How to Win Monopoly

(landing probability study using Python-based virtual Monopoly simulator)

New Mexico

Supercomputing Challenge

Final Report

April 8, 2020

Team 21

Los Alamos Middle School

Team Members

Asher Koh

Hyunoo Kim

Project Mentor

Aik-Siong Koh

S. Jun Kim

How to Win Monopoly

Executive Summary

Monopoly is a classic board game that you probably have played before. For some, it drives them insane. This game is a combination of the luck of the roll and strategic buying of properties. We used Python to simulate a Monopoly game of 150 moves which takes a little less than two hours to play in a real game. The purpose of this project is to figure out which spaces on a Monopoly Board are landed on with the most frequency. We eliminated the element of money so that we could focus on only the movement. For our procedure, we used math to get the dice probabilities where you rethrow on doubles with a maximum of three throws assuming you get doubles twice in a row. We also created a Python program to validate them. From our probabilities, we discovered that seven (17.1%), nine (12.1%), and eight (11.6%) are the most likely numbers to roll on two dice in Monopoly, so you should buy every property that is on a seventh, ninth, and eighth space after Go, Jail, Railroads, Boardwalk, Free Parking, and St. Charles Place because there are Chance and Community Chest Cards that send you to these places, and you start on Go. It is observed in our Virtual Monopoly simulator written in Python that every seventh space after Jail is the best, but the other locations could help because of Chance and Community Chest Cards. Every seventh, ninth, and eighth space after Jail are the best to buy because our simulation shows that Jail is the most frequently

landed on. Jail is very frequently landed on because there are multiple ways to get to Jail without cards. For example, rolling doubles three times in a row or landing on the go to jail space. In conclusion, our project is to help you solve your lifetime dilemma, and maybe even win a game of Monopoly.

Introduction

Monopoly is a very suspenseful and uneasy game, at least, to some of us. In 1930, during the Great Depression a man named Charles Darrow sketched out the street names for Atlantic City on the oilcloth covering his table. He also added the railroads that carried rich vacationers to Atlantic City, and the utility companies that serviced them. Short Line was not actually a railroad but a bus freighting company that had a stop in Atlantic City. After he sketched out the street names, he was given free samples of several colors at a local paint store, and he used them to paint his oilcloth, and a game began taking form in his mind. He cut houses and hotels for his game board city using scraps of wood molding a lumber yard had discarded. Then he used colored buttons for the game tokens, used play money, and a pair of dice. After that he, his wife, and his son would play it every evening. Soon friends started playing, and it became a standard feature at the Darrow's home. Then more people heard of it. A store in Philadelphia wanted some copies, and eventually the workload of making the copies was too much and Charles Darrow sold it to Parker Brothers. (From The Monopoly Book: Strategy and Tactics of the World's Most Popular Game By Maxine Brady)

We chose this project because one of us is really good at Monopoly, but wants to get as far ahead of everyone else as possible, and the other one always loses, and he wants to remedy that. This is why we settled on a project where both of us could benefit along with some other Monopoly players. The only type of strategy in Monopoly is the choice of accepting or declining an offer to buy the property you landed on or someone wants to sell to you.

Description

For our project, we programmed a simulation on Python of a 150 move Monopoly game without money just to see which spaces are landed on the most. We also wrote a program to calculate the probabilities of rolling certain numbers with the rules of Monopoly. For the simulation, we used the Tkinter module for visual aid [See Figure 1] and a system of functions and loops to run the main program. Dictionaries were also used for in Jail and out of Jail purposes and for coordinates. Each player is represented by a ball with a distinctive color. Our code can be divided into 5 main parts, each consisting of a list of smaller parts.

- Visual Aid graphic by Tkinter module
- Game function & variable setup
- Result function & probability count
- Ending the game and displaying a result plot
- Run main loop

