# Blackjack Simulation Project
# ISYE 6644

Hillary Reyes
reyes_hillary@gatech.edu

*Abstract*—In this project I created a Blackjack simulation in Python, testing out two different playing strategies with varying parameters as well as two different betting strategies. The first playing strategy was one when a player stands (does not accept more cards) at varying hand values. The second playing strategy took a "soft" hand (presence of an Ace) into account. The first betting strategy was a conservative approach that increased a player's bet only if they won. The second was an aggressive approach called the Martingale strategy in which a player doubles their bet if they lost the previous round. I found that the best realistic combination that minimizes losses the most was the soft playing strategy and the conservative betting combination. The Martingale had some potential profit, but is not realistic due to cash and betting limits.

## 1 BLACKJACK BACKGROUND

### 1.1 Rules

In this particular simulation, there will be one player and one dealer. The player puts down a starting bet. The player and the dealer are dealt two cards from the start. From there, each player will decide to hit, accept another card, or stand, not accept another card. The goal of the game is to get a higher sum of a hand than the dealer without going over 21. The dealer will hit until they have a hand of 17 or higher. If a player goes over 21, the dealer wins regardless of his or her own hand. If the player is dealt an Ace and a card with a value of 10 from the start, that is a Blackjack and the player receives 2.5x their bet. Otherwise, if a player wins, they gain their bet. If they lose, they forfeit their bet to the dealer.

### 1.1 Card and Hand values

The deck will be a standard 52 card deck. Most casinos use about 4-8 decks, so I will use the standard 4 deck method. Each of the number cards equal their face value. Face cards, on the other hand, are worth 10 points each. An Ace card can equal 1 or 11, whatever the player prefers.

### 1.2 Types of Hands

A soft hand is one in which an Ace is counted as an 11. A hard hand is one in which there is either no Ace or it is used as a value of 1.

## 2 STRATEGIES TESTED

In this simulation, I tested two different types of strategies: playing and betting. I tested two different types within each and combined them to find the one that rendered the highest winnings.

### 2.1 Playing Strategies

#### *2.1.1 Hard*

In the hard strategy, the player will stand at a certain hand value. For example, if the stand parameter is 17, any time the player’s hand hits 17 or higher, they will stand. Otherwise, if the player’s hand is less than 17, they will hit.

#### *2.1.2 Soft*

In the Soft strategy, the player takes into account whether or not their hand is soft. That is, whether their hand contains an Ace and the Ace is used as an 11. For Example, if the stand parameter is 16, if the player has a soft 16, such as an Ace and 5 card, the player will hit. If their hand is less than a 16, the player will also hit. Otherwise, if their hand is greater than 16, they will stand.

### 2.2 Betting Strategies

The starting bet in this simulation will be a standard $10. This will be the starting bet for each round if the requirements in the below strategies are not met.

#### *2.2.1 Conservative*

In this approach, the player will increase their bet by one unit, equal to half of the starting bet, if they win the previous hand.

#### *2.2.2 Martingale*

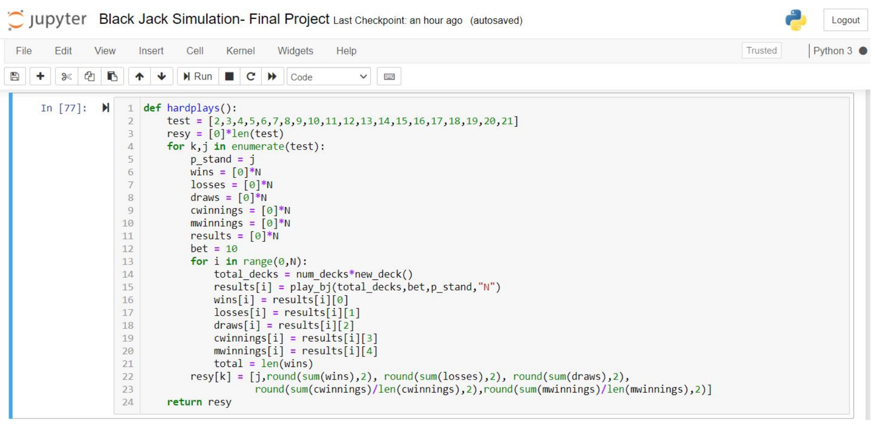
In this approach, the player will double their bet if they lose the previous hand.

## 3 SIMULATION SETUP

I wrote the Monte Carlo simulation in Python, using this open source Blackjack simulation, originally written in R, as the foundation for my play logistics. I used my own code for the creation of the hand total function `total()` , which summed up the hand of both the player and dealer and changed the value of an Ace to whatever was most beneficial for them, as well as other additions, such as the different betting strategies within the main `play_bj()` function, the function `softplays()` which simulated games using the soft playing approach given a set number of iterations, the function `hardplays()` which simulated the hard playing approach, and more. I wrote these play functions so that each game kept a tally of the wins, draws, and losses of each game using the appropriate parameters, as well as the winnings if the player used the conservative or the Martingale strategy. The play functions ran simulations at each potential value for a hand, 2-21, and used these different values as a `p_stand` parameter, for which the player would use as their minimum stand value.

