Data Wrangling:

In [1]:
```
# .csv from data world bank
# https://data.worldbank.org/indicator/SP.POP.TOTL?end=2020&start=1960
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression, Ridge, MultiTaskElasticNet

# read in the .csv
df = pd.read_csv('world_population.csv')
df.head()
```

Out[1]:

	Country Name	Country Code	Indicator Name	Indicator Code	1960	1961	1962	1963
0	Aruba	ABW	Population, total	SP.POP.TOTL	54208.0	55434.0	56234.0	56699.0
1	Africa Eastern and Southern	AFE	Population, total	SP.POP.TOTL	130836765.0	134159786.0	137614644.0	141202036.0
2	Afghanistan	AFG	Population, total	SP.POP.TOTL	8996967.0	9169406.0	9351442.0	9543200.0
3	Africa Western and Central	AFW	Population, total	SP.POP.TOTL	96396419.0	98407221.0	100506960.0	102691339.0
4	Angola	AGO	Population, total	SP.POP.TOTL	5454938.0	5531451.0	5608499.0	5679409.0

5 rows × 65 columns

In [3]:
```
# create and extract into the year_dataframe
yearList = []
popList = []
year = 1960
while year < 2021:
    yearList.append(year)
    popList.append(df[str(year)].sum())
    #increment year counter
    year += 1

# populate year dataframe
d = {'Year': yearList, 'Population': popList}
year_dataframe = pd.DataFrame(data=d)
#year_dataframe.to_csv('cleaned_data.csv', index=False)
year_dataframe.tail()
```

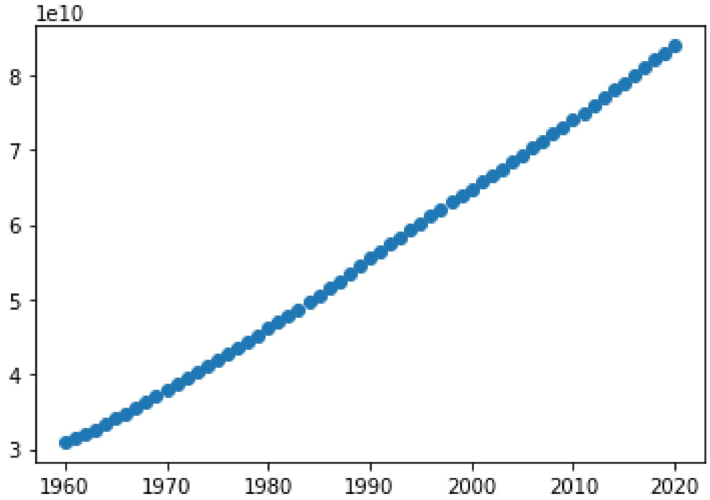
Out[3]:

Year	Population

	Year	Population
56	2016	7.998271e+10
57	2017	8.099699e+10
58	2018	8.199016e+10
59	2019	8.296228e+10
60	2020	8.391063e+10

In [4]:

```
import matplotlib.pyplot as plt
plt.figure()
plt.scatter(year_dataframe["Year"], year_dataframe["Population"])
plt.show()
```



Death rate data:

In [5]:

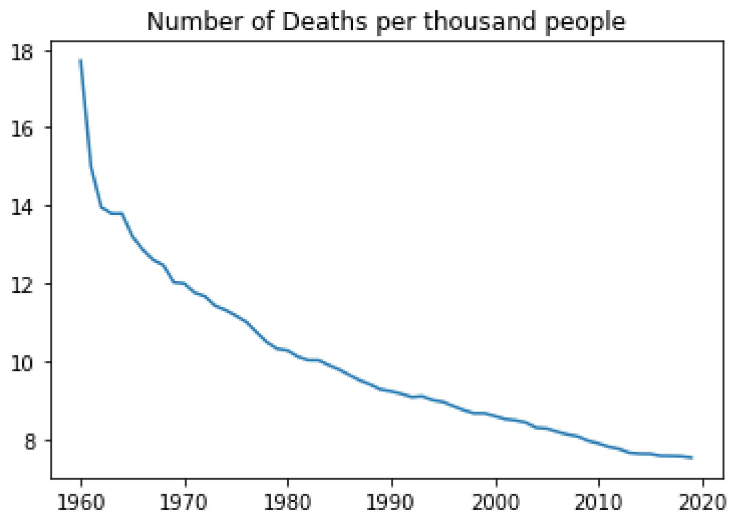
```
# world bank data for death rate over the last 60 years
df_deathrate = pd.read_csv('deathrate_data.csv')
deathRateList = df_deathrate["Death Rate"].to_numpy()
averageDeathRate = sum(deathRateList) / len(deathRateList)
print('average death rate over the last 60 years: ' + str(averageDeathRate))
#show the dataframe
df_deathrate.head()
```

```
average death rate over the last 60 years: 0.009934249999999999
```

