0.0.1 Question 2a)

Let X be a random variable that gives your winnings if you bet on red and the roulette wheel is spun once.

i). What is the probability distribution of X? Give your answer as a table.

ii). Calculate the expected value of your winnings by betting on red.

Write up your full solution in the SAME box below using LaTeX (not code). Show all steps fully justifying your answer.

0.0.2 Part i

In roulette, there are a total of 18 spots that are red out of 38 total spots. This means, roughly, someone has a 18 / 38 chance of their spin landing on a red spot. Here, we are assuming that X is the original wager that is placed on a given color

Color	X (Winnings)	Probability
Red	+X	18 / 38
Black	-X	18 / 38
Green	-X	2 / 38

0.0.3 Part ii

For discrete values, the expectation value of a variable x can be calculated with:

$$E(x) = \sum_{i=1} x_i p_i.$$

In this case, the probability a ball landing on a red pocket is $\frac{18}{38} \approx 0.474$. If we take into account of not landing on red, this probability is then $\frac{20}{38} \approx 0.526$. If we assume that we are placing a \$1 bet each time, then the expected winnings of just betting on Red are

$$E(1) = \$1 \cdot \frac{18}{38} - \$1 \cdot \frac{20}{38} = -\$\frac{2}{38} \approx -\$0.05.$$

This means, if we repeatedly place a \$1 bet on Red, we can expect to lose about 5 cents every time we place a bet on Red.

0.0.4 Question 2b)

Let's simulate this. In the first code box below, write code to simulate one spin of a roulette wheel. Your output should be a string in the form of the number then the color (i.e. `18R` or `00G`)

In the 2nd code box below, write code that takes the number of spins and either the color red or black as input, calculates winnings for each spin assuming you bet on that color for all spins, and then outputs the average winnings out of those spins.

Then run the simulation 3 different times for `num_spins` = 100,000 and compare to your answer from part A.

To receive credit you must write your code such that all lines are visible in your PDF output.

In [20]:
```
def spin_roulette():
    spotNum = np.random.randint(1,39)
    ret = ""
    if ((1 <= spotNum <= 10) or (19 <= spotNum <= 28)):
        if (spotNum % 2 == 0):
            ret = str(spotNum) + "B"
        else:
            ret = str(spotNum) + "R"
    elif ((11 <= spotNum <= 18) or (29 <= spotNum <= 36)):
        if (spotNum % 2 == 0):
            ret = str(spotNum) + "R"
        else:
            ret = str(spotNum) + "B"
    else:
        if (spotNum == 37):
            ret = "0G"
        else:
            ret = "00G"
        return ret
# Your code above this line

spin_roulette()
```

Out[20]: '29B'

In [21]:
```
def color_winnings(color, num_spins):
    winnings = 0
    ret = 0
    for i in (range(num_spins)):
        currentSpin = spin_roulette()
        if (currentSpin[-1] == color):
            winnings += 1
        else:
```

```
                winnings -= 1
            ret = np.round(winnings / num_spins, 5)
            return ret

        # Your code above this line

        print("E[Winnings] = {:.3f}".format(color_winnings(color="R", num_spins=int(1e6))))
        print("E[Winnings] = {:.3f}".format(color_winnings(color="R", num_spins=int(1e6))))
        print("E[Winnings] = {:.3f}".format(color_winnings(color="R", num_spins=int(1e6))))
```

```
E[Winnings] = -0.054
E[Winnings] = -0.053
E[Winnings] = -0.050
```

0.0.5 Question 2c)

In Roulette you can bet on one of three "dozens" segments, called 1st 12, 2nd 12, and 3rd 12. They cover 1-12, 13-24, and 25-36, respectively. If you bet $1 on the first dozen (or second dozen, or third dozen) nonzero numbers and win, then you win $2 (i.e. you get your original dollar back, plus another $2.

Let Y be a random variable that gives your winnings if you bet on any one of the three "dozen" nonzero numbers and the roulette wheel is spun once.

i). What is the probability distribution of Y? Give your answer as a table.

ii). What is $E[Y]$?

