, mean, standard deviation, maximum) for your winning total column, and display the rows in your DataFrame that contain the top 10 winning totals. (1 point for 3.3 and 3.4)

Exercise 4 (2 points)

Step 4.1: Given your DataFrame of simulated game results, calculate the frequency of the number of ties within each game (i.e., the number of rounds in which a tie occurs).

Step 4.2: Calculate the mean maximum round total as a function of the number of ties. What do you observe about the relationship between the number of ties and the mean maximum round total?

HW#3 Rubric