# CSC 521 Final Project
## Jonggoo Kang


### Abstract

I utilize techniques and principles of Monte Carlo theorem that I have learned from this quarter using accident data. I compare the performance of *simulate_once* and *simulate_many* applied on the data I am given.


### Introduction

I choose to investigate on Problem 1 for CSC 521 final project. Since I work at company manipulating the real-time driving data to develop it business model related to an insurance cost, it would be very helpful for me to deeply analyze on *accident.csv* data using theoretical and practical ways of Monte Carlo theorem.

### Data

Accident.csv data I am given has a size of (291,3). I loaded the data by renaming the columns as *plant, days,* and *losses*. By looking at the data with pandas *head()* function and *describe()* function [figure 1], first column named '*plant*' has a binary values of A and B, second column named '*days'* has an integer value ranged from 1 to 1462, and third column named 'losses' has an integer value ranged from 46 to 272,851.

```
By looking at pandas head() function
   plant  days  losses
0      A     1    3348
1      B     4     181
2      B     7     250
3      A    13    5446
4      A    30   38549

By looking at pandas describe() function
              days         losses
count   291.000000     291.000000
mean    697.202749    9189.185567
std     427.485991   21853.188417
min       1.000000      46.000000
25%     327.000000     987.000000
50%     706.000000    2886.000000
75%    1041.500000    8717.000000
max    1462.000000  272851.000000
```

*[figure 1] : The results of pandas head() and describe() of the pandas data frame for accident.csv file*

### Problem 1. Without any simulate from the data, answer the questions:

#### 1-1. The average number of accident per year in plant A and plant B

Before answering the three questions, I assigned new variables for my efficiency when I calculate. Firstly, I created variable called *cnt_plantA* and *cnt_plantB* to check the total count number of plant A and plant B by assigning *df.plant.value_count()[1]* for plant A and *df.plant.value_count()[0]* for plant B. Therefore, there are 135 many of accident in plant A and 156 many of accidents in plant B. The result of the new variables shows as follow:

```
There are 135 accidents in plant A and 156 accidents in plant B
B    156
A    135
Name: plant, dtype: int64
```

*[figure 2] : The total count number of plant A and plant B*

The reason I show the result above is to check the average number of accident per year in plant A and plant B. Since *cnt_plantA* shows that the total number of accidents in plant A during 4 years, I divided *cnt_plantA by* 4 and the result indicates the average number of accident per year in plant A. Also, the average number of accident per year in plant B is calculated in the same way. Therefore, the average number of accidents per year in plant A is 33.74 and 39.0 for plant B, and the result shows as follow:

```
1-1. The average number of accidents per year in plant A is 33.75 and plant B is 39.0
        sol: each of total count number for plant A and plant B are devided by 4 (since 4 years)
```

*[figure 3] :The average number of accidents per year for plant A and plant B*

**1-2. The average loss per accident in plant A and plant B**

I create new variables named *df_plantA* and *df_palntB* to call the data frame where only contains plant A or plant B by coding *df[(df.plant == 'A')]* and *df[(df.plant == 'B')]*. These two variables are used to find the average loss per accident in plant A and plant B. To answer this question, I summed all *losses* values for plant A and plant B by *sum(df_plantA.losses)* and *sum(df_palntB.losses)*, and then I divide them by total count number of each losses for plant A and plant B. Thus,  for average loss per aciident in plant A is assigned in new variable as *avg_lossA = sum(df_plantA.losses) / len(df_planta.losses),* and I create a variable for plant B in the same ways. Therfore, the average loss per accident in plant A is 17470.16 and 2022.96 for plant B, and the result shows as follow:

```
1-2. The average loss per accident in plant A is 17470.155555555557 and plant B is 2022.9615384615386
        sol: each total sum of losses for plant A and plant B are devided by the number of losses
```

*[figure 4] : The average losses per accident for plant A and plant B*

**1-3. The average loss in total per year in plant A and plant B**

Lastly, I created new variable called *avg_lossA_yr* and *avg_lossB_yr* to answer the question of the average loss in total per year in plant A and plant B by summing all losses value for plant A and divide it by 4. Therefore, *avg_lossA_yr = sum(df_plantA.losses) / 4* and I did it same way for plant B. Therefore, the average losses in total per year in plant A is 589617.75 and 78895.5 for plant B,  and the result shows as follow:

```
1-3. The average loss in total per year in plant A is 589617.75 and plant B is 78895.5
        sol: each total sum of losses for plant A and plant B are devided by 4 (since 4 years)
```