Probabilities of two dice with maximum two rethrows on doubles

$$P(1)=0$$
$$P(2)=0$$
$$P(3)=2*P(1,2)=\frac{2}{36}$$
$$P(4)=2*P(1,3)=\frac{2}{36}$$
$$P(5')=2*P(1,4)+2*P(2,3)=\frac{4}{36}$$
$$P(5)=P(5')+P(1,1)*P(3)=\frac{4}{36}+\frac{1}{36}\frac{2}{36}=\frac{4*36+2}{36*36}=\frac{146}{36^2}$$
$$P(6')=2*P(1,5)+2*P(2,4)=\frac{4}{36}$$
$$P(6)=P(6')+P(1,1)*P(4)=\frac{4}{36}+\frac{1}{36}\frac{2}{36}$$
$$P(7')=2*P(1,6)+2*P(2,5)+2*P(3,4)=\frac{6}{36}$$
$$P(7)=P(7')+P(1,1)*P(5)+P(2,2)*P(3)+P(1,1)*P(1,1)*P(3)$$
$$=\frac{6}{36}+\frac{1}{36}\frac{146}{36^2}+\frac{1}{36}\frac{2}{36}+\frac{1}{36}\frac{1}{36}\frac{2}{36}$$
$$=\frac{6*36^2+146+2*36+2}{36^3}=\frac{7996}{36^3}$$
$$P(8')=2*P(2,6)+2*P(3,5)=\frac{4}{36}$$
$$P(8)=P(8')+P(1,1)*P(6')+P(2,2)*P(4)+P(1,1)*P(1,1)*P(4)$$
$$=\frac{4}{36}+\frac{1}{36}\frac{4}{36}+\frac{1}{36}\frac{2}{36}+\frac{1}{36}\frac{1}{36}\frac{2}{36}$$
$$=\frac{4*36^2+4*36+2*36+2}{36^3}=\frac{5402}{36^3}$$
$$P(9')=2*P(3,6)+2*P(4,5)=\frac{4}{36}$$
$$P(9)=P(9')$$
$$+P(1,1)*P(7')+P(2,2)*P(5')+P(3,3)*P(3)$$
$$+P(1,1)*P(1,1)*P(5')+2*P(1,1)*P(2,2)*P(3)$$
$$=\frac{4}{36}+\frac{1}{36}\frac{6}{36}+\frac{1}{36}\frac{4}{36}+\frac{1}{36}\frac{4}{36}+\frac{1}{36}\frac{1}{36}\frac{4}{36}+\frac{2}{36}\frac{1}{36}\frac{2}{36}$$
$$=\frac{4*36^2+6*36+4*36+4*36+4+4}{36^3}=\frac{5696}{36^3}$$
$$P(10')=2*P(4,6)=\frac{2}{36}$$

This is the math we used to get the dice probabilities, and we did this for all 36 numbers.

The first main part is where all the graphics are created for visual aid. (Importing modules are in this section of code too.) Not much else can be

explained about that except that the players are also made in this part. The second part took the longest and without it, the third, fourth, and fifth would be useless. In the second chunk of the code, we defined all our game functions. For example, a player_move() function was defined so our players could move. A roll_dice() function was created so dice could be randomly “rolled”. The dice follow the rules of Monopoly, so if you roll doubles you roll again, but if you roll doubles three times in a row you go to Jail (see Figure 2). Chance and Community chest cards were also accounted for in this part of the code using functions to send the player to the right place like Go, Jail, and Boardwalk depending on what it says. We executed them by using a check() function so that we could check if the players had landed on a card space. We also made it so that the cards would be shuffled, so it is more like a real game. For the get out of jail free card we made it so the player would immediately use the card when put in jail. The third part was put in last because we needed a working program before we could code it. The third part is where the results function is made. In this sector, we counted up all the times the 6 players landed on the forty spaces on the Monopoly board and displayed the outcome on a bar graph. This part was scattered throughout the program because we needed to count() every time a player moved to a different location. The fourth part, of course, is where we end the game and the results pop up. This part resets the game and puts all the players back at the GO space. A “games” setting allows the person running it to control the amount of games that it runs at a time and a “runSpeed” setting allows change between the interval of sleep time between moves. A “MaximumCurrentPlayer” variable

allows change between the number of players that actually move. Two lines can be commented out so that stack will be activated. (This just means that the count for each space will not be reset to zero every game). The last part is just the main loop where we run all the other functions in a neat, coordinated order so the final output becomes what you can experience as a virtual, Python based Monopoly game.