Write up your full solution in the SAME box below using LaTeX (not code). Show all steps fully justifying your answer.

0.0.6 Part i

In Roulette, there are three separate dozen increments: 1-12, 13-24, and 25-36. The probability of our ball landing in any one of three dozen increments is $\frac{12}{38}$ where we still have to account for the probability of our ball landing in a Green pocket. If we assume someone is placing a wager Y on landing in one of these pockets, then the probability distribution is then

Outcome	Y (Winnings)	Probability
Win(1-12, 13-24, 25-36)	+2Y	12 / 38
Lost(0,00, or other dozens)	-Y	26 / 38

0.0.7 Part ii

In this scenario, the probability of our ball landing in one of three dozens that we bet that it would is $\frac{12}{38} \approx 0.316$ and $\frac{26}{38} \approx 0.684$ if it lands in one of the other 2 dozens or the Green pockets. Assuming that someone is placing a $1 bet on their ball landing in the dozen that they bet on, then the expectation value is then

$$E(y) = \$2 \cdot \frac{12}{38} - \$1 \cdot \frac{26}{38} = -\$\frac{2}{38} \approx -\$0.05.$$

This means, for everyone $1 that someone is betting on one of the three possible dozens, they can expect to lost $0.05.

0.0.8 Question 2d)

Write code to simulate `num_spins` spins, record the winnings for each spin if you bet on the first dozen nonzero numbers, and calculate the average winnings out of the total spins.

Then run the simulation 3 different times for `num_spins` = 100,000 and compare to your answer from part C.

```
In [22]: def dozen_winnings(num_spins):
             winnings = 0
             ret = 0
             for i in range(num_spins):
                 currentSpin = spin_roulette()
                 currentSpin = int(currentSpin[0:-1])
                 if (1 <= currentSpin <= 12):
                     winnings += 2
                 else:
                     winnings -= 1
             ret = np.round(winnings / num_spins, 5)
             return ret
         # Your code above this line

         print("E[Winnings] = {:.3f}".format(dozen_winnings(num_spins=100000)))
         print("E[Winnings] = {:.3f}".format(dozen_winnings(num_spins=100000)))
         print("E[Winnings] = {:.3f}".format(dozen_winnings(num_spins=100000)))
```

```
E[Winnings] = -0.050
E[Winnings] = -0.062
E[Winnings] = -0.049
```

Question 2e)

Recall, we showed in class that the expected winnings if you bet on any number is also $-\frac{1}{19}$.

So you're hopefully onto the pattern by now. The payouts in Roulette are designed so that the expected payout for a winning bet is always $-\frac{1}{19}$.

Since we define these payouts in terms of your winnings after betting \$1, we can think of these as payout odds.

For example, since if you bet \$1 on the first dozen nonzero numbers and win, then you win \$2, we say the odds are 2 to 1 (denoted 2:1).

The odds are 35:1 for landing on any particular number. This means if you bet \$1, you'll win \$35.

Suppose the casino wanted to develop odds for a new bet in Roulette, where they allow you to bet on any set of 3 different numbers. Let the odds for this new bet be

$$x : 1$$

What should x be so that the expected payout for a winning bet is still $-\frac{1}{19}$?.

Show your work using LaTeX below.

We know from the problem statement that the expected payout for a winning bet needs to be $-\frac{1}{19}$, we just need to figure out the payout for a win in this case. The probability of winning on any set of 3 different numbers is $\left(\frac{3}{38}\right)$. Using this our expectation value expression is then

$$E(x) = x \cdot \frac{3}{38} - 1 \cdot \frac{35}{38} = -\frac{1}{19}.$$

We now just need to solve for x. Doing so we then have

$$-\frac{2}{38} = x \cdot \frac{3}{38} - 1 \cdot \frac{35}{38} \quad \text{(Expectation Value)} \tag{1}$$

$$\frac{33}{38} = x \cdot \frac{3}{38} \quad \text{(Simplification)} \tag{2}$$

$$\frac{33}{38} \cdot \frac{38}{3} = x \quad \text{(Simplification)} \tag{3}$$

$$\frac{11 \cdot 3}{3} = x \quad \text{(Simplification)} \tag{4}$$

$$11 = x \quad \text{(Final Simplification)} \tag{5}$$

This means, when the casino is going to allow players to wager on hitting any set of three different numbers,

the casino should have a payout of $11:1$ to keep the expected payout for a winning bet of $-\frac{1}{19}$.