*[figure 5] :The average losses in total per year for plant A and plant B*

**Problem2. Now assume the time interval between accidents is exponential and the natural log of a loss due to a single accident is a guassian (aka the loss is lognormal)**

Before answering the questions, I created some other variables that I will use in this problem 2. Firstly, I create *mu*_lossA variable which calculates logged values of average losses for plant A; thus, *mu_lossA = math.log(avg_lossA).* After that, I create another variable which calculates standard deviation values for logged losses for plant A; thus, *sig_loglossA = sd(np.log(df[(df.plant == 'A')].losses)).* Lastly, the variable called *diff_plantA* shows that the day different between two values of column *'days'* using *diff_plantA = df_plantA['days'].diff().* I did the same way to make new variables for plant B of each function.

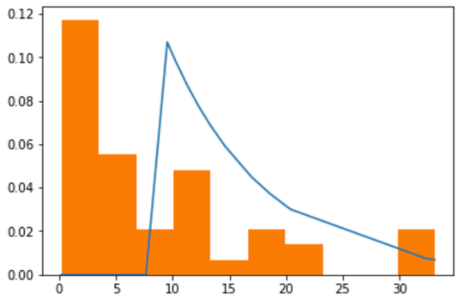
**2-1. Implement a simulate once that simulates one year of losses for both plants.**

I edited the function called *simulate_once.* The edited *simulate_once* function takes a mean value of losses, a sigma value of losses, and a value of the day difference between two values of 'days' and the parameters are named as *mu_loss, sig_loss,* and *mu_diff,* respectively. In the function, two lists of time(day) and loss for one year will be shown. The values of time list are calculated by *random.expovariate(1/a mean value of the day gap)* and loss list are calculated by *random.gauss(a mean value of losses, a sigma value of losses)*. The values in loss list, and then, will be transformed with exponential function; thus, *np.exp(random.gauss(mu_loss, sig_loss).*

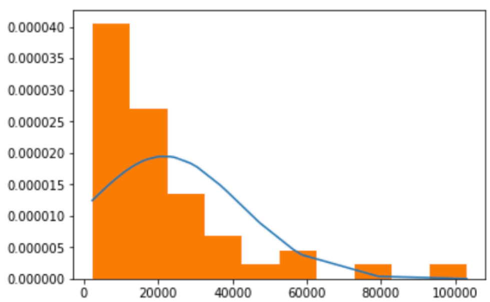
In order to show the outputs, I create a function called *simulate_once_result(mu_loss, sig_loss, mu_diff).* This function shows that the two lists of Time Intervals and Losses. It also shows that the histograms of the two columns and this results are good to check to be satisfied with the assumptions of "the time interval between accidents is exponential and the natural log of a loss due to a single accident is a Gaussian".