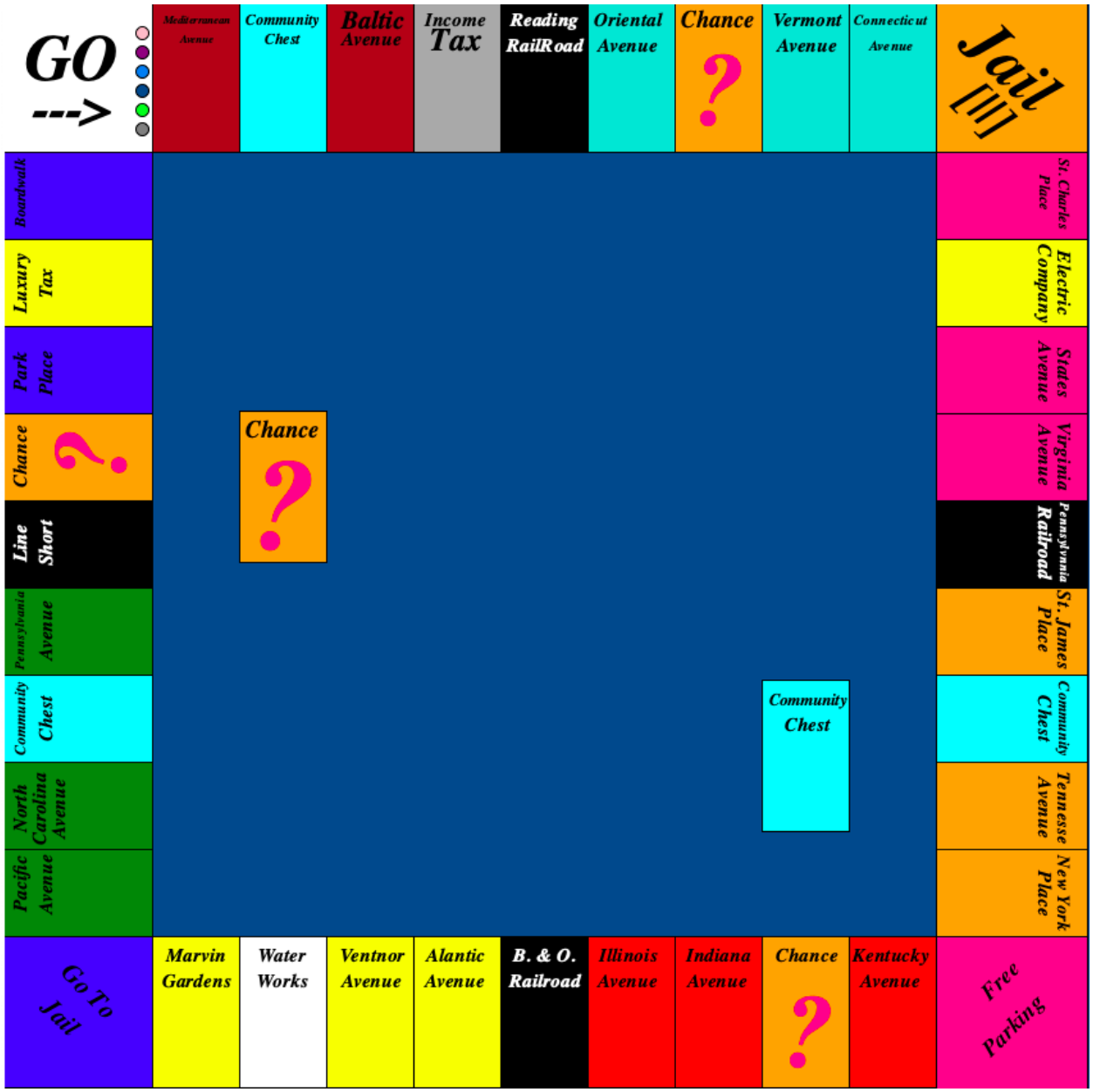


Figure 1. Tkinter graphic output with 6 players

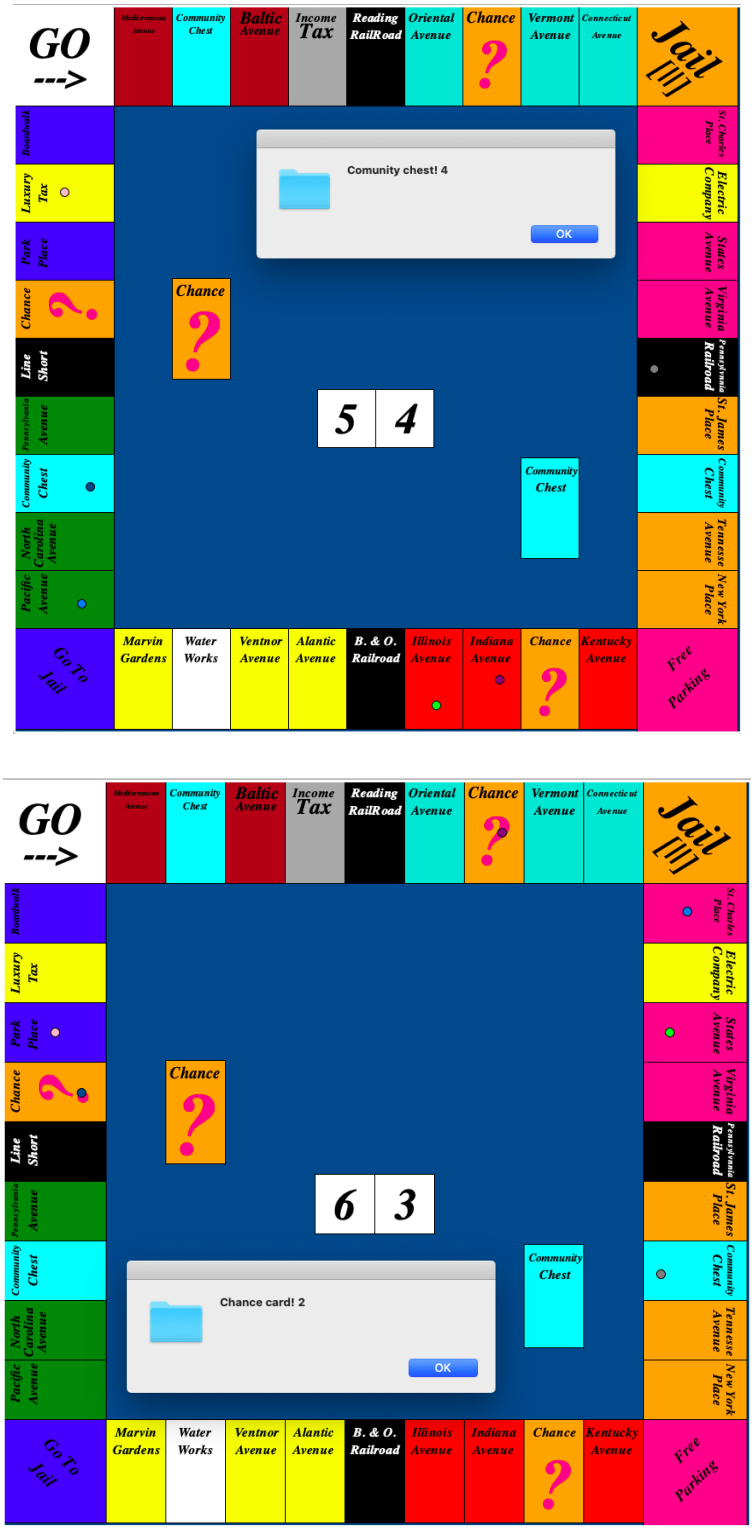


Figure 2. Screenshot of Community chest and Chance card pop-up

Results

For our results we found that seven, nine, and eight are the three most likely numbers to roll with the rules of Monopoly. That should mean that every property seven, nine, or eight after Go, Jail, St. Charles, Illinois Avenue, Boardwalk, and all Railroads because Chance and Community Chest cards send you to these places, and Jail is landed on a lot. For the program that we used to simulate the game we ran it 1000 times, and the results that we got were that Jail was landed on the most, St. James Place (2.68%) was the property that was landed on the most, Tennessee Avenue (2.66%) was landed on second to most, and New York Avenue tied with Pennsylvania Railroad (2.64%) for third in most landed on properties.

