CSC 521 Final Project

Jonggoo Kang

Problem1

In [1]:
```
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
```

It contains recorded accidents over the last 4 years.

The first column indicate where it happened (plant A or plant B).

The second column indcates the day (0 is 4 years ago).

The third column indicates the loss caused by the accidents in dollars.

In [2]:
```
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())
```

```
(291, 3)

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
```

In [3]:
```
df.describe()
```

Out[3]:

	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

Problem1-1. Without any simulation from the data, answer these questions:

1-1-a. The average number of accidents per year in plant A and B

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

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

In [4]:
```
# 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"])
```

In [5]:
```
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)")
```

therefore,

In [6]:
```
prob1()
```

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

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)

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

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)
```

Problem1-2. 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)

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

2-2. Running simulate many, What is the average yearly loss with a relative precision of 10%? Report the bootstrap errors in your result.

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

In [7]:
```
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
```

In [8]:
```
# 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))

# string variable that I will use the function in the next cell
A = 'A'
B = 'B'
```

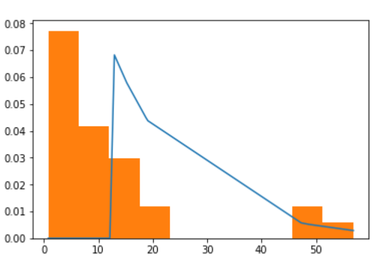
2-1. Implement a simulate once that simulates one year of losses for both plants.

In [9]:
```
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()
```

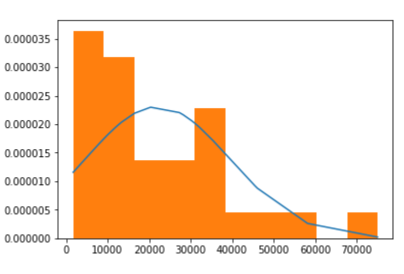
In [10]:
```
color2_1 = colored(("2-1. Implement simulate_once for one year of losses for both plants.")
                   ,"red",attrs = ["bold"])
color_forA = colored(("For plant A"),"blue", attrs = ["bold"])
color_forB = colored(("For plant B"),"blue",attrs = ["bold"])

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))
```

In [11]:
```
prob2_1()
```

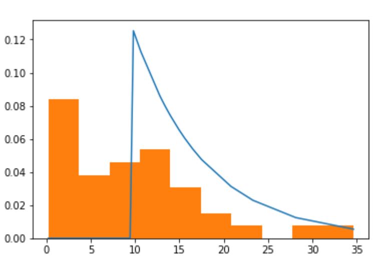
```
2-1. Implement simulate_once for one year of losses for both plants.
For plant A
Time Interval
[0.7834284566620441, 1.0595715339468887, 1.1205241092660052, 2.6909227944504632, 2.7718675557037864, 3.9164488257313868, 3.9442091014642697, 4.260683929624964, 5.173916622598037, 5.258194632685313, 5.309080963162241, 5.729642362532583, 5.881106836679775, 7.313546352118762, 7.6356857366637545, 7.881291057588863, 8.01024652668854, 8.19763702464456, 9.010323229808826, 11.077652682049171, 12.055096517221617, 12.12526651141541, 12.953871170506805, 15.186641346725468, 15.277508346933965, 18.76640445387046, 19.11886959652099, 47.29808838864583, 48.7228936599263, 56.887681875009505]
```



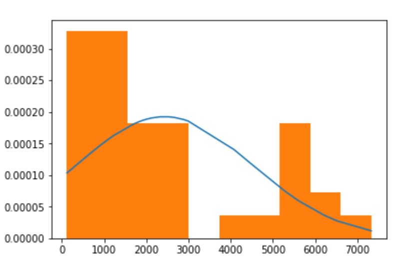
```
Losses
[1724.4551142927123, 5197.725940181831, 5709.447941031977, 5858.264513146545, 7583.42222428901, 7877.3770329268955, 7959.783983014247, 8133.4805469181365, 9270.886428629088, 9678.134532600665, 10029.96020882716, 10044.048125826726, 11719.344534792719, 12553.977914326417, 13402.928409037839, 16486.12168247015, 20308.384392790496, 20408.013801509067, 27280.589995126975, 28209.742827886905, 30336.40488564425, 31417.17069476883, 31832.386567793987, 32202.698601240554, 35363.67273741936, 35699.626950187885, 45538.863671188585, 46103.42664319288, 58166.48598038046, 74998.56562827439]
```



```
None
For plant B
Time Interval
[0.2073497578379524, 0.20751736108568414, 0.24833909719763533, 0.5448463892328022, 1.2283652830753262, 1.8986803951220992, 1.997898948509247, 2.2613378212334094, 2.3632490924301797, 2.658277738224381, 3.2530522619437607, 3.983408986374237, 4.605862751659509, 4.9952201526446185, 5.259317710495939, 6.2286112263027285, 8.188952567422982, 8.517265765661417, 8.876242700802212, 9.016949693017148, 9.439580659580104, 9.80640918946913, 10.650364384477312, 12.774099899800845, 12.889923884402215, 13.20041977381981, 13.322279615685938, 13.388494313472552, 13.677735937380394, 13.998991562132145, 14.95616946161685, 15.598120357454153, 16.50835371886225, 17.51599876955677, 20.820979261668864, 23.294830893029975, 28.12048978380762, 34.636889398412215]
```



