**Tutorial 6: Simulations**
**Due: Friday, March 13, 2020**

## 1: Card Game

In a computer-based trading card game, players use in-game currency to draw cards. The probability of drawing each card type is as follows:

Common: 83.5%
Rare: 14%
Legendary: 2.5%

a) Develop a function that simulates drawing a card and returns the card type.
b) Players open packs of 30 cards at a time. Develop a program that asks the player how many card packs they would like to open, and then returns the number of each type of card that they won.

    **Example:**
    How many packs would you like to open? 3
    You have 71 common cards, 15 rare cards, and 4 legendary cards.

## 2: Dungeons and Dice

A Dungeons and Dragons dice set is typically made up of six types of dice.

1. 20-side die (D20)
2. 12-side die (D12)
3. 10-side die (D10)
4. 8-sided die (D8)
5. 6-sided die (D6) *this is a normal die*
6. 4-sided die (D4)

a) Develop a function to simulate rolling an n-sided dice.
b) Develop a function that simulates rolling all of the Dungeons and Dragons dice types, and returns the total sum of the dice values after the roll.
c) Use a Monte Carlo simulation to estimate the most likely sum of the dice values you would get if you roll the dice.

**3: Stock Market**

Develop a program to plot a graph of 100 Monte Carlo simulations of the stock prices over the next year for each of the following companies. You can use the stockMarket.py code to help you.

a) Netflix https://finance.yahoo.com/quote/NFLX/history?p=NFLX
b) Facebook https://finance.yahoo.com/quote/FB/history?p=FB
c) Amazon https://finance.yahoo.com/quote/AMZN/history?p=AMZN
d) Tesla https://finance.yahoo.com/quote/TSLA/history?p=TSLA

**4: Game Show Problem**

The “Monty Hall Problem” (https://en.wikipedia.org/wiki/Monty_Hall_problem) is a classic introduction to game theory and probability simulations that comes from a live TV game show in the 1960s. A contestant is faced with three doors. Behind one door is a very large prize (say, a car). The contestant tries to guess which door has the prize. We already know that mathematically the contestant should win 33% of the time.

a) Write a program that uses a Monte Carlo simulation to prove it. For each simulation run, the program should use a variable to represent the randomly chosen winning door number. The program should then randomly select a door number from one to three to represent the contestant’s choice. Finally, display the winning percentage after a large number of simulation runs, and confirm that it is very close to 33%.

b) The second part of the “Monty Hall Problem” is the most interesting. After selecting a door, but before the prize door is revealed, the contestant is shown a losing door. The contestant now has the option of switching to the other, as-yet unopened door. Should the contestant stick with their original choice, switch to the other unopened door, or does it not matter? Modify your program from #2 to always choose to switch to the other door, and display the winning percentage after a large number of simulation runs. What would you conclude to be the best strategy?