I experimented with different iteration parameters, running my simulation for 1,000 iterations and 20,000 iterations

I ran my simulation in Jupyter Notebooks for ease of writing and organization since I do not use Python regularly. A screenshot of how I ran my program can be found below. A complete copy of my code can be found in the appendix.



```python
def hardplays():
    test = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]
    resy = [0]*len(test)
    for k,j in enumerate(test):
        p_stand = j
        wins = [0]*N
        losses = [0]*N
        draws = [0]*N
        cwinnings = [0]*N
        mwinnings = [0]*N
        results = [0]*N
        bet = 10
        for i in range(0,N):
            total_decks = num_decks*new_deck()
            results[i] = play_bj(total_decks,bet,p_stand,"N")
            wins[i] = results[i][0]
            losses[i] = results[i][1]
            draws[i] = results[i][2]
            cwinnings[i] = results[i][3]
            mwinnings[i] = results[i][4]
            total = len(wins)
        resy[k] = [j,round(sum(wins),2), round(sum(losses),2), round(sum(draws),2),
                   round(sum(cwinnings)/len(cwinnings),2),round(sum(mwinnings)/len(mwinnings),2)]
    return resy
```

*Figure 1*—Jupyter Notebook using hardplays() function.

## 4 RESULTS

Full results tables can be found in the appendix.

In the two tables below, I included the averaged results of games in simulations of 20,000 iterations, an iteration being a game from start until the deck runs low and is refreshed. Each row represents 20,000 iterations using a certain hand threshold, or stand value. For example, in Table 1, a stand value of 14 meant that in every game, the player would only take a hit if their hand value was 13 or below. If the hand hit a value of 14 or above, they would stand. In table 2, a stand value of 15 means that in every game, the player would hit if their hand was a soft 15, a hand equal to 15 and with an Ace. If their hand was below 15, they would hit. Otherwise if their hand was above 15, they would stand.

In both tables, lower stand values had lower win rates and winnings. This is due to the fact that a lower hand is less likely to win since the goal of the game is to beat the hand of the dealer. Similarly, higher stand values also had low win rates due to the higher chances of busting. For example, if a player randomly drew two 10s, the hand would equal 20 and with a hard 21 strategy, they would draw. The probability of not busting after that would be 4/50 = 8%, the chances of pulling an Ace. There would be a 92% chance of pulling a card that would cause a player to bust.

The next columns describing profits have to do with the average profits per game using the two different betting strategies described in section 2. In this case, the player had higher losses than wins, thus most profits are negative.

The last column includes the average max Martingale bet per game. For this, I recorded the maximum bet using the Martingale betting strategy per game and then found the average over the 20,000 iterations per row. I recorded this to demonstrate the scale of bets needed in order to make some of the Martingale profits in the below tables.

*Table 1*—Hard Strategy Results with 20,000 iterations (truncated)

| **Stand Value** | **Win%** | **Losses%** | **Draws%** | **Avg Conservative Profits per Game** | **Avg Martingale Profits per Game** | **Average Max Martingale Bet per Game** |
|---|---|---|---|---|---|---|
| 14 | *42.624%* | 48.833% | 8.543% | -$71.19 | $5,547,607.18 | $24,830,324.75 |
| 15 | *42.283%* | 48.152% | 9.565% | -$62.02 | -$16,564,106.48 | $16,063,310.80 |
| 16 | ***41.672%*** | **47.713%** | **10.615%** | **-$57.85** | **-$73,399.49** | **$3,198,068.27** |
| 17 | *40.562%* | 47.515% | 11.924% | -$61.39 | -$307,853.88 | $1,271,671.00 |
| 18 | *39.186%* | 50.022% | 10.792% | -$96.26 | -$412,400.41 | $2,653,157.14 |
| 19 | *35.187%* | 55.155% | 9.658% | -$170.74 | -$2,764,076.09 | $7,549,753.32 |
| 20 | *28.244%* | 63.473% | 8.283% | -$274.39 | -$40,486,477.95 | $44,416,730.28 |
| 21 | *14.786%* | 79.971% | 5.243% | -$396.68 | -$1,850,569,188.15 | $1,360,984,790.25 |

In the above table, we can see that when utilizing a hard stop value of 16, that being a player standing once they hit a 16 or above, renders the least amount of loss for the conservative betting strategy. We see that the Martingale strategy also renders negative profits most of the time, but there are a couple values with positive profits, associated with large bets being used in the game.

*Table 2*—Soft Strategy Results with 20,000 iterations, shortened for stand values 14+

| **Stand Value** | **Wins** | **Losses** | **Draws** | **Conservative Profits** | **Martingale Profits** | **Average Max Martingale Bet of each game** |
|---|---|---|---|---|---|---|
| 14 | *42.600%* | 48.875% | 8.525% | -$70.03 | $2,107,923.27 | $16,206,130.75 |
| 15 | *42.172%* | 48.270% | 9.558% | -$63.14 | -$492,442.30 | $7,024,913.00 |
| 16 | ***41.533%*** | **47.733%** | **10.734%** | **-$58.97** | **-$1,456,136.95** | **$3,888,324.90** |

| | | | | | | |
|---|---|---|---|---|---|---|
| 17 | *40.567%* | 47.561% | 11.872% | -$60.49 | $894,561.68 | $2,069,605.58 |
| 18 | *39.215%* | 50.009% | 10.775% | -$96.47 | -$1,095,198.53 | $3,336,232.09 |
| 19 | *35.225%* | 55.199% | 9.576% | -$171.44 | -$5,019,269.02 | $8,409,196.36 |
| 20 | *28.236%* | 63.465% | 8.298% | -$273.40 | -$41,857,763.30 | $50,033,848.86 |
| 21 | *14.740%* | 79.974% | 5.286% | -$395.53 | -$1,913,655,836.24 | $1,341,999,574.10 |