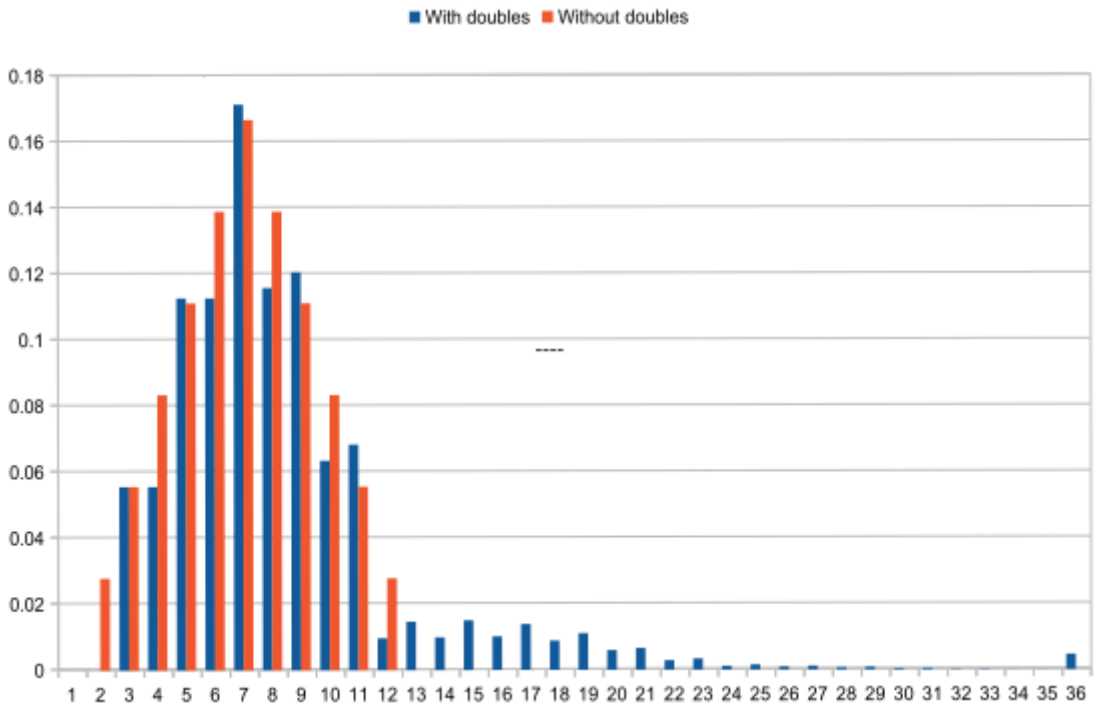
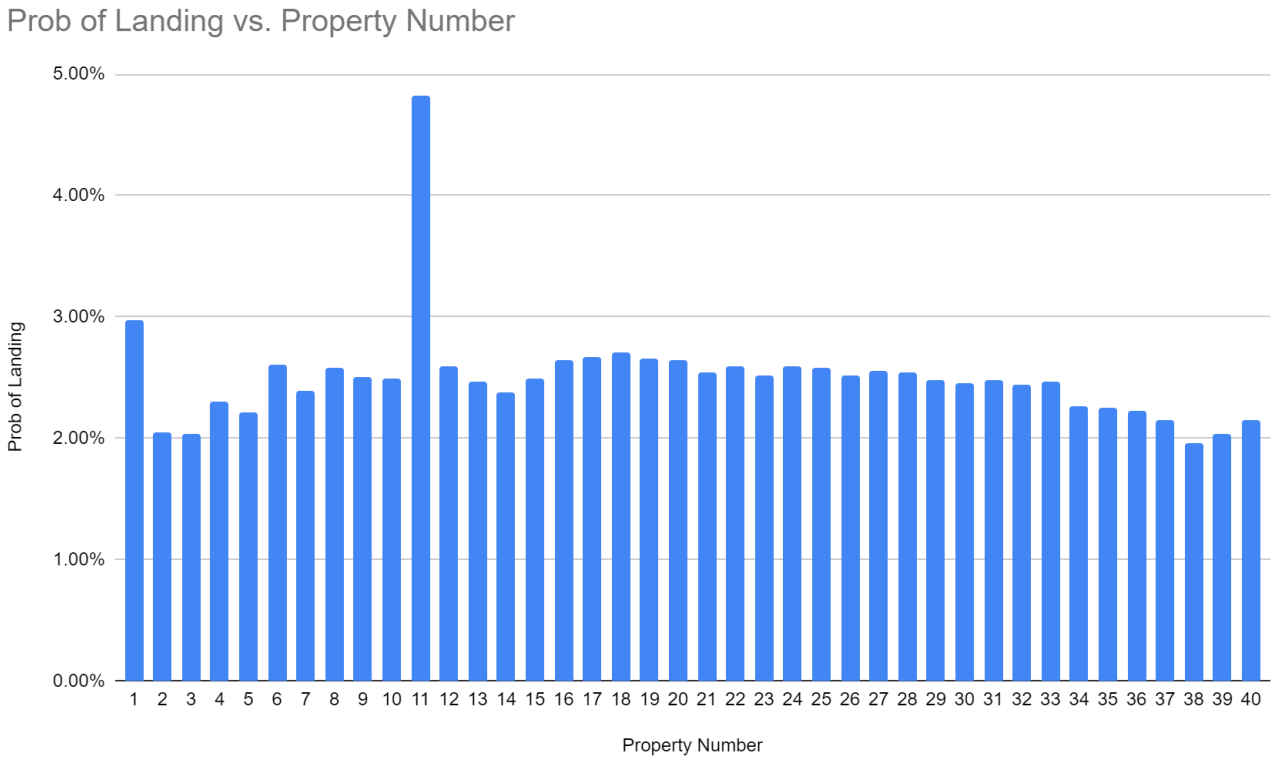


Figure 3. Random Dice rolling sum probability.