Question 2f)
Let's generalize this!

Define a function $x(n)$ that describes the odds the casino should give for betting \$1 on any set of n numbers if the casino wants to keep the expected payout for a winning bet at $-\frac{1}{19}$ for any n. (For example, the odds for betting on any 3 different numbers should be set at $x(3)$ to 1. The odds for betting on any 4 different numbers should be set at $x(4)$ to 1).

To achieve this, we essentially need our expectation value to follow the form of

$$x \cdot \frac{n}{38} - 1 \cdot \frac{38-n}{38} = -\frac{1}{19}.$$

From the above expression, we just need to solve for x. Doing this we then have

$$x \cdot \frac{n}{38} - 1 \cdot \frac{38-n}{38} = -\frac{1}{19} \qquad \text{(Original Expression)} \tag{6}$$

$$x \cdot \frac{n}{38} = \frac{38-n}{38} - \frac{1}{19} \qquad \text{(Simplification)} \tag{7}$$

$$x = \frac{38}{n}\left(\frac{38-n}{38} - \frac{1}{19}\right) \qquad \text{(Simplification).} \tag{8}$$

From the above, we can say that our final expression for $x(n)$ is then

$$x(n) = \frac{38}{n}\left(\frac{38-n}{38} - \frac{1}{19}\right).$$

If we baseline this and check it for when we just bet on one number, the payout should be 35:1. Let's check,

$$x(1) = \frac{38}{1}\left(\frac{38-1}{38} - \frac{1}{19}\right) = 38\left(\frac{37}{38} - \frac{2}{38}\right) = 38 \cdot \frac{35}{38} = 35.$$

We can see, that this works for $x(1)$, and then should work for other values as well.

Answer all of the parts below in the SAME cell below using LaTeX. Show all of your steps.

3a). Determine the value of a such that this defines a valid probability distribution. Use that value for the rest of the problem.

3b). Calculate $P(X \leq 3)$.

3c). What is $E[X]$? (Show steps calculating this).

3d). What is the standard deviation of X? (Show all steps calculating this).

Answer all of the parts above in SINGLE cell provided below using LaTeX.

0.0.9 Part A

For this probability distribution to be valid, we constitute that $P(x = 2) + P(x = 3) + P(x = 4) = 1$, essentially all of the probabilities need to add up to 1 for all values of k. For starters, we can simplify the the probability distribution in a simpler form:

$$P(X = k) = 2ak(k - 1).$$

If we now plug in all the values for k into this expression, sum them, and set them equal to 1 we then have

$$P(X = 2) + P(X = 3) + P(X = 4) = 2a(2)(2 - 1) + 2a(3)(3 - 1) + 2a(4)(4 - 1) = 1.$$

We then can solve this for a:

$$\begin{aligned}
2a(2)(2-1) + 2a(3)(3-1) + 2a(4)(4-1) &= 1 && \text{(Original Expression)} && (9)\\
2a(2)(1) + 2a(3)(2) + 2a(4)(3) &= 1 && \text{(Simplification)} && (10)\\
2a(2) + 2a(6) + 2a(12) &= 1 && \text{(Simplification)} && (11)\\
4a + 12a + 24a &= 1 && \text{(Simplification)} && (12)\\
40a &= 1 && \text{(Simplification)} && (13)\\
a &= \frac{1}{40} && \text{(Simplification).} && (14)
\end{aligned}$$

This then means that for the probability distribution to be valid, we constitute that $a = \frac{1}{40} = 0.025$.

