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.

i).

k	P(x=k)
1	$\frac{18}{38}$
-1	$\frac{20}{38}$

ii). $E[X] = (1)(\frac{18}{38}) + (-1)(\frac{20}{38}) = \frac{-2}{38}$

0.0.2 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():
             wheel = ['00G','0G', '1R', '2B', '3R', '4B', '5R', '6B', '7R', '8B',
                      '9R', '10B', '11B', '12R', '13B', '14R', '15B', '16R', '17B', '18R',
                      '19R', '20B', '21R', '22B', '23R', '24B', '25R', '26B', '27R', '28B',
                      '29B', '30R', '31B', '32R', '33B', '34R', '35B', '36R']

             return np.random.choice(wheel)


         # Your code above this line

         spin_roulette()
```

```
Out[20]: '31B'
```

```
In [49]: def color_winnings(color='R', num_spins=100000):

             numerator = 0;
             for ii in range(num_spins):
                 spin = spin_roulette();
                 c = spin[-1]
                 if (c == color):
                     numerator+=1
                 else:
                     numerator-=1
                 denom=ii+1

             return numerator / denom
         # 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] = -0.052
E[Winnings] = -0.052
E[Winnings] = -0.053
```

0.0.3 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.

i).

k	P(Y = k)
2	$\frac{12}{38}$
-1	$\frac{26}{38}$

ii). $(2)(\frac{12}{38}) + (-1)(\frac{26}{38}) = \frac{-2}{38}$

0.0.4 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 [50]: def dozen_winnings(num_spins):
             first = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12']
             numerator = 0;
             for ii in range(num_spins):
                 n = 0;
                 spin = spin_roulette()
                 if len(spin) == 3:
                     n = spin[0:2]
                 else:
                     n = spin[0]

                 if (n in first):
                     numerator+=2
                 else:
                     numerator-=1
                 denom=ii+1

             return numerator / denom
         # 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] = -0.048
E[Winnings] = -0.055
E[Winnings] = -0.050
```

Question 2e)
Recall, we showed in class that the expected winnings if you bet on any number is also $-\dfrac{1}{19}$.

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 $-\dfrac{1}{19}$.

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 $-\dfrac{1}{19}$?.

Show your work using LaTeX below.

The probability of choosing one of the 3 winning numbers is $\frac{3}{38}$

To calculate the expected value:

Expected Value = (Probability of Winning) * (Winnings) + (Probability of Losing) * (Losses)

So for our new bet, we'd calculate:

$-\frac{1}{19} = (\frac{3}{38}) * (W) + (\frac{35}{38}) * (-1)$

$\implies \frac{35}{38} - \frac{1}{19} = (\frac{3}{38}) * (W)$

$\implies \frac{35}{38} - \frac{2}{38} = (\frac{3}{38}) * (W)$

$\implies \frac{33}{38} = (\frac{3}{38}) * (W)$

$\implies \frac{33}{38} * \frac{38}{3} = W$

$\implies 11 = W$

Thsu X should be 11.

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).

The probability of choosing one of the n winning numbers is $\frac{n}{38}$

So for our new bet, we'd calculate:

$-\frac{1}{19} = (\frac{n}{38}) * (W) + (\frac{38-n}{38}) * (-1)$

$\implies \frac{38-n}{38} - \frac{1}{19} = (\frac{n}{38}) * (W)$

$\implies \frac{38-n}{38} - \frac{2}{38} = (\frac{n}{38}) * (W)$

$\implies \frac{36-n}{38} = (\frac{n}{38}) * (W)$

$\implies \frac{36-n}{38} * \frac{38}{n} = W$

$\implies \frac{36-n}{n} = W$

Thus $x(n) = \frac{36-n}{n}$

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.

3a). To ensure a valid probability distribution, we have to check that they all add up to 1 and none of the probabilities are negative.

Since $k = 2, 3, 4$ we can calculate the probability of each k:

$P(X = 2) = 2a2^2 - 2a2 = 8a - 4a = 4a$

$P(X = 3) = 2a3^2 - 2a3 = 18a - 6a = 12a$

$P(X = 4) = 2a4^2 - 2a4 = 32a - 8a = 24a$

Then we check our first condition of validity to solve for a:

$4a + 12a + 24a = 1$

$\implies 40a = 1$

$\implies a = \frac{1}{40}$

Since a is positive, none of the probabilities are negative. Thus, we know we still have a valid probability distribution.

3b).

$P(X \leq 3) = P(X = 2) + P(X = 3)$

From 3a we can calculate:

$P(X = 2) = (4)(\frac{1}{40}) = \frac{1}{10}$

$P(X = 3) = (12)(\frac{1}{40}) = \frac{3}{10}$

$\implies P(X \leq 3) = \frac{4}{10}$

3c).

$E[X] = (2)(\frac{1}{10}) + (3)(\frac{3}{10}) + (4)(\frac{6}{10}) = \frac{35}{10} = 3.5$

3d).