```
Losses
[119.35122363084277, 196.4729681539276, 270.98738531370395, 294.4124162104193, 530.9253367600459, 613.7536225294997, 619.7658697983101, 701.9324615628221, 726.4439708071459, 932.6680135281908, 944.3789789293396, 947.1922190189042, 966.3617453405707, 1026.3868769856147, 1051.9397827795522, 1071.6659254057722, 1185.960509262916, 1226.7329065621832, 1707.8740467744426, 1848.2728862149397, 1928.6814308452008, 2009.8146639971258, 2137.9237125440045, 2329.947143573908, 2529.419049002382, 2661.832149866338, 2875.9807035406466, 2991.8522834220676, 4079.23402829987, 4499.271682464508, 5261.594017861785, 5282.170142341417, 5538.571817096471, 5564.192233475261, 5651.11776189287, 6229.369461457273, 6531.238342908159, 7334.031967780787]
```



```
None
```

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

In [12]:
```
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
```

In [13]:
```
# 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))
```

In [14]:
```
prob2_2()
```

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

For plant A
[3721525.7398204645, 1025676.3214487784, 928666.2159123545, 1109950.9287458884, 1183638.215621488, 533611.52279716, 981942.4579938618, 1148941.1820191944, 1491161.2110493677, 640939.5317437744, 736567.8433248427, 1640098.7989561139, 727922.8648073215, 658158.4057101267, 690178.8575305064, 921001.5326630775, 1585486.831382626, 755832.3768965938, 1409991.452840144, 1198874.5495655371, 923403.7816401913, 1040309.271903371, 961174.9490853811, 586695.8252148072, 845722.1292919592, 945597.6006255664, 1391375.0199774678, 923977.5622485747, 893293.401929265, 1213963.8889519826, 1126131.1438770879]
For plant B
[157946.31321776874, 236356.22243638142, 141475.6822680515, 98181.53094967852, 183856.7509501861, 226216.64170457458, 83181.51129365705, 159575.12373180865, 186648.36140394688, 111194.96620567625, 126214.40711250513, 162418.4121168092, 135964.27225400062, 178484.61450054045, 112624.01511851931, 155519.66652118257, 149457.83332123014, 193398.76865566542, 168647.51572531348, 234123.90178912977, 83803.23209600954, 196173.59250665503, 136442.7638386264, 194117.61289856583, 221003.45367297914, 154219.99150759025, 186606.30483773185, 184236.86599478178, 133991.6523030945, 200010.6902279068, 147587.29643957247]
```

2-3. Report the boostrap erros in your result.

How much should the company budget to make sure that it can cover these losses in 90% of the simulated secnarios?

In [15]:
```
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)))
```

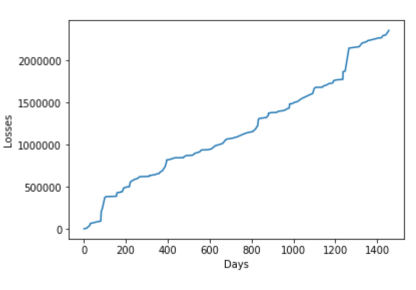
In [16]:
```
prob2_3()
```

```
2-3. Report the boostrap erros in your result.
For plant A
The confidence Interval at 70 is from 1015236.6844 to 1133091.8762
The confidence Interval at 80 is from 994722.082 to 1165153.0
The confidence Interval at 90 is from 963236.359 to 1164626.8115

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 963236.359 to maximumly 1164626.8115

For plant B
The confidence Interval at 70 is from 151467.0932 to 169551.7947
The confidence Interval at 80 is from 147389.2245 to 169070.6512
The confidence Interval at 90 is from 148451.7496 to 171798.7104

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 148451.7496 to maximumly 171798.7104
```

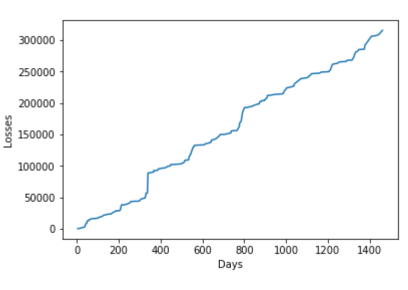
In [17]:
```
def graphs():
    days_lst = [df_plantA.days, df_plantB.days]
    plat_lst = [df_plantA.losses, df_plantB.losses]
    print('The relationship between days and cumulative losses')
    for i, j, k in zip(days_lst, plat_lst, range(0,2)):
        if k == 0:
            print("Plant A")
        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()
```

In [18]:
```
graphs()
```

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




```
Plant B
```




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