0.0.10 Part B

To calculate $P(X \leq 3)$, we use

$$P(X \leq 3) = P(X = 2) + P(X = 3) = 2(0.025)(2)(2-1) + 2(0.025)(3)(3-1) = 0.1 + 0.3 = 0.4.$$

So, this means $P(X \leq 3) = 0.4$. Consequently, we can deduce that $P(X = 2) = 0.1$, $P(X = 3) = 0.3$, and $P(X = 4) = 0.6$.

0.0.11 Part C

We can calculate the expectation value for this probability distribution with

$$E(x) = \sum_{i=2}^{n} x_i p_i.$$

This then means that that the expectation value for $x = k$ is

$$E(x) = x_2 p_2 + x_3 p_3 + x_4 p_4 = 2(0.1) + 3(0.3) + 4(0.6) = 0.2 + 0.9 + 2.4 = 3.5.$$

0.0.12 Part D

The formula for calculating the standard deviation is

$$\sigma = \sqrt{\nu^2}$$

where ν is defined as the variance. The variance is calculated with

$$\nu^2 = \sum_i (x_i - \mu)^2 \cdot P(X = x_i).$$

This then means that the standard deviation is

$$\sigma = \sqrt{\sum_i (x_i - \mu)^2 \cdot P(X = x_i)} \quad \text{(Standard Deviation Formula)} \quad (15)$$

$$= \sqrt{(2-3.5)^2(0.1) + (3-3.5)^2(0.3) + (4-3.5)^2(0.6)} \quad \text{(Plugging In Values)} \quad (16)$$

$$= \sqrt{0.45} \quad \text{(Simplification)} \quad (17)$$

$$= 0.6708 \quad \text{(Simplification).} \quad (18)$$

This means the standard deviation is then $\sigma = 0.6708$.

0.0.13 Question 3e

Plot a histogram of the discrete probability distribution for X.

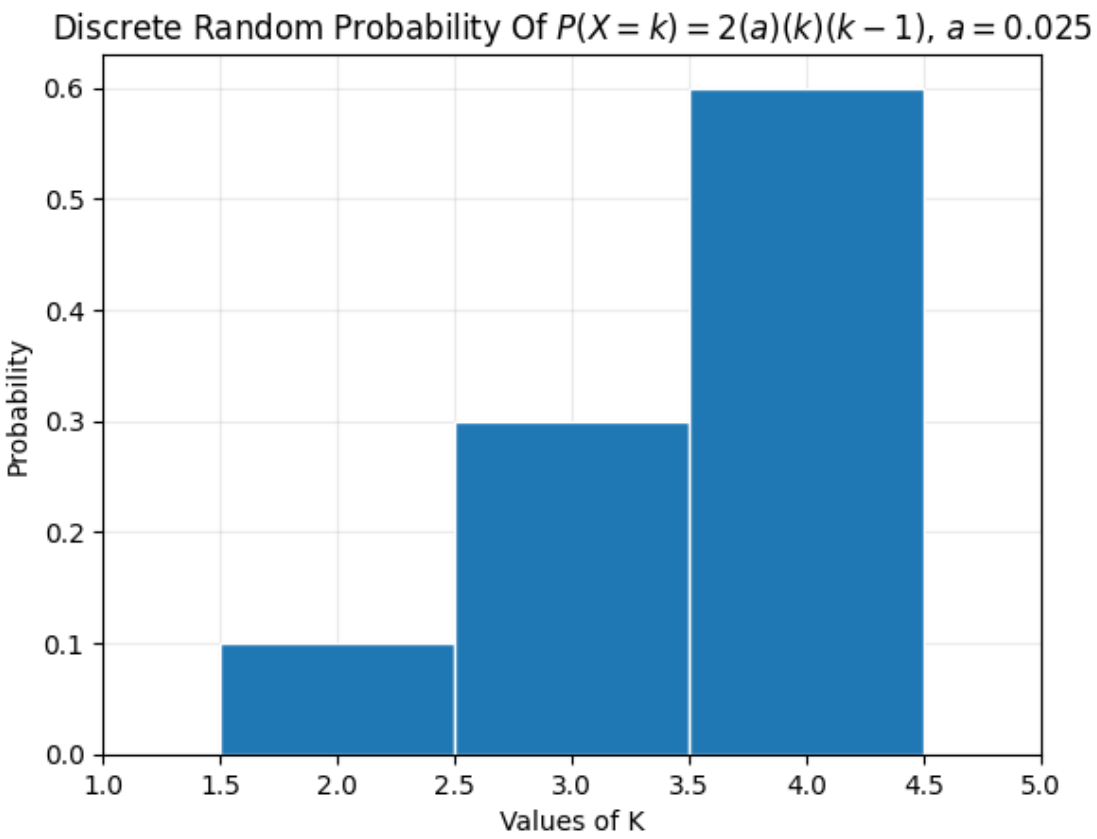
Use the same plotting guidelines as shown in Problem 1 so we can interpret area in the histogram as representing probability: - Set the bin widths to be equal to 1 - Add white lines between each bar