Here is a graph with the probabilities of two dice with doubles making you have to roll again according to the rules of Monopoly where 36 is the probability of going to jail from rolling doubles three times in a row. (This bar is higher because there is more than one way to get three doubles in a row). Next to the probabilities with double repeats in blue is the probability of two dice with no repeats after doubles in orange.



Here are the experimental probabilities of each Monopoly space on the board, and we ran a simulation of 1000 games. We used numpy.amax and numpy.where to find the highest value and to locate where it was in the graph. A note may be made about a slight bell curve appearing after GO and JAIL. This is a reasonable outcome because there is a bell curve in Figure 2 (Rolling Dice sum output). If GO and JAIL were landed on the most, then bell curves should start appearing there.

Conclusion

In our results we saw that Illinois Avenue, Marvin Gardens, and Ventnor Avenue are the properties that were the most frequently landed on. We can see that these results are sensible because St. James Place is six spaces away from jail, and that is close to the number seven which is the most likely number. Tennessee Avenue is eight spaces away from jail, and jail was landed on a lot, and eight is in the top three most likely numbers. New York Avenue is nine spaces from Jail, and nine is the second most likely number. Pennsylvania Railroad is fifteen spaces from Go and that is almost divisible by eight. We suggest that you get a monopoly with the orange, and railroads properties to win. Some of the areas on the Monopoly board that are landed on more frequently than others, and that's because those properties are usually almost divisible or divisible by seven, eight, or nine.

Appendix A.

Landing Probability calculation using Python

```
probability = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]

for i in range(1,7):
    for j in range(1,7):
        if i == j:
            for k in range(1,7):
                for l in range(1,7):
                    if k == l:
                        for m in range(1,7):
                            for n in range(1,7):
                                if m == n:
                                    probability[36] = probability[36]+(1/36**3)
                                else:
                                    probability[i+j+k+l+m+n] = probability[i+j+k+l+m+n]+(1/36**3)
                    else:
                        probability[i+j+k+l] = probability[i+j+k+l]+(1/36**2)
        else:
            probability[i+j] = probability[i+j]+(1/36)
print(probability)
print(len(probability))
```

```
sum = 0
for prob in probability:
    sum = sum + prob
print(sum)
for prob in probability:
    number=round(prob*36**3,0)
    print(number)
```

Appendix B.

Virtual Monopoly simulator python code

The original python code can download in the link below

https://www.dropbox.com/home/50.%20Hayden.K?preview=Monopoly20200406.ipynb

Appendix C.

Graph key

1 = Go

2 = Mediterranean Avenue

3 = Community Chest

4 = Baltic Avenue

5 = Income Tax

6 = Reading Railroad

7 = Oriental Avenue

8 = Chance

9 = Vermont Avenue

10 = Connecticut Avenue

11 = Jail

12 = St. Charles Place

13 = Electric Company

14 = States Avenue

15 = Virginia Avenue

16 = Pennsylvania Railroad

17 = St. James Place

18 = Community Chest

19 = Tennessee Avenue

20 = New York Place

21 = Free Parking

22 = Kentucky Avenue

23 = Chance

24 = Indiana Avenue

25 = Illinois Avenue

26 = B & O Railroad

27 = Atlantic Avenue

28 = Ventnor Avenue

29 = Water Works

30 = Marvin Gardens

31 = Go To Jail

32 = Pacific Avenue

33 = North Carolina Avenue

34 = Community Chest

35 = Pennsylvania Avenue

36 = Short Line

37 = Chance

38 = Park Place

39 = Luxury Tax

40 = Boardwalk

Bibliography

“In the Beginning, There Was the Great Depression.” *The Monopoly Book: Strategy and Tactics of the World's Most Popular Game*, by Maxine Brady, David McKay Company, Inc New York, 1974, pp. 14–24.

https://www.hasbro.com/common/instruct/monins.pdf

https://www.learnpython.org/

https://www.w3schools.com/

Stowell, Louie, et al. Coding for Beginners Using Python. Usborne, 2017.

https://wiki.python.org/moin/TkInter