Technical Goal of Project:

Write Python script that will test my Roulette system against a one million spin dataset to try and cash in on the new casino in Everett.

The inputs for the simulation will be 1) an established number of bets to be made in each betting 'round', 2) a dollar amount of cash, and 3) the dataset itself.

The script will iterate through the million spins, using the first spin as the starting point for the first 'round' of betting, the second spin as the starting spin for the second 'round', and so forth. The results for each round will be recorded and analyzed, so we can get a sense for whether my roulette system offers any real value.

The Roulette System:

If you bet even or odd in Roulette, the odds of a win are slightly worse than ½, and the payout for a win is double your bet. My system calls for you to 1) bet $1 on odd, 2) whenever you win pocket your winnings and then bet $1, and 3) whenever you lose, triple your last bet.

If you look at the math quickly, you will see that my system sets it up so that no matter how many times you lose, you only need to win once to win everything back and more. Let's say you lose your first bet (net loss = -1) and then your second bet, which of course tripled (-1 - 3 = net loss = -4). Your net loss is now -4, but your next bet is 9 - you win. So you are up 5. This math carries out to eternity, meaning if you had unlimited money, you will eventually walk away a winner. Of course you don't. That's why we have a starting bankroll in our script. And also, casinos have bet limits. We implemented a maximum bet of $10k, which is pretty standard.

To start off, let's see how we do with a $1000 starting bankroll, betting on 100 spins in each 'round'

The Results:

Of the 1 million betting 'rounds' we performed, 729,310 resulted in us ending with more money than we started with. The average winning bank was $3,140 and the median winning bank was $2,546. The overall win rate was 72.9%. In 26% of cases, we lost everything.

The base bet that we were utilizing (betting on odd) has odds of ½ and a payout of double your bet. Our system put $1000 on the line, offered an average win of $3,140 (more than triple the bet) and had a win rate of 72.9%, a much better proposition than the base bet! It is clear from the large increase in both win rate and win *amount* that our strategy offers significant advantages over placing a single, isolated bet.

We calculated a net value of $2,290 for a full round of our system which, at a $1,000 buy-in, is very attractive.

I'll see you in Everett.