Using *simulate_once_result(mu_lossA, sig_loglossA, mu_diffA)*, I can check the two lists values and the two histograms of Time Interval and losses for plant A. Therefore, simulated_once for one year of losses for plant A are calculated as follow:

```
For plant A
Time Interval
[0.18061981949851832, 0.4885673545307967, 0.5510799807691418, 0.7771993013224597, 1.0385355033522918, 1.098879377787231, 1.6371519305426743, 1.8680397271525875, 2.046492708248823, 2.080001784636605, 2.272714286530562, 2.279504319236917, 2.358370177137891, 3.0929608597007037, 3.1817279856430467, 3.411662006281855, 3.436887878036077, 4.0588098552324405, 4.1248889747132464, 4.861644207877225, 4.898631115078417, 5.0567306790113316, 5.768388021986172, 6.041773757367133, 6.660460898173384, 7.649863839463447, 9.529424214214595, 9.752584830567553, 10.20424139855306, 10.275230775566301, 11.221786771722238, 11.353613704715794, 12.222054339280687, 12.434351901029784, 13.325408214546597, 14.649606449868845, 16.896203031208305, 17.08428733047314, 18.577637877499992, 20.109988511576734, 20.4506689639178, 32.297099970923014, 32.505791058913296, 33.12946837266225]
```




```
Losses
[2217.563037873844, 2727.250856242717, 3305.8862466612354, 3528.673586886858, 4616.940284370559, 5424.045599126739, 5501.845611084168, 6206.39734514255, 7371.024891029611, 7669.0173833209365, 7793.998817902215, 8493.858635810253, 10397.646440799175, 10691.24481361832, 10905.246517533395, 11338.00982467167, 11559.99149211613, 11786.026818382215, 12830.269775879016, 13602.566139156443, 14444.147774328683, 14928.632725729725, 15921.128367032323, 16452.38106065178, 16456.679665324486, 17082.52579793651, 17134.408842054327, 19694.388458587917, 20400.530038760426, 20578.813173095434, 23204.097392652548, 23998.514584104338, 24516.40357494226, 28670.75118637599, 30184.027744762254, 30579.222465151845, 36678.476598199966, 38598.889124130685, 39905.94096571755, 47315.174533818084, 56297.11422066549, 58750.768386548654, 79310.68660307619, 103059.88924233917]
```

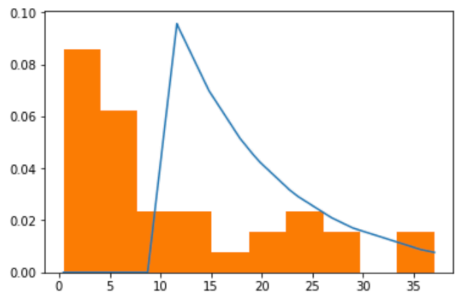



*[figure 6] : The result of simulate_once for one year of losses for plant A*

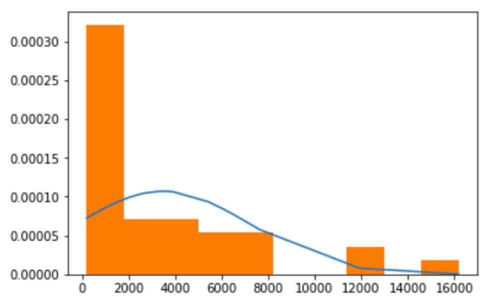
As seen in the result above, the function for plant A shows that lists of time and losses for one year. Also the function outputs the histogram of the values in the lists with plotting the lines. According to the graphs above, the graph for Time Interval shows an exponential distribution after 7 on x-axis. Furthermore, the graph for plant Losses shows a gauss distribution. Therefore, I can be satisfied that the assumptions.

By replacing the parameter for plant B, *simulate_once_result(mu_lossB, sig_loglossB, mu_diffB)*, simulated_once for one year of losses for plant B are calculated as follow:

```
For plant B
Time Interval
[0.4310141903760765, 1.2938174368772029, 1.6649852637797804, 1.715183596957332, 2.5751049586457877, 2.615026469325326
5, 2.6777769553728326, 3.465863014973536, 3.544863406467836, 3.6128922197094813, 3.894049416301447, 4.57391186664930
7, 4.91667366734957, 4.921444982113516, 5.410921280091086, 6.367783064146304, 6.425933615434883, 7.376428116271144,
 7.500526374969628, 8.427495262951773, 8.566425854890824, 8.740302624741332, 11.627759441385775, 11.735000123878445,
 14.788656038288988, 17.832463027131617, 19.12521748592595, 19.842228813379073, 22.735090700434515, 23.11218006183355
6, 23.603867627796525, 26.88385940010949, 29.002668571593055, 35.693896230686256, 37.024774399840176]
```




```
Losses
[203.89352518587788, 214.96066446738243, 563.2502963509767, 650.8501413610192, 684.8135892573059, 842.152569893662, 8
98.0908711410415, 898.1378214631949, 903.9708186319086, 1071.7738439459347, 1109.9745149328512, 1186.7500205921433, 1
201.7415682294918, 1304.6834438078033, 1324.1609814498224, 1359.8761360831288, 1373.8841796529973, 1521.669860083783
8, 2031.0734902697611, 2527.9524814760034, 2676.0267578222547, 3283.12747567638, 3469.774882493909, 3664.00448569463
3, 3788.3999612570365, 3928.4286072299387, 5420.765236354524, 6110.565339792684, 6452.701311653969, 6609.91032636723
1, 7225.5661044049675, 7636.010981010742, 11872.605614868797, 12024.59857996748, 16184.083345736728]
```