Be sure to include a title on your plot.

```
In [25]: k = np.arange(2,5,1)
         p = 2 * (0.025) * (k) * (k - 1)

         fig, ax = plt.subplots()

         ax.bar(k, p, width=1, ec='white');
         ax.set_axisbelow(True)
         ax.grid(alpha=0.25)
         plt.xlim(1,5)
         plt.xlabel(f"Values of K")
         plt.ylabel(f"Probability")
         plt.title(f"Discrete Random Probability Of $P(X = k) = 2(a)(k)(k - 1)$, $a = 0.025$");
         # Your code for the histogram above this line
```


Discrete Random Probability Of $P(X = k) = 2(a)(k)(k-1)$, $a = 0.025$
Probability
0.0
0.1
0.2
0.3
0.4
0.5
0.6
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Values of K

Answer all of the parts below in the SAME cell below using LaTeX. Show all of your steps.

4a). What is the probability that exactly 6 customers pass through John's line in the next 10 minutes?

4b). What is the probability that exactly 6 customers pass through the self check-out in the next 10 minutes, assuming that it is working?

4c). What is the probability that exactly 6 customers pass through the self check-out in the next 10 minutes, assuming that it is frozen?

4d). Use your results from 4b and 4c and the Law of Total Probability to calculate the probability that the self check-out tends exactly 6 customers in the next 10 minutes. Show all steps using LaTeX.

Answer all of the parts above in SINGLE cell provided below using LaTeX.

0.0.14 Part A

To answer these questions, we need to use the Poisson probability function, namely:

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

where λ is the average rate of events per interval and k is the number of events. For this part, where $k = 6$, the probability is

$$P(X = 6) = \frac{4^6 e^{-6}}{6!} = 0.014013457.$$

0.0.15 Part B

To answer this part of the question, we use the Poisson probability function again:

$$P(X = 6) = \frac{5^6 e^{-5}}{6!} = 0.1462228081.$$

0.0.16 Part C

We once again use the Poisson probability function again:

$$P(X = 6) = \frac{1^6 e^{-1}}{6!} = 0.0005109436683.$$

0.0.17 Part D

The probability that the self check-out tends exactly 6 customers in the next 10 minutes is

$$P(X = 6) = P(\text{Working}) \cdot P(X = 6|\text{Working}) + P(\text{Frozen}) \cdot P(X = 6|\text{Frozen}).$$

Plugging in the values from this we then have

$$P(X = 6) = 0.9 \cdot \frac{5^6 e^{-5}}{6!} + 0.1 \cdot \frac{1^6 e^{-1}}{6!} = 0.1316516216927174. \tag{19}$$

0.0.18 Question 4e)

S'pose John is working a 5-hour shift from 4-9 PM after school. He gets no breaks, because the year is 1870 and worker's rights is not yet a thing.