Again in this table, we see a threshold of 16 also renders the least loss (or most profit) for a conservative betting strategy, and again see mostly negative, but some positive profits for the martingale strategy.

## 5 ANALYSIS

When comparing the soft strategy to the hard strategy for the conservative betting strategy, the hard strategy yielded the lowest losses (or max profits) when hitting on a hard 16 with average -$57.85 vs the soft strategy's -$58.97. According to this Monte Carlo Simulation, the probability of a player winning against a dealer in this situation is 41.533% using the soft 16 strategy compared to 41.672% in the hard strategy.

Overall the reason the hard strategy minimized losses more was due to an increase in wins. By standing at hand value of 16, whether hard or soft, a player loses any chance of improving their hand. However, if the hand is a soft 16 and the player chooses to hit, they have the chance to improve their hand.

For example, let's say I randomly draw an Ace and a 5 out of 52 card deck. In blackjack, that would equal a soft 16. If I randomly draw another card out of the deck, there is a 36% chance the card would increase the hand without busting, equal to the probability of pulling a 2, 3, 4, 5, or Ace card. This is often thought of as a reason for casinos to use a soft strategy.

$$P_2 + P_3 + P_4 + P_5 + P_A = \frac{4}{50} + \frac{4}{50} + \frac{4}{50} + \frac{3}{50} + \frac{3}{50} = \frac{18}{50} = 36\%$$

However, should the player pull a card with a value from 6-10, the Ace would revert to a value of 1 and thus either lower and maintain the hand's value without busting. The probability of this situation happening has a higher probability.

$$P_6 + P_7 + P_8 + P_9 + P_{10} = \frac{4}{50} + \frac{4}{50} + \frac{4}{50} + \frac{4}{50} = \frac{32}{50} = 64\%$$

Looking at the winnings distributions for the conservative strategies using a hard 16 playing strategy over 10,000 iterations, the winnings converge to a normal distribution, but skew towards the negative side, reinforcing the idea that it is unlikely to make a profit using this strategy combination.

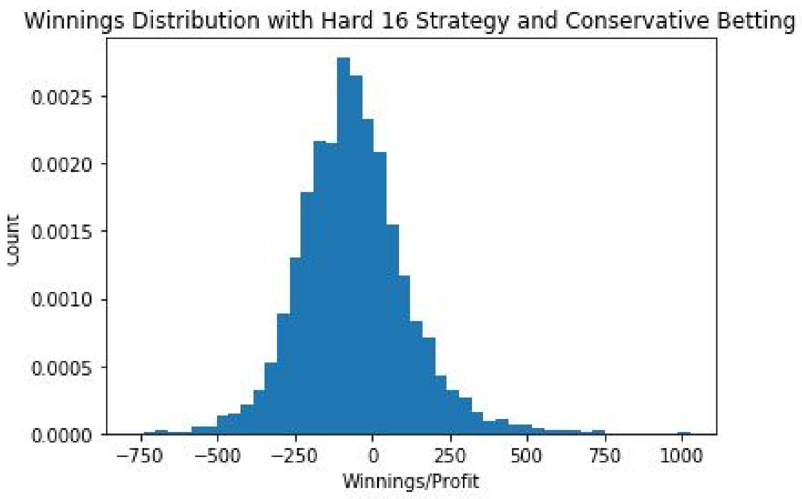


***Figure 2***—Histogram of winnings from 10,00 iterations using a Hard 16 strategy and Conservative betting

When comparing the betting strategies, Martingale rendered some positive profits in some games when we only ran a few simulations. Why not use the Martingale strategy? When using many more simulations, we saw that the strategy eventually averaged to losses. The ability to win large sums using the Martingale strategy hinges on the player's ability to use bet large amounts of money. As we can see in Table 1, the average of the highest bets used in the Martingale strategy for a positive profit in about 1000 random games reached as high as $1,000,000,000. To reach a bet this high, a player using the Martingale strategy would have to lose at about log(1,000,000,000) = 20.7 hands in a row. This would likely happen if the stand value was too high or too low, causing the player to lose more, thus increasing their bet more. However, this is not very realistic given most casinos have bet limits, such as $50,000, and the fact that most people don't have $ billion to bet.