*[figure 7] : The result of simulate_once for one year of losses for plant B*

According to the result above, I can conclude that the output occurring from *simulate_once()* function for plant B also satisfies the assumption. The graphs for Time Intervals shows that the exponential distribution after 8 on x-axis and the graph for Losses shows that the gauss distribution in general. The results are created by the function called *prob2_1()*.

**2-2. Running simulate many, what is the average yearly loss with a relative precision of 10%?**

I edit the function called *simulate_many()* to get the list values of Losses with a condition that if the count number of random generator is greater than 30 and absolute value of dmu are smaller than absolute value of relative precision for mu loss; thus, *if k > 30 and abs(dmu) < (abs(mu_loss) * rp)*. Therefore, *simulate_many()* functions for plant A and plant B results in the average yearly loss with a relative precision of 10%. The results above are shown by the function called *prob2_2()* that I create, and it shows as follow:

```
For plant A
[902980.4541347282, 606810.4468825108, 1352281.8597438138, 1425362.210182194, 705341.0781004516, 1516601.329392308, 1
280010.1337657077, 1198028.4095245213, 926849.8777665778, 797096.125319147, 820177.4403974053, 1407332.9984300947, 12
06370.0386100134, 728495.091491887, 1295025.6080161182, 970131.996813691, 739607.3694791058, 1455730.4934734094, 1627
399.2588831752, 905050.1515513079, 1145507.5466748578, 579417.6883521524, 928100.1672288186, 628201.9096055182, 10358
24.2221035135, 467342.86368233204, 1203092.7746718072, 640681.9464485879, 693626.9772420438, 902976.5846293691, 81594
4.7637825228]
For plant B
[104664.56281600575, 154272.57900946782, 106885.54263229067, 74147.4763753453, 162177.23440247166, 84960.54282589653,
 149327.3899912155, 86573.09736280126, 129561.93099272833, 190210.49355486446, 222240.7408326659, 300851.31918914034,
 205954.16716102944, 262285.92347424856, 183162.6419296141, 183961.5091298536, 211823.27649322286, 147106.2393672865
6, 148543.17062901036, 120751.70226280828, 159157.77839783143, 155191.96690887923, 155836.39307187765, 119883.9257471
2512, 97728.59650660564, 133133.94161008942, 126527.43552378372, 122811.1723314722, 231091.93296280137, 335946.446917
1446, 95925.6748696262]
```

*[figure 8] : The average yearly losses with a relative precision of 10% for plant A and plant B*

**2-3. Report the bootstrap errors in your result. How much should the company budget to make sure that it can cover these losses in 90% of the simulated scenarios?**

The function called *prob2_3()* that I create shows the outputs of the values of minimum mean and maximum mean within the confidence interval at 70%, 80, and 90% using *bootstrap()* function. Also, it shows that the result of the budget the company can cover the losses in 90%. Therefore, for plant A, when confidence interval at 70%, the losses would be in the range from approximately 1060263 to 1170141. Moreover, when confidence interval at 80%, the losses would be in the range from approximately 1035745 to 1174562. Finally, when confidence interval at 90%, the losses would be in the range from approximately 999217 to 1189473.