Plot a histogram of the probability distribution of the number of customers he serves in his 5 hour shift. For the **domain of the histogram, include x values between 75 and 160 in your plot.**

Hint: Python has a built-in function to calculate the Poisson distribution for different values of μ. See the documentation for `poisson.pmf` in `scipy.stats` (https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.poisson.html)

Hint: Since we are changing the time interval over which we are counting customers, you will need to update the parameter μ in the Poisson distribution to be the average number of customers John can serve in a 5-hour shift. You can assume that his rate of 4 customers per 10 minutes scales up consistently during his 5 hour shift.

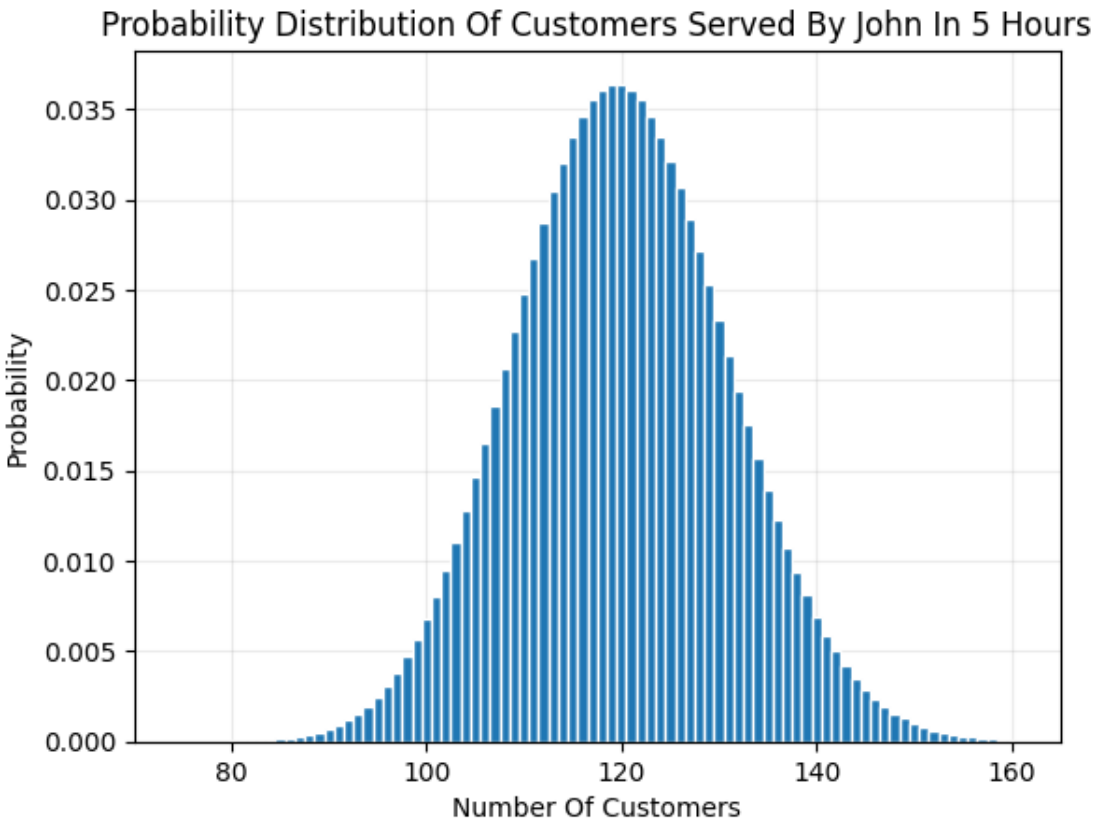
```
In [28]: from scipy.stats import poisson
```

```
In [29]: mu = 4 * 30
         domain = np.arange(75, 161)
         prob = poisson.pmf(domain, mu)

         fig, ax = plt.subplots()

         ax.bar(domain, prob, width=1, ec='white');
         ax.set_axisbelow(True)
         ax.grid(alpha=0.25)
         plt.xlim(70,165)
         plt.xlabel(f"Number Of Customers")
         plt.ylabel(f"Probability")
         plt.title(f"Probability Distribution Of Customers Served By John In 5 Hours");

         # your code above this line
```


Probability Distribution Of Customers Served By John In 5 Hours
Probability
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
80
100
120
140
160
Number Of Customers

0.0.19 Question 4g)

Time to simulate!