The Martingale betting strategy also remained relatively volatile. I increased the number of simulations and took a look at some statistical parameters for the betting strategies, and the mean varied, but the median did stay relatively stable. The median amount of winnings for the Martingale strategy stayed stable around $480 over two different simulations with 100,000 iterations. While the Martingale betting strategy may render profits, it will require very high bets.

| R | Bet | Mean | Median | Max | Min | Standard Deviation |
|---|---|---|---|---|---|---|
| 1 | C. | -$57.91 | -$65.00 | $1,235.00 | -$1,120.00 | $177.80 |
|  | M. | -$696,470 | $480.00 | $16,444M | -$21,097M | $128M |
| 2 | C. | -$58.60 | -$70.00 | $1,485.00 | -$1,075.00 | $178.67 |
|  | M. | $379,011 | $480.00 | $16,106M | -$10,603M | $115M |

## 6 CONCLUSION

For a simple blackjack strategy, I found that standing on a hard 16 and a conservative betting strategy was the best combination to minimize losses given realistic betting circumstances and the rules described in this report. These results would likely vary given additional rules, which vary casino to casino.

On the other hand, we saw that there was a chance using a Martingale betting strategy could render some profits. If you happen to have a massive amount of money and win on a large bet at a casino that has no bet maximum, cut your losses and run before you hit another loss on a large bet.

During my research, I found countless strategies, both playing and betting. In terms of future work, I would like to explore more complex strategies, including playing strategies that take the dealer's hand and remaining cards into account. Famously, there are card counting strategies to maximize wins and profits which would be fun to code. There are also other environments one might take into account. For example, many casinos apply a "hit on soft 17" strategy instead of the hard stand on 17+ the dealer in my simulation used. Traditionally this method is seen by casinos as increasing the dealer's chances to improve their hands and thus minimize losses. Given all these different factors, especially the more complex playing and betting strategies, it would be interesting to see the win percentages, loss percentages, profit distributions, and how they vary. I learned that in Blackjack, if you want to win and make a profit in these circumstances, you will likely to take a more active approach and consider other factors, such as the dealer's hand when deciding to stand or hit and the cards remaining in the deck. While the stand on 17 rule is a popular approach for people who want to maintain a simple strategy, it will take a more complex one to maximize wins and profits, or perhaps a player would need Lady Luck on their side.

## 7 REFERENCES

1. Topor, James (2016). Blackjack Simulation via Monte Carlo Methods. https://rpubs.com/jt_rpubs/279263
2. Wikipedia (2020) Martingale (betting system). https://en.wikipedia.org/wiki/Martingale_(betting_system)
3. Sheldon, David. The Martingale Betting System. https://www.casino.org/blog/the-martingale-myth-does-this-betting-system-really-work/#:~:text=The%20Martingale%20Betting%20System%20can,in%20both%20theory%20and%20practice.

## 8 APPENDIX

Full table of results for soft strategy at 10,000 iterations

| Stand Value | Wins | Losses | Draws | Conservative Profits | Martingale Profits | Average Max Martingale Bet Of each Iteration |
|---|---|---|---|---|---|---|
| 2 | *39.30%* | 55.61% | 5.09% | -$252.97 | -$750,044,509.83 | $3,057,594,225.37 |
| 3 | *39.29%* | 55.66% | 5.06% | -$253.65 | -$393,085,107.67 | $1,918,222,718.08 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 4 | *39.27%* | 55.67% | 5.06% | -$256.11 | -$1,221,248,861.94 | $2,933,023,671.36 |
| 5 | *39.39%* | 55.58% | 5.03% | -$249.93 | -$276,736,159.08 | $2,061,118,146.93 |
| 6 | *39.36%* | 55.54% | 5.11% | -$248.65 | -$808,810,125.63 | $2,127,502,976.82 |
| 7 | *39.40%* | 55.57% | 5.03% | -$246.49 | -$255,173,751.02 | $1,415,055,280.61 |
| 8 | *39.33%* | 55.44% | 5.23% | -$241.38 | -$738,785,093.69 | $1,586,607,575.15 |
| 9 | *39.64%* | 55.01% | 5.35% | -$228.54 | -$768,167,262.12 | $1,246,593,698.02 |
| 10 | *40.03%* | 54.33% | 5.64% | -$208.83 | -$1,313,367,864.13 | $1,126,578,761.87 |
| 11 | *41.18%* | 52.65% | 6.18% | -$163.43 | $74,304,905.52 | $357,511,255.61 |
| 12 | *42.45%* | 50.94% | 6.62% | -$117.09 | -$90,449,604.65 | $150,177,982.21 |
| 13 | *42.65%* | 49.83% | 7.52% | -$88.24 | -$40,042,584.57 | $87,308,370.35 |
| 14 | *42.84%* | 48.67% | 8.50% | -$66.38 | -$2,343,881.05 | $15,708,683.53 |
| 15 | *42.04%* | 48.38% | 9.58% | -$67.41 | $3,106,276.35 | $10,577,397.33 |
| 16 | *41.60%* | 47.76% | 10.64% | **-$59.09** | -$516,072.89 | $4,718,575.38 |
| 17 | *40.54%* | 47.61% | 11.86% | -$63.99 | -$252,388.01 | $1,463,858.34 |
| 18 | *39.34%* | 49.82% | 10.84% | -$93.47 | -$555,164.26 | $2,379,276.60 |
| 19 | *35.22%* | 55.23% | 9.55% | -$170.85 | -$5,106,831.91 | $8,537,528.54 |
| 20 | *28.22%* | 63.54% | 8.24% | -$274.53 | -$66,383,293.15 | $56,260,019.86 |
| 21 | *14.80%* | 79.90% | 5.30% | -$396.23 | -$1,834,235,436.67 | $1,294,309,432.86 |