For plant B, when confidence interval at 70%, the losses would be in the range from approximately 151361 to 166402. Moreover, when confidence interval at 80%, the losses would be in the range from approximately 149206 to 166997. Finally, when confidence interval at 90%, the losses would be in the range from approximately 147264 to 167303. The result shows as follow:

```
For plant A
The confidence Interval at 70 is from 1060263.2043 to 1170141.0923
The confidence Interval at 80 is from 1035745.5513 to 1174562.3479
The confidence Interval at 90 is from 999217.9871 to 1189473.987

How much should the company budget to make sure that it can cover these losses in 90% of the simulated secnarios?
Therefore, company can budget to make sure that it can cover the losses in 90%
with a range from minimumly 999217.9871 to maximumly 1189473.987

For plant B
The confidence Interval at 70 is from 151361.3025 to 166402.6975
The confidence Interval at 80 is from 149206.2828 to 166997.0567
The confidence Interval at 90 is from 147264.053 to 167303.1013

How much should the company budget to make sure that it can cover these losses in 90% of the simulated secnarios?
Therefore, company can budget to make sure that it can cover the losses in 90%
with a range from minimumly 147264.053 to maximumly 167303.1013
```

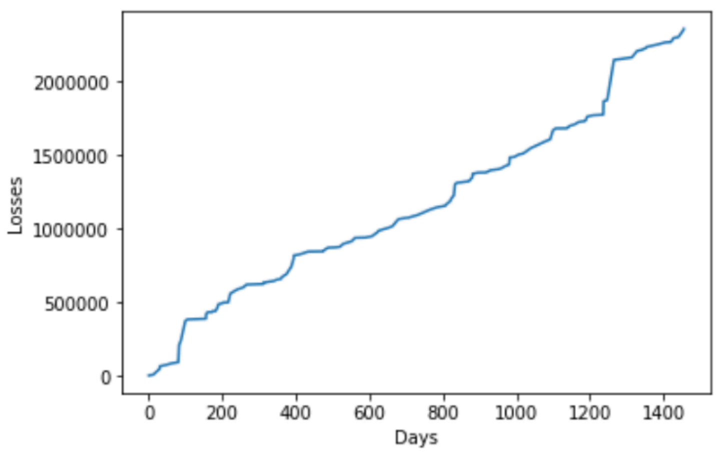
*[figure 9] : The results of bootstrap errors, confidence intervals for 70%, 80%, 90%, and the budget the company can cover.*

Therefore, the company can cover approximately maximumly $118,9473 for plant A and $ 167,303 for plant B.

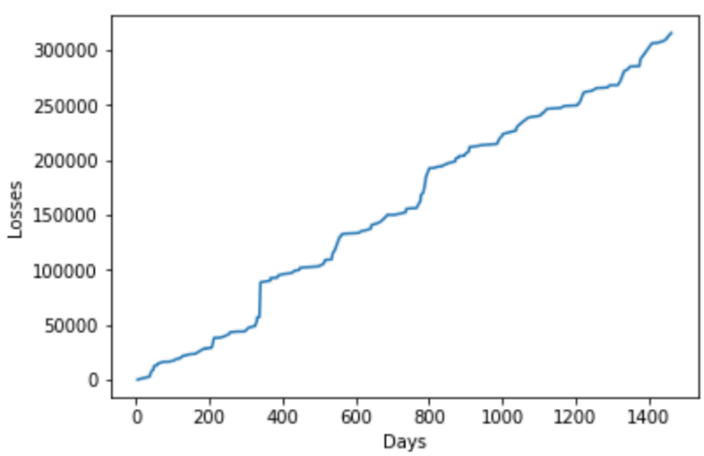
**3. The paper should contain at one graph of losses vs time for one simulated scenario and one histograms of the distribution on losses.**

I create a function to show the graph of losses vs time for one simulated scenario. In the function, there are two for-loops. One loop draws the graph to see the relationship between a *days* columns and cumulated *losses* column for plant A and plant B. According to the result below shows that the relationship between them has a positive correlative relationship.

```
The relationship between days and cumulative losses
Plant A
```