$SD(X) = \sqrt{Var(X)}$

$Var(X) = E[X^2] - (E[X])^2$

$(E[X])^2 = 3.5^2 = 12.25$

$E[X^2] = (2^2)(\frac{1}{10}) + (3^2)(\frac{3}{10}) + (4^2)(\frac{6}{10})$

$= (4)(\frac{1}{10}) + (9)(\frac{3}{10}) + (16)(\frac{6}{10}) = \frac{127}{10} = 12.7$

$\implies 12.7 - 12.25 = 0.45$

$\implies \sqrt{0.45} = 0.6708$

0.0.5 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

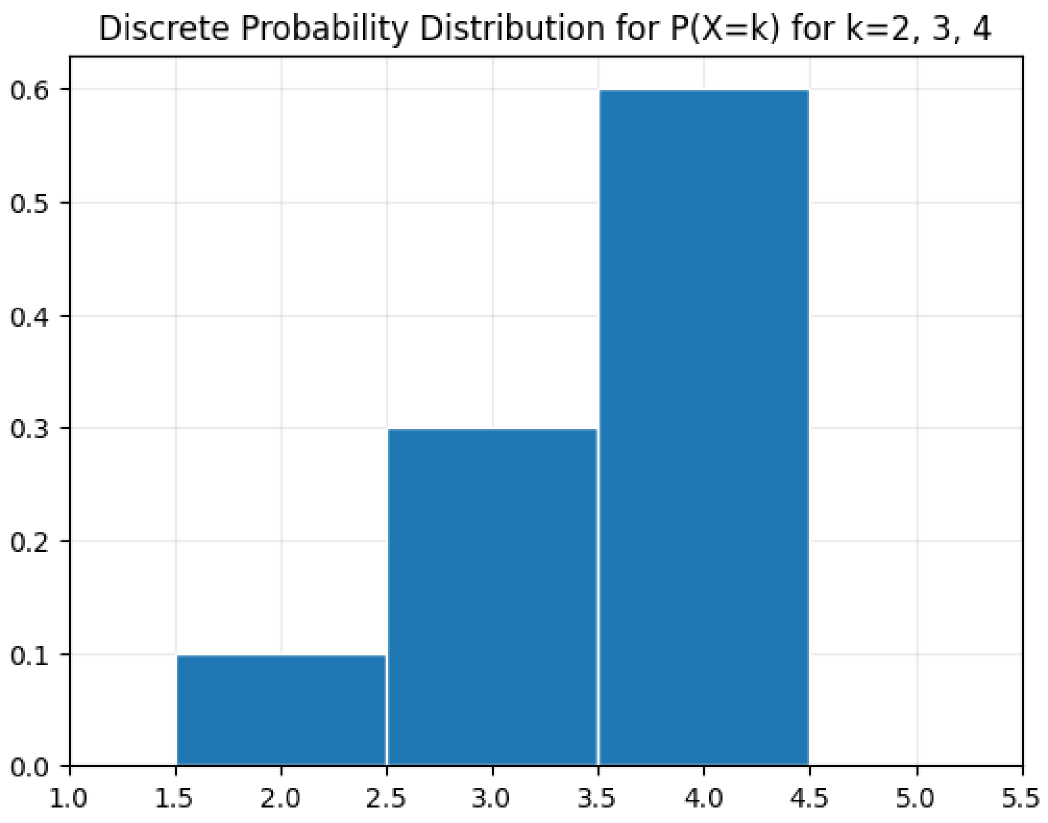
Be sure to include a title on your plot.

```
In [25]: k = np.arange(2,5)
         p = 2*(1/40)*(k*k) - (2*(1/40)*k)

         fig, ax = plt.subplots()

         ax.bar(k, p, width=1, ec='white');
         ax.set_axisbelow(True)
         ax.grid(alpha=0.25)
         plt.xlim(1,5.5)
         plt.title("Discrete Probability Distribution for P(X=k) for k=2, 3, 4");

         # Your code for the histogram above this line
```



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.

4a).

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

$\lambda=4$

$k=6$

$P(X=6)=\frac{e^{-4}*4^6}{6!}=.104$

4b).

$\lambda=5$

$k=6$

$P(X=6)=\frac{e^{-5}*5^6}{6!}=.146$

4c).

$\lambda=1$

$k=6$

$P(X=6)=\frac{e^{-1}*1^6}{6!}=.000511$

4d).