Full table for soft strategy at 20,000 iterations

| Stand Value | Wins | Losses | Draws | Conservative Profits | Martingale Profits | Average Max Martingale Bet<br><br>Of each Iteration |
|---|---|---|---|---|---|---|
| 2 | *39.290%* | 55.641% | 5.069% | -$253.70 | -$1,620,515,741.70 | $3,134,686,481.30 |
| 3 | *39.448%* | 55.506% | 5.047% | -$250.34 | -$893,600,072.71 | $2,400,116,952.27 |
| 4 | *39.365%* | 55.590% | 5.045% | -$253.73 | -$210,790,005.07 | $3,065,253,431.12 |
| 5 | *39.402%* | 55.578% | 5.020% | -$253.69 | -$1,597,941,600.12 | $3,577,307,378.71 |
| 6 | *39.374%* | 55.595% | 5.031% | -$250.25 | -$933,132,309.97 | $2,273,823,752.45 |
| 7 | *39.404%* | 55.537% | 5.059% | -$246.46 | -$1,249,349,399.54 | $2,288,957,218.40 |
| 8 | *39.323%* | 55.500% | 5.177% | -$242.78 | -$716,052,906.61 | $1,976,507,623.42 |
| 9 | *39.601%* | 54.998% | 5.402% | -$228.18 | -$550,302,281.50 | $1,625,226,662.93 |
| 10 | *40.068%* | 54.296% | 5.636% | -$206.21 | -$191,059,587.85 | $692,680,987.97 |
| 11 | *41.086%* | 52.758% | 6.156% | -$163.74 | -$131,846,405.28 | $356,954,139.88 |
| 12 | *42.458%* | 50.895% | 6.647% | -$114.77 | -$12,724,431.59 | $169,241,370.82 |
| 13 | *42.774%* | 49.705% | 7.521% | -$86.84 | -$4,346,312.96 | $43,416,049.86 |
| 14 | *42.600%* | 48.875% | 8.525% | -$70.03 | $2,107,923.27 | $16,206,130.75 |
| 15 | *42.172%* | 48.270% | 9.558% | -$63.14 | -$492,442.30 | $7,024,913.00 |
| 16 | *41.533%* | 47.733% | 10.734% | -**$58.97** | -$1,456,136.95 | $3,888,324.90 |
| 17 | *40.567%* | 47.561% | 11.872% | -$60.49 | $894,561.68 | $2,069,605.58 |
| 18 | *39.215%* | 50.009% | 10.775% | -$96.47 | -$1,095,198.53 | $3,336,232.09 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 19 | *35.225%* | 55.199% | 9.576% | -$171.44 | -$5,019,269.02 | $8,409,196.36 |
| 20 | *28.236%* | 63.465% | 8.298% | -$273.40 | -$41,857,763.30 | $50,033,848.86 |
| 21 | *14.740%* | 79.974% | 5.286% | -$395.53 | -$1,913,655,836.24 | $1,341,999,574.10 |

Full table for hard strategy at 1,000 iterations

| **Stand Value** | **Wins** | **Losses** | **Draws** | **Conservative Profits** | **Martingale Profits** | **Average Max Martingale Bet Of each Iteration** |
|---|---|---|---|---|---|---|
| 2 | *39.513%* | 55.392% | 5.096% | -$253.33 | -$525,205,718.04 | $2,036,841,612.00 |
| 3 | *39.623%* | 55.321% | 5.056% | -$249.71 | $624,599,917.04 | $983,040,144.96 |
| 4 | *39.290%* | 55.596% | 5.115% | -$256.10 | -$3,884,387,570.72 | $2,872,412,043.52 |
| 5 | *39.404%* | 55.567% | 5.028% | -$261.43 | -$2,741,797,182.80 | $2,867,476,294.72 |
| 6 | *39.017%* | 55.842% | 5.141% | -$266.40 | -$1,534,201,296.96 | $4,067,409,590.24 |
| 7 | *39.412%* | 55.478% | 5.111% | -$246.02 | -$318,341,967.94 | $1,118,054,111.36 |
| 8 | *39.392%* | 55.377% | 5.231% | -$247.56 | -$480,358,987.06 | $1,203,363,981.12 |
| 9 | *39.195%* | 55.398% | 5.407% | -$241.29 | $192,176,124.46 | $1,279,373,597.12 |
| 10 | *40.105%* | 54.232% | 5.663% | -$208.66 | $189,989,792.44 | $512,210,948.16 |
| 11 | *41.318%* | 52.430% | 6.252% | -$158.59 | $3,201,089.17 | $387,654,242.56 |
| 12 | *42.429%* | 50.865% | 6.706% | -$116.41 | $42,151,599.16 | $270,547,168.00 |
| 13 | *43.061%* | 49.451% | 7.488% | -$82.53 | -$234,527.43 | $25,904,658.96 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 14 | *43.117%* | 48.180% | 8.704% | -$61.20 | $680,471.87 | $6,371,457.20 |
| 15 | *42.342%* | 48.212% | 9.446% | -$58.26 | -$405,580.06 | $9,005,591.60 |
| 16 | *41.176%* | 48.016% | 10.807% | -$63.96 | -$596,246.10 | $2,643,365.64 |
| 17 | *40.251%* | 47.995% | 11.754% | -$72.02 | -$409,589.58 | $2,493,999.60 |
| 18 | *39.362%* | 49.936% | 10.702% | -$95.94 | $704,582.25 | $2,718,056.24 |
| 19 | *35.071%* | 55.682% | 9.247% | -$181.50 | -$14,536,086.19 | $15,139,588.88 |
| 20 | *27.798%* | 63.828% | 8.374% | -$278.87 | -$79,139,284.57 | $55,465,127.04 |
| 21 | *14.668%* | 80.103% | 5.230% | -$401.86 | -$2,481,727,536.50 | $1,571,792,628.48 |