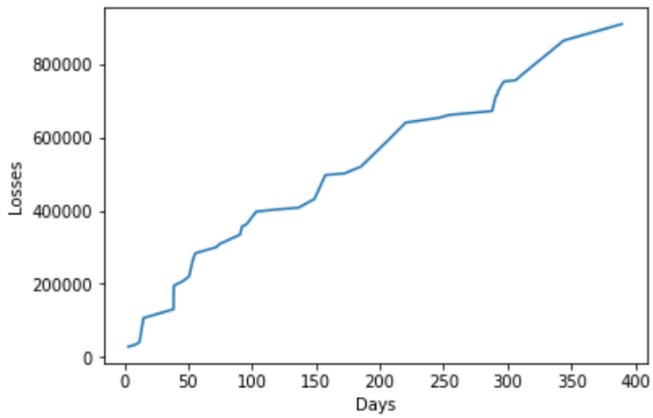

```
Plant B
```



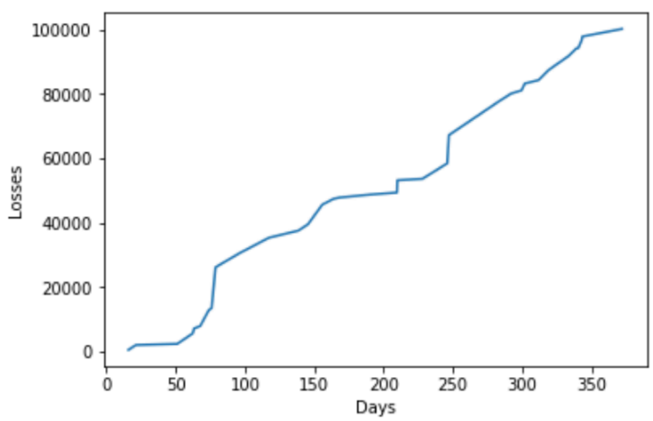

*[figure 10] : The relationship between days and cumulative losses for plant A and plant B*

Another loop draws the graph to see the relationship between cumulative-simulated-days and a cumulative-simulated-log values for plant A and plant B. The results below shows that the relationship between them has a positive-correlative relationship.

The relationship between cumulative-simulated-time and cumulative-simulated-losses
Plant A




Plant B




*[figure 11] : The relationship between cumulative-simulated-days and cumulative-simulative- losses for plant A and plant B*

**Conclusion**

Implementing the function of *simulate_once()* provides a random-expovariate number of a mean of time intervals and a random-gauss number of mean of losses and sigma of losses for plant A and plant B per year. Also, the result satisfies with the assumptions of the time interval between accidents is exponential and the natural log of a loss due to a single accident is a guassian distribution. By implementing the function of *simulate_many()*, I could get the average yearly losses values with a relative precision of 10%. Lastly, using the values from *simulate* functions and *bootstrap()*, I could get the specific amount of losses that the company can cover.