Recall from lecture that if the number of random events follows a Poisson distribution, the lapse of time between these events follows an Exponential distribution. For example, if the number of occurrences per 10 minute interval is distributed $X \sim Pois(4)$, then the time (in units of 10 minutes) between arrivals is $Y \sim Exp(4)$.

We're going to simulate the number of customers served using this knowledge.

i). Write a function `checkout_count` to simulate the number of customers served by the **self check-out machine** in a **5-hour** shift.

Your function should take as input the time length `time_len`, for calculating the arrivals, the working and broken customer arrival rate parameters (based on the time length given), and the probability, `p` that the machine is working properly.

Your function should simulate customer arrival times at the front of the line by sampling between-customer times from $Exp(\lambda)$ via Numpy's random.exponential function, where the argument λ will depend on the state of the machine (working or broken). Read the documentation carefully for the format of the input for the exponential function in Numpy.

Your simulation should model the arrival of each new customer, and sample whether or not the machine is working properly for each new customer.

Your function should **return the number of customer arrivals in a 5-hour shift**.

Make sure all code is visible in your PDF, or you won't receive points for this problem

ii). Use 10,000 simulations of this function to estimate the probability of the self check-out machine serves 100 or more customers in a 5-hour shift, and report your result.

iii). Finally, use 10,000 simulations of **this same function** to verify your answer to **Part 4e**.

```
In [32]: def checkout_count(time_len, rate_work, rate_broken, p):
             '''
             time_len    = time interval (minutes)
             rate_work   = rate when machine is working (customers/time unit)
             rate_broken = rate when machine is broken (customers/time unit)
             p           = probability machine is working
             '''
             total_time = 0
             customer_count = 0
             while (total_time < time_len):
```

```
            working = np.random.rand() < p
            if working:
                rate = rate_work
            else:
                rate = rate_broken
            next_customer = np.random.exponential(scale=1 / rate)
            total_time += next_customer
            if total_time < time_len:
                customer_count += 1
        return customer_count
        # your code for part i above here
```

In [33]:
```
time_len = 5 * 60
rate_work = 5
rate_broken = 1
p = 0.9
simulations = 10000
count_100_or_more = 0
for i in range(simulations):
    served = checkout_count(time_len, rate_work / 10, rate_broken / 10, p)
    if served >= 100:
        count_100_or_more += 1
prob_100_or_more = count_100_or_more / simulations
print(f"Probability of 100 or more customers: {prob_100_or_more}")
# Your code for part ii above this line
# Output should be approximately 0.70 if code is correct.
```

```
Probability of 100 or more customers: 0.7034
```

In [34]:
```
time_len = 5 * 60
rate_work = 4
rate_broken = 4
p = 1
simulations = 10000
count_100_or_more = 0
for i in range(simulations):
    served = checkout_count(time_len, rate_work / 10, rate_broken / 10, p)
    if served >= 100:
        count_100_or_more += 1
prob_100_or_more = count_100_or_more / simulations
print(f"Empirical Probability: {prob_100_or_more}")
print(f"Theoretical Probability: {1 - poisson.cdf(99, 4 * 5 * 6)}")
# Your code for part iii above this line
# Output should match your theoretical answer to Part 4f
```

```
Empirical Probability: 0.972
Theoretical Probability: 0.9721362601094794
```

QUESTION 4h: Comment on the results you found above in **Parts F and G** comparing the probabilities that John and the self check-out machine will serve 100 or more customers in a 5-hour block. Which seems like a better investment for the grocery store? Justify your answer.

From parts F and G, we can see that the empirical results are extremely close to the theoretical results from part 4f. This shows that John is on par with the theoretical average for the probability of him serving over 100 customers in a 5 hour period. We can also see that the probability of the self check-out machine serving over 100 students is a lot less than that of John's, with a probability of about 0.70. This means the better investment for the grocery store (assuming they never give John breaks) would be for them to clone John and have as many Johns as possible. John is the better investment.