Full table for hard strategy at 10,000 iterations

| **Stand Value** | **Wins** | **Losses** | **Draws** | **Conservative Profits** | **Martingale Profits** | **Average Max Martingale Bet Of each Iteration** |
|---|---|---|---|---|---|---|
| 2 | *39.31%* | 55.64% | 5.05% | -$253.71 | -$1,024,912,927.22 | $2,587,726,396.37 |
| 3 | *39.35%* | 55.61% | 5.04% | -$254.76 | -$715,731,334.34 | $3,164,837,238.46 |
| 4 | *39.37%* | 55.58% | 5.05% | -$254.95 | -$373,648,200.32 | $2,753,515,655.23 |
| 5 | *39.37%* | 55.55% | 5.07% | -$254.37 | -$922,137,136.46 | $2,467,383,580.24 |
| 6 | *39.35%* | 55.58% | 5.07% | -$248.44 | -$408,578,718.78 | $2,311,023,851.66 |
| 7 | *39.38%* | 55.55% | 5.06% | -$245.42 | -$869,445,809.89 | $2,353,972,630.17 |
| 8 | *39.36%* | 55.41% | 5.23% | -$242.75 | -$485,178,378.93 | $1,498,904,803.78 |
| 9 | *39.60%* | 55.03% | 5.37% | -$227.27 | -$252,653,180.01 | $1,051,282,364.70 |
| 10 | *40.03%* | 54.26% | 5.72% | -$204.91 | -$504,054,180.15 | $780,990,327.19 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 11 | *41.14%* | 52.72% | 6.14% | -$160.26 | -$46,220,941.58 | $303,487,487.81 |
| 12 | *42.60%* | 50.76% | 6.63% | -$110.74 | -$8,902,148.20 | $122,668,435.79 |
| 13 | *42.66%* | 49.83% | 7.52% | -$87.37 | -$27,992,771.86 | $60,622,452.14 |
| 14 | *42.63%* | 48.91% | 8.46% | -$70.33 | -$16,585,962.75 | $32,213,802.25 |
| 15 | *42.35%* | 48.15% | 9.50% | -$59.61 | -$2,811,703.39 | $7,292,028.96 |
| 16 | *41.60%* | 47.72% | 10.69% | **-$56.74** | -$803,988.76 | $3,060,219.47 |
| 17 | *40.48%* | 47.57% | 11.94% | -$61.92 | -$310,793.97 | $1,476,595.21 |
| 18 | *39.10%* | 50.13% | 10.77% | -$97.66 | -$261,711.25 | $2,047,794.19 |
| 19 | *35.21%* | 55.14% | 9.65% | -$171.93 | -$4,339,671.21 | $7,720,176.30 |
| 20 | *28.21%* | 63.50% | 8.29% | -$273.61 | -$38,284,076.67 | $43,035,554.83 |
| 21 | *14.96%* | 79.78% | 5.27% | -$395.94 | -$2,011,834,782.92 | $1,379,761,649.54 |