Out[5]:

	Year	Number Deaths	Death Rate
0	1960	17.713	0.017713
1	1961	15.001	0.015001
2	1962	13.954	0.013954
3	1963	13.792	0.013792
4	1964	13.792	0.013792

In [6]:
```
import matplotlib.pyplot as plt
plt.figure()
plt.plot(df_deathrate["Year"].to_numpy(), df_deathrate["Number Deaths"].to_numpy())
plt.title('Number of Deaths per thousand people')
plt.show()
```



Birth Rate Data: The birth rate is expected to decline so we will use the birth rate in 2020. We call it average birth rate

In [7]:
```
#2020 birth rate extracted from
averageBirthRate = 0.01788
```

Method #1 Linear Regression:

Model #1 Linear Regression:

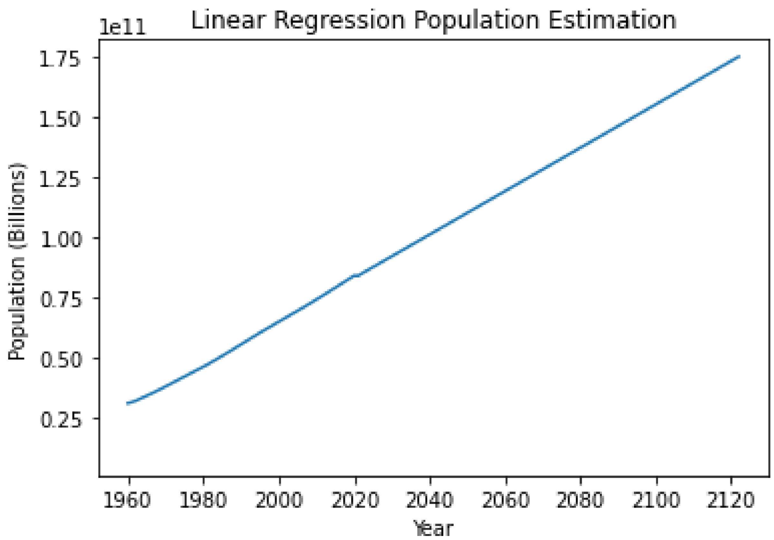
In [8]:
```
x = year_dataframe.iloc[:, 0].values.reshape(-1, 1) # 2d array of years
y = year_dataframe.iloc[:, 1].values.reshape(-1, 1) # 2d array of population
model = LinearRegression().fit(x, y)
y_pred = model.predict([[2122]])
print(y_pred)
y_pred[0][0]
```

```
[[1.74780654e+11]]
```

Out[8]:
```
174780653685.0857
```

In [9]:
```
yearList2 = yearList
popList2 = popList
year = 2021
while year < 2123:
```

```
    y_pred = model.predict([[year]])
    popList2.append(y_pred[0][0])
    yearList2.append(year)
    year = year + 1

plt.figure()
plt.plot(yearList2, popList2)
plt.xlabel('Year')
plt.ylabel('Population (Billions)')
plt.ylim(1e9)
plt.title('Linear Regression Population Estimation')
plt.show()
```



Model #2 Ridge Regression:

Linear least squares with l2 regularization.

