6.0002/6.00 Fall 2018

Problem Set 4: Blackjack Game Simulation

Handed out: Monday November 12, 2018
Due: November 21, 2018 at 9 PM
Checkoffs: November 26 – December 6th

Because this pset is due right before Thanksgiving, for this pset *only*, the late day policy is as follows:

- **0 late days**: due November 21 at 9 PM
- **1 late day:** due November 25 at 9 PM
- **2 late days:** due November 26 at 9 PM
- **3 late days:** due November 27 at 9 PM

What is Blackjack?

Blackjack, also known as Twenty-One, is a card game, usually played with one or more standard decks of cards. A standard deck of cards contains 52 cards, 13 each of four suits: clubs, diamonds, hearts, and spades. There are 13 cards for every suit with the following ranks: A (ace), 2, 3, 4, 5, 6, 7, 8, 9, 10, J (Jack), Q (Queen), and K (King). Cards have different point values based on their rank.

Point values of the cards:

- **Face cards (Jack, Queen, King)**: 10 points
- **Ace**: either 1 point or 11 points
- **All others**: numerical value on card

The objective of the game is to get the sum of your card point values as close to 21 without exceeding it. The player (or the dealer) busts if their points go over 21.

Here is a simplified progression of the game:

1. Player **puts a bet down** before the cards are dealt.

2. One card is dealt face up to the player, and then one card is dealt face up to the dealer. Another card is dealt face up to the player, and then another card is dealt *face down* to the dealer. The face-down card is the **hole-card**, and it is hidden from the player[1].
3. The player (or the dealer) has a **blackjack** if the point values of the 2 cards initially dealt to them add up to exactly 21.
    a. If the dealer gets a blackjack, and the player does not, the dealer automatically wins the game.
    b. If the player gets a blackjack, and the dealer does not, then they win the game.
    c. If both the dealer and player get blackjack, neither party wins, and the game ends.
    d. Otherwise, game play continues with the next step.
4. The player has fewer than 21 points, so they have two options:
    a. **Hit:** Get another card. The player can do this as many times as they want in a single turn.
    b. **Stand:** Do not get any new cards. Stay at the current point value.
5. The dealer will always employ the same *dealer's strategy*. If the dealer does not have 21, they hit until they get to a value of 17 or above. This is also known as "stand on soft 17."

End game scenarios

1. The original bet is returned to the player, and the player **wins an additional 1.5*(original bet)** if:
    a. they get a blackjack (i.e. the initial 2 cards dealt to them add to exactly 21), and the dealer does not
2. The original bet is returned to the player, and the player **wins an additional amount equal to the original bet** if:
    a. the dealer busts (and the player did not)
    b. the player has a higher point value than the dealer
3. The player **loses and receives nothing**—effectively losing the original bet—if:
    a. the player busts (it doesn't matter what the dealer has)
    b. the dealer has a higher point value than the player
4. The dealer and the player **tie**, and the original bet is returned to the player if:
    a. both have blackjack (i.e. the initial 2 cards dealt to them add to exactly 21)
    b. their hands are worth the same number of points

Getting Started

- `blackjack.py` – implement the functions in this file to complete the pset. **Turn this in when you are done.**

[1] If the face-up card is a ten-card or an ace, then the dealer will look at the hole-card to see if they have a blackjack with the first two cards they were dealt (i.e. the value of their cards is exactly 21). Otherwise, the dealer reveals the face-up card only after serving the player (i.e. after the player has bust or chosen to stand).

- `ps4_classes.py` – contains class definitions that will be helpful for the pset. **Do not modify.**
- Do not rename files, change function/method names, or delete/edit given docstrings.
- Please review the Style Guide on Stellar. Points will be deducted if there are violations.

Problem 1

We will be representing a hand of Blackjack using the **`BlackJackHand`** class, which keeps track of the `deck` and the cards dealt to the `player` and the `dealer`.

You will be implementing 3 different playing strategies as part of this problem:

- **Mimic dealer strategy:** The player will mimic the strategy used by the dealer to determine their next move. In other words, the player will hit until the best value of their cards (i.e. best sum of their cards) is greater or equal to 17.

- **Peek strategy:** The player knows the value of the dealer's face-down card (the hole-card), and makes a decision on whether to hit or stand based on that. The player will hit until the best value of their cards is greater or equal to the best value of the dealer's cards.