Full table for hard strategy at 20,000 iterations

| **Stand Value** | **Wins** | **Losses** | **Draws** | **Conservative Profits** | **Martingale Profits** | **Average Max Martingale Bet Of each Iteration** |
|---|---|---|---|---|---|---|
| 2 | *39.250%* | 55.716% | 5.034% | -$259.84 | -$2,180,706,621.70 | $3,228,745,624.58 |
| 3 | *39.419%* | 55.544% | 5.037% | -$253.84 | -$3,140,158,182.11 | $3,677,777,005.67 |
| 4 | *39.314%* | 55.636% | 5.050% | -$256.77 | -$1,767,441,370.54 | $3,210,534,333.98 |
| 5 | *39.291%* | 55.673% | 5.036% | -$252.70 | -$805,016,516.89 | $2,321,835,284.06 |
| 6 | *39.406%* | 55.518% | 5.076% | -$246.28 | -$621,606,847.26 | $2,152,493,773.18 |
| 7 | *39.269%* | 55.666% | 5.065% | -$249.64 | -$1,032,522,524.08 | $2,408,740,519.36 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8 | *39.441%* | 55.352% | 5.207% | -$241.51 | -$827,263,952.20 | $1,548,928,165.93 |
| 9 | *39.535%* | 55.042% | 5.424% | -$230.42 | -$574,035,090.70 | $1,151,576,864.58 |
| 10 | *40.151%* | 54.109% | 5.740% | -$199.10 | -$427,831,848.15 | $772,658,841.56 |
| 11 | *41.057%* | 52.781% | 6.162% | -$164.27 | -$93,942,630.95 | $305,005,625.23 |
| 12 | *42.446%* | 50.938% | 6.616% | -$114.64 | -$121,080,473.14 | $210,673,694.63 |
| 13 | *42.601%* | 49.913% | 7.485% | -$91.80 | -$11,606,382.78 | $57,206,660.36 |
| 14 | *42.624%* | 48.833% | 8.543% | -$71.19 | $5,547,607.18 | $24,830,324.75 |
| 15 | *42.283%* | 48.152% | 9.565% | -$62.02 | -$16,564,106.48 | $16,063,310.80 |
| 16 | *41.672%* | 47.713% | 10.615% | **-$57.85** | -$73,399.49 | $3,198,068.27 |
| 17 | *40.562%* | 47.515% | 11.924% | -$61.39 | -$307,853.88 | $1,271,671.00 |
| 18 | *39.186%* | 50.022% | 10.792% | -$96.26 | -$412,400.41 | $2,653,157.14 |
| 19 | *35.187%* | 55.155% | 9.658% | -$170.74 | -$2,764,076.09 | $7,549,753.32 |
| 20 | *28.244%* | 63.473% | 8.283% | -$274.39 | -$40,486,477.95 | $44,416,730.28 |
| 21 | *14.786%* | 79.971% | 5.243% | -$396.68 | -$1,850,569,188.15 | $1,360,984,790.25 |