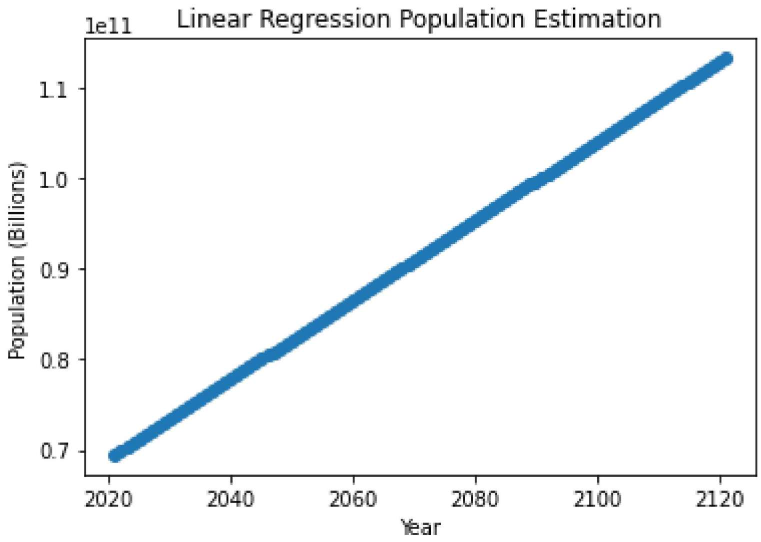
In [10]:

```
x = year_dataframe.iloc[:, 0].values.reshape(-1, 1) # 2d array of years
y = year_dataframe.iloc[:, 1].values.reshape(-1, 1) # 2d array of population
model2 = Ridge(alpha=20000)
model2.fit(x, y)
y2_pred = model2.predict([[2122]])
print(y2_pred[0][0])
```

```
113683804561.7854
```

In [11]:

```
# graph output
yearList2 = [x for x in range(2021,2122)]
popList2 = [model2.predict([[year]]) for year in range(2021,2122)]

plt.figure()
plt.scatter(yearList2, popList2)
plt.xlabel('Year')
plt.ylabel('Population (Billions)')
plt.title('Linear Regression Population Estimation')
plt.show()
```



Model #3 MultiTaskElasticNet:

Multi-task ElasticNet model trained with L1/L2 mixed-norm as regularizer.

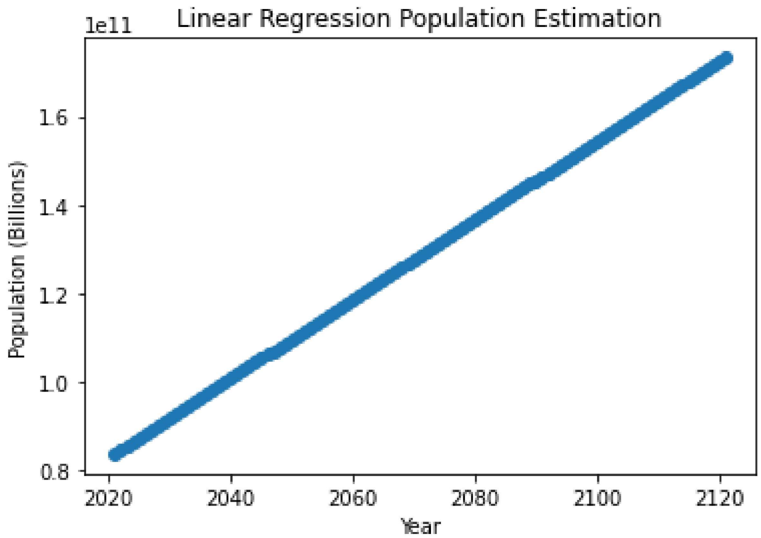
In [12]:

```
x = year_dataframe.iloc[:, 0].values.reshape(-1, 1) # 2d array of years
y = year_dataframe.iloc[:, 1].values.reshape(-1, 1) # 2d array of population
model3 = MultiTaskElasticNet(alpha=3)
model3.fit(x, y)
y3_pred = model3.predict([[2122]])
print(y3_pred[0][0])
```

```
174208275257.6311
```

In [13]:

```
# graph output
yearList2 = [x for x in range(2021,2122)]
popList2 = [model3.predict([[year]]) for year in range(2021,2122)]

plt.figure()
plt.scatter(yearList2, popList2)
plt.xlabel('Year')
plt.ylabel('Population (Billions)')
plt.title('Linear Regression Population Estimation')
plt.show()
```



Method #2 Birth & Death Rates:

We will be using the birdepy library for birth rate and death rate calculations. This package can be easily installed and supports various models for our calculations

In [14]:

```
#pip install birdepy
#pip install gwr_inversion
import birdepy as bd

def getAverageRate(param, model, z0, times):
    result = []
    for i in range(0,5):
        estimationList = bd.simulate.discrete(param, model, 83, times)
        result.append(estimationList[-1])

    return sum(result) / len(result)
```

Model#4 Linear Calculation:

