Homework #2 - Python Essentials + NumPy

Re-submit Assignment

Due Oct 1, 2019 by 4pm **Points** 10 **Submitting** a file upload **File Types** html

Homework #2 - Python Essentials + NumPy

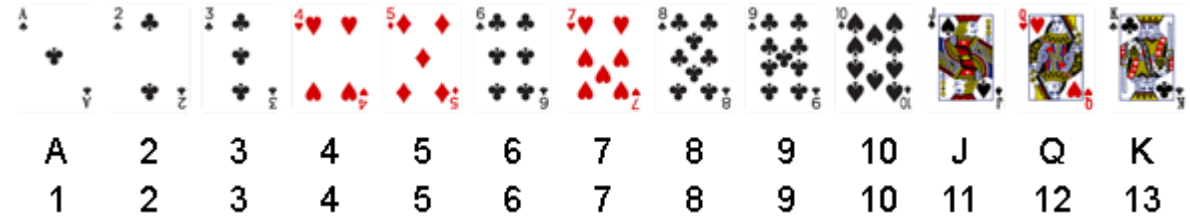
Complete these exercises and submit a single Jupyter Notebook file (in .html format, not .ipynb) that contains your responses by **4PM on Tuesday October 1st**. Late assignments will be penalized up to 2 points per day (20%), unless prior arrangements have been made to submit the assignment after the 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

In the classic card game War, a standard deck of (52) playing cards is shuffled and distributed evenly among two players (26 cards each). For each round of gameplay, the players lay out one card at a time (for our purposes, in order of how they have been dealt) and the player with the higher card wins the round (where Ace = 1, 2-10 equal their face value, and Jack, Queen, and King = 11, 12, 13, respectively, across each of the four suits: diamonds, hearts, clubs, spades).



Ties are broken with additional plays until one player wins the full round
(i.e., all of the cards played during the round, including those that were tied). Although variations of the game can continue until one player wins all of the cards, let's consider the simple case in which cards won during a previous round cannot be replayed.

Traditional War

Winner

A variant of War, called Addition War, consists of two cards being played by each player at a time, where the player with the higher sum of two card values wins the round. Each player has a "discard" pile. The winner of the round takes their winning two cards and the loser's two cards and places the cards in his or her discard pile.

Addition War

Winner

Similar to the original game, ties are broken with additional plays of two cards each until one player wins all of the cards in the round.

Once all cards have been played (once), the player with the higher score (sum of card values in the discard pile) wins the game. If the players run out of cards in the middle of a tiebreaker, assume that each player's respective cards (for that round) are returned to them for the final score.

Exercise 1 (1 Point)

Step 1.1: Write (and execute) a statement that will generate a list of values associated with a standard deck of playing cards. You do not need to differentiate between suits (i.e., diamonds, hearts, clubs, spades), as they have no relevance for this particular game. The "deck" should have four 1's, four 2's, ... , four 11's (Jacks), and so on.

Step 1.2: Report the total value associated with a full deck of cards, which represents the maximum total for a game.

Exercise 2 (2 Points)

Step 2.1: Define a function **deal** that will shuffle (randomize) and distribute the 52 playing cards evenly to two players (26 each) and return a tuple of each player's hand (as a list of values). The function does not need to take in any arguments, and should create the deck of values internally (i.e., you should not need to input the deck of values into the function; you may reuse the statement you developed for part a). You may assume that you are always playing with a standard deck (52 cards). You may make use of any NumPy functionality to perform the shuffle operation. Hint: check out this link for the functions available in np.random.

https://docs.scipy.org/doc/numpy-1.14.0/reference/routines.random.html#
(https://docs.scipy.org/doc/numpy-1.14.0/reference/routines.random.html#)

Step 2.2: Deal a hand to two players and display (print) each player's hand. Print each player's number of cards and the total value of the hand.

Exercise 3 (3.5 Points)

Simulate a game of **Addition War** between two players.

Step 3.1: In a markdown cell, write out the steps of what should occur in a game of Addition War. This will provide an outline for your code to follow. I have written the first and last two steps for you below. Be sure to include what happens in the case of a tie and what happens if the deck is exhausted in the middle of a tie. What is the maximum number of rounds that could be played in the game (occurs when there are no ties)?
1. Shuffle deck
2. Deal half of deck to each player
...

XX. Game terminates after a maximum of _?_ rounds (occurs when there are no ties) and the deck is exhausted.
XX. The player with the highest total card points (in their discard pile) wins the game.

Step 3.2: Define a function **play** that will simulate a single game of Addition War between two players. This function will have no arguments. As written above, note that you will need to account for rounds in which a given player wins outright (i.e., after the initial play of two cards) and for rounds in which one or more ties occur. You will also need to terminate the game appropriately (i.e., after each player has played all of their cards once).

Step 3.3: From your play function, return the winner of the game and the card totals for the winner and loser. Print the result.

Exercise 4 (3.5 Points)

Given your implemented game, let's perform some analysis of the gameplay. For brevity in your notebook, you may want to create a copy of your play function that does not include any (active) print statements to use for this part of the assignment.

Set Up: Simulate n = 20 games of Addition War, and provide the following results and analysis (1 point each):

Step 4.1: Extract the final scores for the winning and losing players, and produce descriptive statistics for each scenario, specifically the
minimum, mean, and maximum totals after each game. What do you observe about the winning and losing totals?

Step 4.2: What proportion of games resulted in a winner that had a lower initial total than their opponent? What was the lowest initial total to win a
game (or conversely, the highest initial total to lose a game)?

Step 4.3: Calculate the (linear) correlation between the initial and final totals for winners vs. losers? What do you observe about the respective
correlations, and what do these correlations tell you about the relationship between the initial and final totals?

Homework #2