## Appendix

```
import pandas as pd
import numpy as np
import math
import random
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import scipy.stats as stats
from scipy.stats import expon
from termcolor import colored

col = ['plant','days','losses']
df = pd.read_csv('accidents.csv', header = None, names = col)
print(df.shape)
print()
a = colored("By looking at pandas head() function\n", "blue",attrs
= ['bold'])
b = colored("By looking at pandas describe() function\n",
"blue",attrs = ['bold'])
print(a,df.head())
print()
print(b, df.describe())

# 1-1-a. Total number of events for plant A and B
cnt_plantA = df.plant.value_counts()[1]
cnt_plantB = df.plant.value_counts()[0]
# 1-1-a. The average number of accidents per year in plant A and B
avg_plantA_yr = cnt_plantA/4
avg_plantB_yr = cnt_plantB/4

# 1-1-b. The data frame for plant A and plant B
df_plantA = df[(df.plant == 'A')]
df_plantB = df[(df.plant == 'B')]
# 1-1-b. the average loss per accident in plant A and plant B
avg_lossA =  sum(df_plantA.losses) / len(df_plantA.losses)
avg_lossB = sum(df_plantB.losses) / len(df_plantB.losses)

# 1-1-c. the average loss in total per year in plant A and plant B
avg_lossA_yr = sum(df_plantA.losses) / 4
avg_lossB_yr = sum(df_plantB.losses) / 4

color1_1 = colored(("There are 135 accidents in plant A and 156
accidents in plant B"), "blue",attrs = ["bold"])
color1_2 = colored(("\n1-1. The average number of accidents per
year in plant A is"), "blue",attrs = ["bold"])
color_B = colored("and plant B is","blue", attrs = ['bold'])
color1_3 = colored(("1-2. The average loss per accident in plant A
is"), "blue", attrs = ["bold"])
color1_4 = colored(("1-3. The average loss in total per year in
plant A is"),"blue", attrs = ["bold"])

def prob1():
   '''This function show the results for prob1'''
   # 1-1-a. sol: total number of count for plant A and plant B are
devided by 4 (since 4 years)
   # 1-1-b. sol: sum of losses devided by the number of losses
   # 1-1-c. sol: sum of losses devided by 4 (since 4 years)

   print(color1_1)
   print(df.plant.value_counts())
   print(color1_2, avg_plantA_yr, color_B, avg_plantB_yr)
   print("\tsol: each of total count number for plant A and plant B
are devided by 4 (since 4 years)\n")
   print(color1_3, avg_lossA, color_B, avg_lossB)
   print("\tsol: each total sum of losses for plant A and plant B are
devided by the number of losses\n")
   print(color1_4, avg_lossA_yr, color_B, avg_lossB_yr)
   print("\tsol: each total sum of losses for plant A and plant B are
devided by 4 (since 4 years)")
prob1()

def E(f,S): return float(sum(f(x) for x in S))/(len(S) or 1)
def mean(X): return E(lambda x:x, X)
def variance(X): return E(lambda x:x**2, X) - E(lambda x:x,
X)**2
def sd(X): return math.sqrt(variance(X))
def resample(v):
   return [random.choice(v) for k in range(len(v))]
def bootstrap(scenarios, confidence):
   # len(scenarios) == 1000
   samples = []
   for x in range(100):
      samples.append(mean(resample(scenarios)))
   samples.sort()
   # len(samples) == 100
   i = int((100-confidence)/2)
   j = 99-i
   mu_plus = samples[j]
   mu_minus = samples[i]
   return mu_minus, mu_plus

# Day different for plant A and B
diff_plantA = df_plantA['days'].diff()
diff_plantB = df_plantB['days'].diff()
### The result shows that first low has missing value
### this is because the first column has nothing to subtract
### Therefore, I will add 0 to the first row

# Add 0 to the first row for diff_plantA and diff_plantB
diff_plantA[:1] = 0
diff_plantB[:1] = 0

# mu for diff_plantA and diff_plantB
mu_diffA = diff_plantA.mean()
mu_diffB = diff_plantB.mean()
###################################################
#####
# mu_loss for plant A and B
mu_lossA = math.log(avg_lossA)
mu_lossB = math.log(avg_lossB)

# sigma for plant A and B
sig_loglossA = sd(np.log(df[(df.plant == 'A')].losses))
sig_loglossB = sd(np.log(df[(df.plant == 'B')].losses))

def simulate_once(mu_loss, sig_loss, mu_diff):
   """This function results two outputs:
      the time interval between accidents
      and loss for a single accident"""
   # mu_loss : mu of losses for plant A or plant B
   # sig_loss: sigma of losses for plant A or plant B
   # mu_diff : mu of the time gap between days
   time_once_lst = []
   loss_exp_lst = []
   while sum(time_once_lst) <= 365:
      time_once_lst.append(random.expovariate(1/mu_diff))
      loss_exp_lst.append(np.exp(random.gauss(mu_loss,
sig_loss)))

   return time_once_lst, loss_exp_lst


def simulate_once_result(mu_loss, sig_loss, mu_diff):