Code

```
#Import library and set seed
import random
import multiprocessing
import math
import time
import matplotlib.pyplot as plt
random.seed(1234)

#Set number of decks to be used in each game
```

```
num_decks = 4

#Create function to total hand value and optimize Ace
def total(hand):
    total=0
    for i, card in enumerate(hand):
        if card != "A":
            total+= card
        if card=="A":
            if total>= 11:
                total += 1
                hand[i] = 1
            else:
                total+=11
                hand[i]=11

    return total

#Creation of one deck, with traditional 52 card deck. Face
cards are added as "10" values.
def new_deck():
    std_deck = [
          # 2  3  4  5  6  7  8  9  10  J   Q   K   A
            2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, "A",
            2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, "A",
            2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, "A",
            2, 3, 4, 5, 6, 7, 8, 9, 10, 10, 10, 10, "A"]


        # add more decks
    std_deck = std_deck * num_decks

    random.shuffle(std_deck)

    return std_deck[:]

#Implements standard blackjack rules logic for player's
hand
def player_h(p_hand,cards,i,p_stand,soft):
    ncards = len(cards)
    stand = False
    while stand != True:
        #Total the current hand
        current = total(p_hand)
        #Stop if card index is equal to num of card in
deck (when deck runs out)
        if i == ncards:
            stand = True
        if soft == "N":
            if current < p_stand:
                i += 1
                p_hand.append(cards[i])
            else:
                stand = True
        elif soft =="Y":
            #Checking for a soft hand, when Ace is counted
as 11
            if "A" in p_hand and current == p_stand and
current > 11:
                p_hand.append(cards[i])
            elif current < p_stand:
                i += 1
                p_hand.append(cards[i])
            else:
                stand = True
    x = [current,i]
    return x

#Implements standard blackjack rules logic for player's
hand
def player_h(p_hand,cards,i,p_stand,soft):
    ncards = len(cards)
    stand = False
    while stand != True:
        #Total the current hand
        current = total(p_hand)

        #Stop if card index is equal to num of card in
deck (when deck runs out)
        if i == ncards:
            stand = True
        if soft == "N":
            if current < p_stand:
                i += 1
                p_hand.append(cards[i])
            else:
                stand = True
        elif soft =="Y":
            #Checking for a soft hand, when Ace is counted
as 11
            if "A" in p_hand and current == p_stand and
current > 11:
                p_hand.append(cards[i])
            elif current < p_stand:
                i += 1
                p_hand.append(cards[i])
            else:
                stand = True
    x = [current,i]
    return x

#Implements logic for dealer
def dealer_h(d_hand,cards,i):
    ncards = len(cards)
    stand = False
    while (stand != True):
        current = total(d_hand)
        if i == ncards:
            stand = True
        elif current <= 16:
            i += 1
            d_hand.append(cards[i])

        else:
            stand = True
    x = [current,i]
    return x

#Main function, performs game
def play_bj(cards,bet,p_stand,soft):
    ncards = len(cards)
    i = -1
    cwinnings = 0
    mwinnings = 0
    won = 0
    lost = 0
    draw = 0
    bet = bet
    unit = bet/2
    global prev_result
    prev_result = 0
    while i < ncards-20:
        if prev_result == 0:
            cbet = bet
            mbet = bet
        elif prev_result == -1:
            mbet = 2*mbet
        elif prev_result ==1:
            cbet = cbet+unit
        #make sure at least 4 cards left in deck
        if i <= (ncards-8):
            ## Initial deals
            i += 2
            p_hand = [cards[i-1],cards[i]]
            i += 2
            d_hand = [cards[i-1],cards[i]]

        else: #when deck is near empty
            return [won,lost,draw,winnings]

        p_res = player_h(p_hand,cards,i,p_stand,soft)

        i = p_res[1]
        d_res = dealer_h(d_hand,cards,i)
        #if player didn't go over 21, then move on to
dealer
        if p_res[0] < 22 :
            d_res = dealer_h(d_hand,cards,i)
            i = d_res[1]

        if p_res[0] == d_res[0]:
            draw += 1
            prev_result = 0
        elif p_res == [10,"A"] or p_res== ["A",10]:
            cwinnings += 3*(cbet/2)
            mwinnings += 3*(mbet/2)
            won += 1
            prev_result = 1
        elif p_res[0] > 21:
            cwinnings -= cbet
            mwinnings -= mbet
            lost +=1
            prev_result = -1
        elif d_res[0] > 21:
            cwinnings += cbet
            mwinnings += mbet
            won += 1
            prev_result = 1
        elif p_res[0] > d_res[0]:
            cwinnings += cbet
            mwinnings += mbet
            won += 1
            prev_result = 1
        elif p_res[0] < d_res[0]:
            cwinnings -= cbet
            mwinnings -= mbet
            lost +=1
            prev_result = -1
    return [won,lost,draw,cwinnings,mwinnings]

#Set number of simulations
N = 1000000

#Create function to run simulations using soft strategy
def softplays():
    test =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]
    resy = [0]*len(test)
    for k,j in enumerate(test):
        p_stand = j
        wins = [0]*N
        losses = [0]*N
        draws = [0]*N
        cwinnings = [0]*N
        mwinnings = [0]*N
        results = [0]*N

        for i in range(0,N):
            total_decks = num_decks*new_deck()
            results[i] =
play_bj(total_decks,10,p_stand,"Y")
            wins[i] = results[i][0]
            losses[i] = results[i][1]
            draws[i] = results[i][2]
            cwinnings[i] = results[i][3]
            mwinnings[i] = results[i][4]
            total = sum(wins)+sum(losses)+sum(draws)

        resy[k] = [j,round(sum(wins)/total,2),
round(sum(losses)/total,2), round(sum(draws)/total,2),

round(sum(cwinnings)/len(cwinnings),2),round(sum(mwinnings
)/len(mwinnings),2)]
    return resy

#Run soft simulations
softplays()

#Create function to run simulation using hard strategy
def hardplays():
    test =
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]
    resy = [0]*len(test)
    for k,j in enumerate(test):
        p_stand = j
        wins = [0]*N
        losses = [0]*N
        draws = [0]*N
        cwinnings = [0]*N
        mwinnings = [0]*N
        results = [0]*N
        bet = 10
        for i in range(0,N):
            total_decks = num_decks*new_deck()
            results[i] =
play_bj(total_decks,bet,p_stand,"N")
            wins[i] = results[i][0]
            losses[i] = results[i][1]
            draws[i] = results[i][2]
            cwinnings[i] = results[i][3]
            mwinnings[i] = results[i][4]
            total = len(wins)
        resy[k] = [j,round(sum(wins),2),
round(sum(losses),2), round(sum(draws),2),

round(sum(cwinnings)/len(cwinnings),2),round(sum(mwinnings
)/len(mwinnings),2)]
    return resy

#Run hard simulations
hardplays()

#Create table of winnings for a conservative betting
```

```
strategy on a hard 16 and plot winnings in histogram
x = 10000
winnings = [0]*x
results= [0]*x

for i in range(0,x):
    results[i] = play_bj(new_deck(),10,16,"N")
    winnings[i] = results[i][3]

plt.xlim([min(winnings)-5, max(winnings)+5])

plt.hist(winnings, density=True, bins=50)
plt.title('Winnings Distribution with Hard 16 Strategy and
Conservative Betting')
plt.xlabel('Winnings/Profit')
plt.ylabel('Count')

plt.show()

#Create function to compare winnings statistics over
different number of iterations
def softsixteen(iterations):

    resy = [0]*len(iterations)
    for k,j in enumerate(iterations):
        N = j

        p_stand = 16
        cwinnings = [0]*N
        mwinnings = [0]*N
        results = [0]*N
        bet = 10

        for i in range(0,N):
            total_decks = new_deck()
            results[i] =
play_bj(total_decks,bet,p_stand,"N")
            cwinnings[i] = results[i][3]
            mwinnings[i] = results[i][4]


        #Create table of mean, med, max, min, standard
deviation, variance of winnings
        resy[k] =
[j,"c",round(sum(cwinnings)/len(cwinnings),2),
                    statistics.median(cwinnings),
                    max(cwinnings),min(cwinnings),

statistics.stdev(cwinnings),statistics.variance(cwinnings)
,"m",round(sum(mwinnings)/len(mwinnings),2),
                    statistics.median(mwinnings),
                    max(mwinnings),min(mwinnings),

statistics.stdev(mwinnings),statistics.variance(mwinnings)
]
    return resy

softsixteen([1000,10000,100000])
```