Uses paramaters *y* and *v* (birth and death rate respectively) to estimate the population in 2122. The birth and death rate are held linear at the same rate

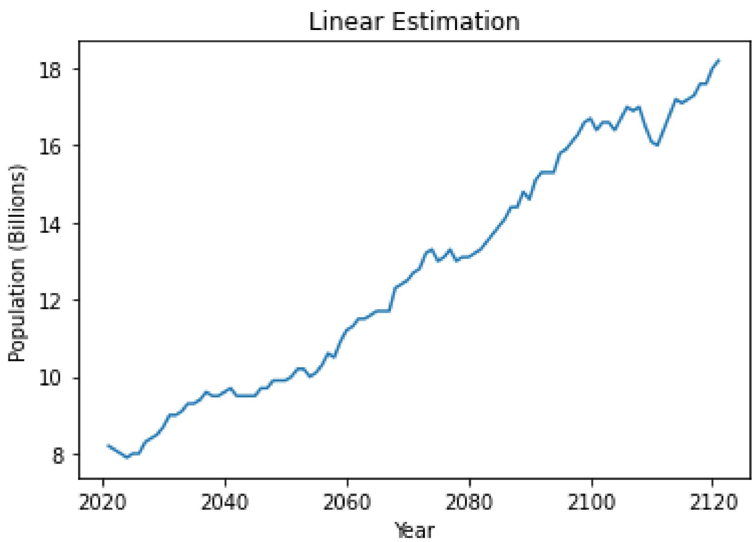
In [18]:

```
#pip install birdepy
#pip install gwr_inversion
import birdepy as bd

y = averageBirthRate # birth rate (upislon)
v = averageDeathRate # death rate
z0 = 83 # starting population for the earth in billions (year 2020 8.3 billion)
param = [y,v]
model = "linear"
# discrete / continous times
t_max = 3
times = [x for x in range(1,102)]
```

```
populationOverYears = bd.simulate.discrete(param, model, z0, times)

averageRate = getAverageRate(param, model, z0, times)
#get an average rate from 10 trials bd.simulate.discrete() birdepy function
#print('Average Final Population Estimate: ' + str(averageRate / 10) + ' billion')
```

In [19]:

```
plt.figure()
plt.plot([x for x in range(2021,2122)], [x /10 for x in populationOverYears])
plt.xlabel('Year')
plt.ylabel('Population (Billions)')
plt.title('Linear Estimation')
plt.show()
print('Final population estimation: ' + str(populationOverYears[-1] / 10) + ' billion')
```




```
Final population estimation: 18.2 billion
```

In [20]:

```
output = [x / 10 for x in populationOverYears]
print(output) # in billions
```

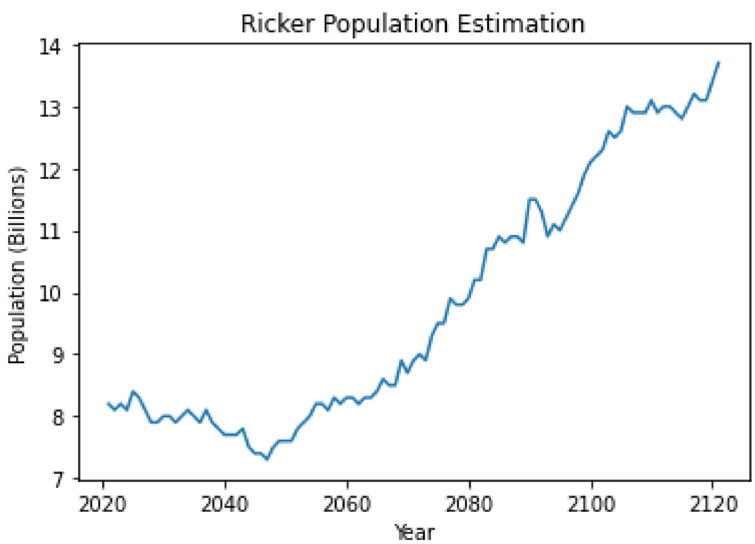
```
[8.2, 8.1, 8.0, 7.9, 8.0, 8.0, 8.3, 8.4, 8.5, 8.7, 9.0, 9.0, 9.1, 9.3, 9.3, 9.4, 9.6, 9.
5, 9.5, 9.6, 9.7, 9.5, 9.5, 9.5, 9.5, 9.7, 9.7, 9.9, 9.9, 9.9, 10.0, 10.2, 10.2, 10.0, 1
0.1, 10.3, 10.6, 10.5, 10.9, 11.2, 11.3, 11.5, 11.5, 11.6, 11.7, 11.7, 11.7, 12.3, 12.4,
12.5, 12.7, 12.8, 13.2, 13.3, 13.0, 13.1, 13.3, 13.0, 13.1, 13.1, 13.2, 13.3, 13.5, 13.
7, 13.9, 14.1, 14.4, 14.4, 14.8, 14.6, 15.1, 15.3, 15.3, 15.3, 15.8, 15.9, 16.1, 16.3, 1
6.6, 16.7, 16.4, 16.6, 16.6, 16.4, 16.7, 17.0, 16.9, 17.0, 16.5, 16.1, 16.0, 16.4, 16.8,
17.2, 17.1, 17.2, 17.3, 17.6, 17.6, 18.0, 18.2]
```