Let A be the event that the machine is working and let B be the event that the machine is frozen. In order to find the probability that teh self check-out tebds exactly 6 customers in the next 10 minutes, we must calculate

$P = P(X = 6|A) * P(A) + P(X = 6|B) * P(B)$

From above, we know that

$P(X = 6|A) = .146$

$P(X = 6|B) = .000511$

$P(A)$ and $P(B)$ were given, thus

$P = (.146)(.9) + (.000511)(.1) = .1317$

0.0.6 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)

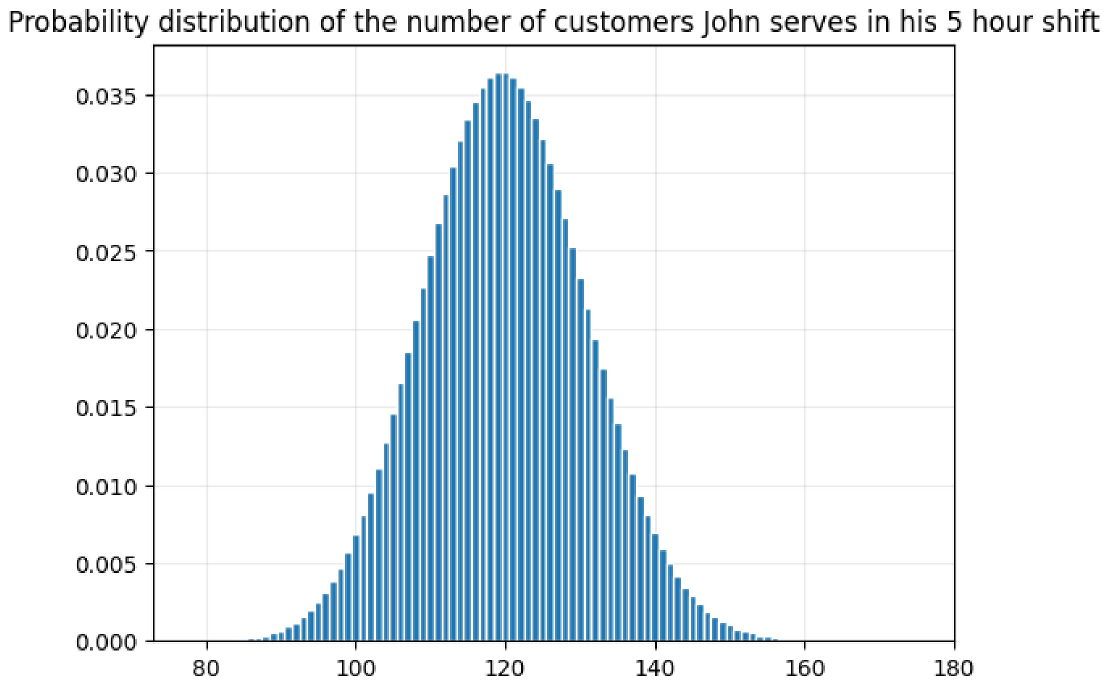
Hint: Since we are changing the time interval over which we are counting customers, you will need to update the parameter μ in the Poisson distribution to be the average number of customers John can serve in a 5-hour shift. You can assume that his rate of 4 customers per 10 minutes scales up consistently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

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

```
In [29]: average_converted = (4/10) * 60* 5
         domain = np.arange(75,161)
         pmf = poisson.pmf(domain, mu=average_converted)

         fig, ax = plt.subplots()

         ax.bar(domain, pmf, width=1, ec='white');
         ax.set_axisbelow(True)
         ax.grid(alpha=0.25)
         plt.xlim(73,180)
         plt.title("Probability distribution of the number of customers John serves in his 5 hour shift
         # your code above this line
```


Probability distribution of the number of customers John serves in his 5 hour shift
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
80
100
120
140
160
180

0.0.7 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 [42]: 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
             '''
             served = 0;
             timer = 0;
             state = [rate_work/10, rate_broken/10]
```

```
        prob_work = p
        prob_froz = 1-p

        while timer < time_len:
            rate = np.random.choice(state, p=[prob_work, prob_froz])
            timer += np.random.exponential(1/rate)
            if timer < time_len:
                served += 1

        return served

        # your code for part i above here
```

```
In [43]: def prob_sim(num_simulations=10000):
             count = 0;

             for ii in range(num_simulations):
                 customers_served = checkout_count(300, 5, 1, .9)
                 if(customers_served >= 100):
                     count += 1
                     # print(customers_served)
                     # print(count)

             return count / num_simulations

         prob_sim(10000)
         # Your code above this line
         # Your code for part ii above this line
         # Output should be approximately 0.70 if code is correct.
```

```
Out[43]: 0.6982
```

```
In [44]: def prob_sim_john(num_simulations=10000):
             count = 0;

             for ii in range(num_simulations):
                 customers_served=checkout_count(300, 4, 4, 1)
                 if(customers_served >= 100):
                     count+=1

             return count / num_simulations

         prob_sim_john(10000)
         # Your code for part iii above this line
         # Output should match your theoretical answer to Part 4f
```

```
Out[44]: 0.9731
```

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.

The better investment for the grocery store seems to be John. The machine only has the probability of about .7 to serve 100 or more customers in a 5-hour shift while john has the probability of about .97. Since .97 is closer to 1 (very high guarentee of always serving 100 or more in 5-hours) than .7, John would be the better investment.