- **Simple strategy:** The player will hit until either of these two cases is true: (1) the player's best value is greater than or equal to 17 or (2) the player's best value is between 12 and 16 (inclusive) AND the dealer's face up card is between 2 and 6 (inclusive).

Implement the methods in the `BlackJackHand` class, as per the docstrings. You may find the classes defined in `ps4_classes.py` helpful when implementing `BlackJackHand`.

You should now pass the following tests in **`test_ps4_student.py`**:

```
test_best_val_no_aces_*,
test_best_val_one_ace_*,
test_best_val_multiple_aces_*,
test_mimic_dealer_strategy_*,
test_peek_strategy_*,
test_simple_strategy_*,
test_play_player_*,
test_play_dealer_*
```

Problem 2

Now that you have implemented the strategies and rules for playing a hand of Blackjack in `BlackJackHand`, let's implement the logic required to play one round of Blackjack and to determine how much money the player receives at the end of the game.

At the end of the game the player will receive:

- 2.5 times their bet if they win/get blackjack with the first 2 cards dealt to them, and the dealer does not
- 2 times their bet if:
    - the dealer busts,
    - the best value of the player's hand is higher than that of the dealer
- the original amount they bet if the game ends in a tie
- 0 if the dealer wins, or the player busts

Implement this logic in the function `play_hand` in `blackjack.py`. This function should return the total amount of money the player gets back. Remember this includes the amount they initially bet.

You should now pass the following tests in **`test_ps4_student.py`**:
`test_play_hand_mimic_*,`
`test_play_hand_peek_*,`
`test_play_hand_simple_*`

Problem 3

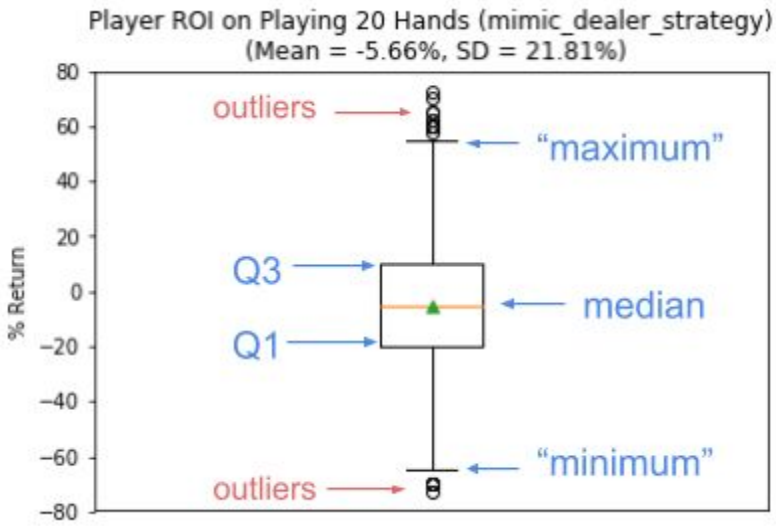
Create a simple simulation that determines the average and standard deviation for the **rate of return** associated with a given playing strategy. Your simulation should be able to run many trials. Each trial may have multiple hands of Blackjack. For each trial, calculate the *rate of return* after playing all the hands**.** If the player lost money, then the total amount of money earned—and thus the rate of return—will be negative.

$$rate\ of\ return\ (\%)\ = 100 * \frac{total\ money\ received - total\ money\ bet}{total\ money\ bet}$$

Once you have determined the rate of return for each trial, find the mean and standard deviation across all the trials. You should then plot the distribution of rates of return using a box plot.

Box Plots

Box plots provide a useful visual representation of the data distribution. The five number summary includes the "minimum", first quartile (Q1), median, third quartile (Q3), and "maximum".