Model #5 Using Ricker calculation:

The Ricker model uses exponentially decreasing birth rate and increasing death rate from the previous year to calculate next year's state. The Richter model is a "classic discrete population model which gives the expected number N t+1 (or density) of individuals in generation t + 1 as a function of the number of individuals in the previous generation." https://en.wikipedia.org/wiki/Ricker_model

In [21]:
```
#pip install birdepy
#pip install gwr_inversion
import birdepy as bd

y = averageBirthRate # birth rate (upislon)
v = averageDeathRate # death rate
z0 = 83 # starting population for the earth in billions (year 2020 8.3 billion)
param = [y,v,0,1]
model = "Ricker"

# discrete / continous times
t_max = 3
times = [x for x in range(1,102)]

populationOverYears = bd.simulate.discrete(param, model, z0, times)

averageRate = getAverageRate(param, model, z0, times)
#get an average rate from 10 trials bd.simulate.discrete() birdepy function
#print('Final Population Estimate after 10 trials: ' + str(averageRate / 10) + ' billio
```

In [139…
```
# from an older notebook run, like this graph
plt.figure()
plt.plot([x for x in range(2021,2122)], [x /10 for x in populationOverYears])
plt.xlabel('Year')
plt.ylabel('Population (Billions)')
plt.title('Ricker Population Estimation')
plt.show()
print('Final population estimation: ' + str(populationOverYears[-1] / 10) + ' billion')
```




```
Final population estimation: 13.7 billion
```

In [137…
```
output = [x / 10 for x in populationOverYears]
print(output) # in billions
```

```
[8.2, 8.7, 8.6, 8.7, 9.0, 9.3, 9.3, 9.4, 9.4, 9.2, 9.2, 9.1, 9.3, 9.4, 9.5, 9.8, 10.1,
9.9, 10.0, 10.4, 10.5, 10.4, 10.3, 10.3, 10.5, 10.5, 10.6, 10.5, 10.6, 10.7, 10.8, 11.0,
10.9, 11.1, 11.2, 11.2, 11.4, 11.5, 11.6, 11.4, 11.3, 11.4, 11.5, 11.6, 11.5, 11.4, 11.
```

```
5, 11.6, 11.6, 11.5, 11.9, 11.8, 11.7, 11.8, 11.8, 11.3, 11.1, 10.9, 10.9, 10.9, 10.9, 1
0.8, 11.0, 10.9, 10.7, 10.7, 10.6, 10.5, 10.6, 10.8, 10.7, 10.9, 10.9, 11.3, 11.2, 11.3,
11.1, 11.3, 11.3, 11.5, 11.8, 11.9, 12.1, 12.1, 12.2, 12.0, 12.1, 12.1, 12.1, 12.1, 12.
3, 12.3, 12.3, 12.3, 12.3, 12.2, 12.2, 12.8, 12.9, 13.0, 12.8]
```

Conclusion

As obtained using the Ricker model, the population of the earth in 2122 will be 12.8 billion people. The Ricker model produced our most reasonable number estimation while utilizing the most logical math. The Ricker model predicts based on birth and death rates and utilizes an exponentially decreasing birth rate. The exponentially decreasing birth rate seems logical after examining the decreasing human birth rate over the last 60 years. Overall, the Ricker model produces a logical final population estimation while utilizing a logical mathematical method to arrive at the conclusion.

Please note the values vary slightly in the notebook from the final conclusion due to being run again

In [ ]:

```
```