   '''This function returns "simulate_once" outputs for one year'''
```

```
    # mu_lossA = log of average losses for plant A
    # sigma_A = log of standard deviation losses for plant A
    # num = the length of dataframe of plant A for one year

    time, loss_exp = simulate_once(mu_loss, sig_loss, mu_diff)
    time_lst = sorted(time)
    loss_exp_lst = sorted(loss_exp)

    print(colored(('Time Interval'),"blue", attrs = ["bold"]))
    print(time_lst)
    y = stats.expon.pdf(time_lst, np.mean(time_lst),
np.std(time_lst))
    plt.plot(time_lst, y, label = 'Log Loss')
    plt.hist(time_lst, normed = True)
    plt.show()

    print(colored(('Losses'),"blue", attrs = ["bold"]))
    print(loss_exp_lst)
    y = stats.norm.pdf(loss_exp_lst, np.mean(loss_exp_lst),
np.std(loss_exp_lst))
    plt.plot(loss_exp_lst, y, label = 'Log Loss')
    plt.hist(loss_exp_lst, normed = True)
    plt.show()

color2_1 = colored(("2-1. Implement simulate_once for one year
of losses for both plants.")
                  ,"red",attrs = ["bold"])
color_forA = colored(("For plant A"),"blue", attrs = ["bold"])
color_forB = colored(("For plant B"),"blue",attrs = ["bold"])

def prob2_1():
    print(color2_1)
    print(color_forA)
    print(simulate_once_result(mu_lossA, sig_loglossA, mu_diffA))
    print(color_forB)
    print(simulate_once_result(mu_lossB, sig_loglossB, mu_diffB))

def simulate_many(mu, sigma, muTI, rp = 0.10, ns = 1000):
    loss_many_lst = []
    time_many_lst = []
    s1 = s2 = 0.0
    for k in range(1, ns):
        time, loss = simulate_once(mu, sigma, muTI)
        time_many_lst.append(sum(time))
        loss_many_lst.append(sum(loss))
        s1 += sum(loss)
        s2 += sum(loss) * sum(loss)
        mu_loss = float(s1) / k
        var_loss = float(s2) / k - mu_loss * mu_loss
        dmu = math.sqrt(var_loss / k)
        if k > 30 and abs(dmu) < (abs(mu_loss) * rp):
            break
    return loss_many_lst

# What is the average yearly loss with a relative precision of 10%?
color2_2 = colored(("2-2. Running simulate many,"),"red",attrs =
["bold"])
color2_3 = colored(("What is the average yearly loss with a
relative precision of 10%?\n")
                 ,"red",attrs = ["bold"])


def prob2_2():
    print(color2_2)
    print(color2_3)
    print(color_forA)
    print(simulate_many(mu_lossA, sig_loglossA, mu_diffA))
    print(color_forB)
    print(simulate_many(mu_lossB, sig_loglossB, mu_diffB))
prob2_2()

color2_4 = colored(("2-3. Report the boostrap erros in your
result."),"red",attrs = ["bold"])
color2_5 = colored(("\nHow much should the company budget to
make sure that it can cover these losses in 90% of the simulated
secnarios?"),"red",attrs = ["bold"])

sim_A = simulate_many(mu_lossA, sig_loglossA, mu_diffA)
sim_B = simulate_many(mu_lossB, sig_loglossB, mu_diffB)
sim_lst = [sim_A, sim_B]


def prob2_3():
    print(color2_4)
    for i in sim_lst:
        if i == sim_A: print(color_forA)
        else: print(color_forB)
        for conf in list(range(70,100,10)):
            mu_minus, mu_plus = bootstrap(i, conf)
            print("The confidence Interval at {} is from {} to
{}".format(conf, round(mu_minus,4), round(mu_plus,4)))
            if conf == 90:
                print(color2_5)
                print("Therefore, company can budget to make sure that
it can cover the losses in 90%")
                print("with a range from minimumly {} to maximumly
{}\n".format(round(mu_minus,4),round(mu_plus,4)))
prob2_3()

def graphs():
    days_lst = [df_plantA.days, df_plantB.days]
    plat_lst = [df_plantA.losses, df_plantB.losses]
    print('The relationship between days and cumulative losses')
    for i, j, k in zip(days_lst, plat_lst, range(0,2)):
        if k == 0:
            print("Plant A")
        else:
            print("Plant B")
        plt.plot(np.array(i), np.array(j).cumsum())
        plt.xlabel("Days")
        plt.ylabel("Losses")
        plt.show()

    timeA, logA = simulate_once(mu_lossA, sig_loglossA,
mu_diffA)
    timeB, logB = simulate_once(mu_lossB, sig_loglossB,
mu_diffB)
    time_lst = [timeA, timeB]
    log_lst = [logA, logB]

    print("The relationship between cumulative-simulated-time and
cumulative-simulated-losses")
    for i, j, k in zip(time_lst, log_lst, range(0,2)):
        if k == 0:
            print("Plant A")
        else: print("Plant B")
        plt.plot(np.array(i).cumsum(), np.array(j).cumsum())
        plt.xlabel("Days")
        plt.ylabel("Losses")
        plt.show()
graphs()
```