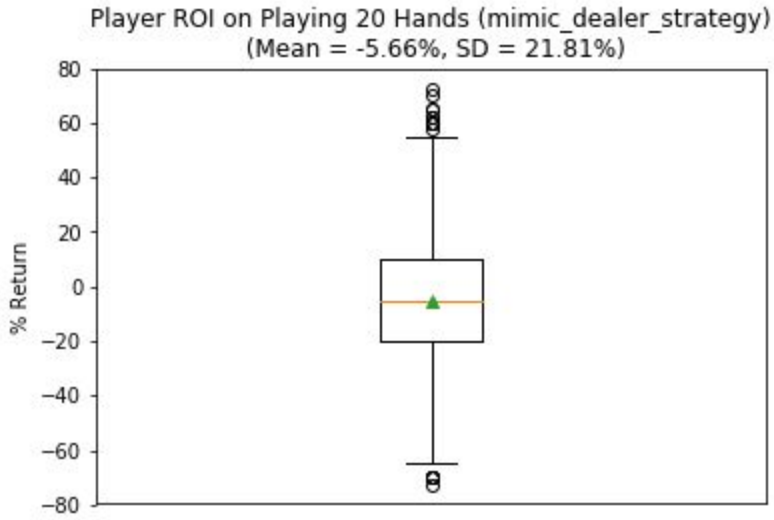


median = value at 50th percentile
first quartile (Q1) = value at 25th percentile
third quartile (Q3) = value at 75th percentile
IQR = Q3-Q1
"minimum" = Q1 - 1.5 * IQR
"maximum" = Q3 + 1.5 * IQR
outliers = values smaller than "minimum" or larger than "maximum"

You can read more about box plots at
https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

You may find `matplotlib.pyplot.boxplot` helpful in generating the required plot. Use the `showmeans` parameter to show the average on the boxplot. Remember to include the title, y_axis, and corresponding labels in your final plot.

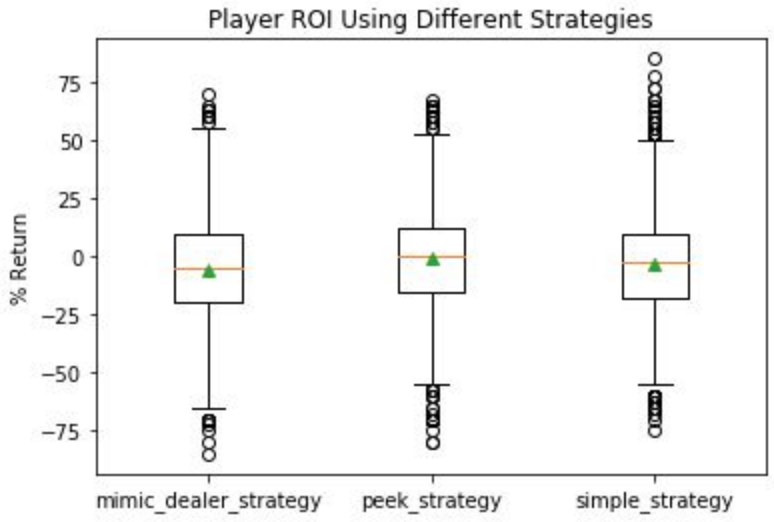
Here is an example:



Implement the function **`run_simulation`**. This function takes in a boolean parameter `show_plot`, which determines whether or not to display the plot.

Next, we want a way to visually compare the three strategies we implemented in problem 1. Implement the function **`run_all_simulations`**. The function runs the simulations for each strategy and displays a single graph with one box plot for each of the strategies.

Here is an example:



You should now pass these tests in **`test_ps4_student.py`**:
`test_run_simulation_mimic`,
`test_run_simulation_peek`,
`test_run_simulation_simple`

Hand-in Procedure

1. Naming Files

Save your solutions with the original file name: **blackjack.py**. **Do not ignore this step or save your file with a different name!**

2. Remove print statements

Please remove all print statements from your code before you submit your pset.

3. Time and Collaboration Info

In a comment at the start of your file, write down the number of hours (roughly) you spent on the problems in that part and the names of the people with whom you collaborated.

For example:

```
# Problem Set 4
# Name: Jane Lee
# Collaborators: John Doe
# Time Spent: 3:30
# Late Days Used: 1 (only if you are using any)
# … your code goes here …
```

4. Sanity checks

After you are done with the problem set, do sanity checks. Run the tester, and make sure your code passes all the tests. **You should never submit code that immediately generates an error when the tester runs!**

5. Submit

To submit blackjack.py, upload it to the problem set website linked from Stellar. You may upload new versions of each file until the **9 PM deadline**, but anything uploaded after that time will be counted towards your late days, if you have any remaining. If you have no remaining late days, you will receive no credit for a late submission.

After you submit, please check your submission and double-check that you submitted the right file.