US Population Growth (a) Import the data and create two new columns. Create one column that is the number of years since 1790. Create another column that is the population in millions.

In [7]:

```
# import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.metrics import r2_score
from sklearn.linear_model import LinearRegression

# read in csv data
import pandas as pd
data = pd.read_csv('us_pop_data.csv')

# add columns
data['years_since_1790'] = data['year'] - 1790
data['pop_in_millions'] = data['us_pop'] / 1e06

print(data)
```

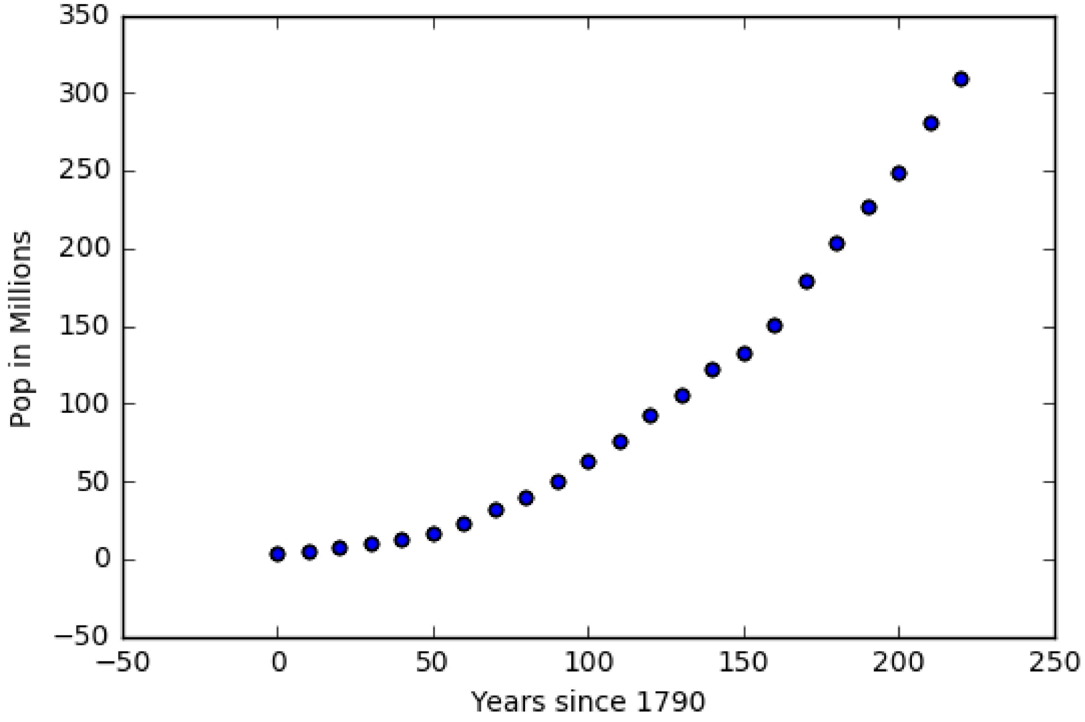
```
    year     us_pop  years_since_1790  pop_in_millions
0   1790    3929326                 0         3.929326
1   1800    5308483                10         5.308483
2   1810    7239881                20         7.239881
3   1820    9638453                30         9.638453
4   1830   12866020                40        12.866020
5   1840   17069453                50        17.069453
6   1850   23191876                60        23.191876
7   1860   31443321                70        31.443321
8   1870   39818449                80        39.818449
9   1880   50189209                90        50.189209
10  1890   62947714               100        62.947714
11  1900   76212168               110        76.212168
12  1910   92228496               120        92.228496
13  1920  106021537               130       106.021537
14  1930  122775046               140       122.775046
15  1940  132164569               150       132.164569
16  1950  150697361               160       150.697361
17  1960  179323175               170       179.323175
18  1970  203302031               180       203.302031
19  1980  226545805               190       226.545805
20  1990  248709873               200       248.709873
21  2000  281421906               210       281.421906
22  2010  308745538               220       308.745538
```

US Population Growth (b) Plot the US population (in millions) versus the years since 1790.

In [13]:

```
# identify x and y axis
x = data['years_since_1790'].values[:,np.newaxis]
y = data['pop_in_millions'].values

# create scatter plot
plot = plt.figure(1)
plt.scatter(data['years_since_1790'], data['pop_in_millions'])
plt.xlabel("Years since 1790")
plt.ylabel("Pop in Millions")
plot
```

Out[13]:



US Population Growth (c) Create a linear regression model to predict the US population (in millions) t years from 1790. Find and report the R2-value of this model.

In [14]:

```
# create linear regression model

model = LinearRegression()
model.fit(x,y)
y_pred = model.predict(x)
r2_score(y, y_pred)
```

Out[14]:

```
0.91924374470804415
```

US Population Growth (d) Create another new column in your data by squaring the number of years since 1790.

In [19]:

```
# create new column and square the years

data['years_squared'] = data['years_since_1790']**2
print(data)
```

```
    year     us_pop  years_since_1790  pop_in_millions  years_squared
0   1790    3929326                 0         3.929326              0
1   1800    5308483                10         5.308483            100
2   1810    7239881                20         7.239881            400
3   1820    9638453                30         9.638453            900
4   1830   12866020                40        12.866020           1600
5   1840   17069453                50        17.069453           2500
6   1850   23191876                60        23.191876           3600
7   1860   31443321                70        31.443321           4900
8   1870   39818449                80        39.818449           6400
9   1880   50189209                90        50.189209           8100
10  1890   62947714               100        62.947714          10000
11  1900   76212168               110        76.212168          12100
12  1910   92228496               120        92.228496          14400
13  1920  106021537               130       106.021537          16900
14  1930  122775046               140       122.775046          19600
15  1940  132164569               150       132.164569          22500
16  1950  150697361               160       150.697361          25600
17  1960  179323175               170       179.323175          28900
18  1970  203302031               180       203.302031          32400
19  1980  226545805               190       226.545805          36100
20  1990  248709873               200       248.709873          40000
21  2000  281421906               210       281.421906          44100
22  2010  308745538               220       308.745538          48400
```

US Population Growth (e) Run another linear regression, where your input feature is the square of the number of years since 1790. Find and report the R2-value of this model.

In [21]:

```
# square the number of years since 1790

x2 = data['years_squared'].values[:,np.newaxis]
model.fit(x2,y)
y2_pred = model.predict(x2)
r2_score(y,y2_pred)
```

Out[21]:

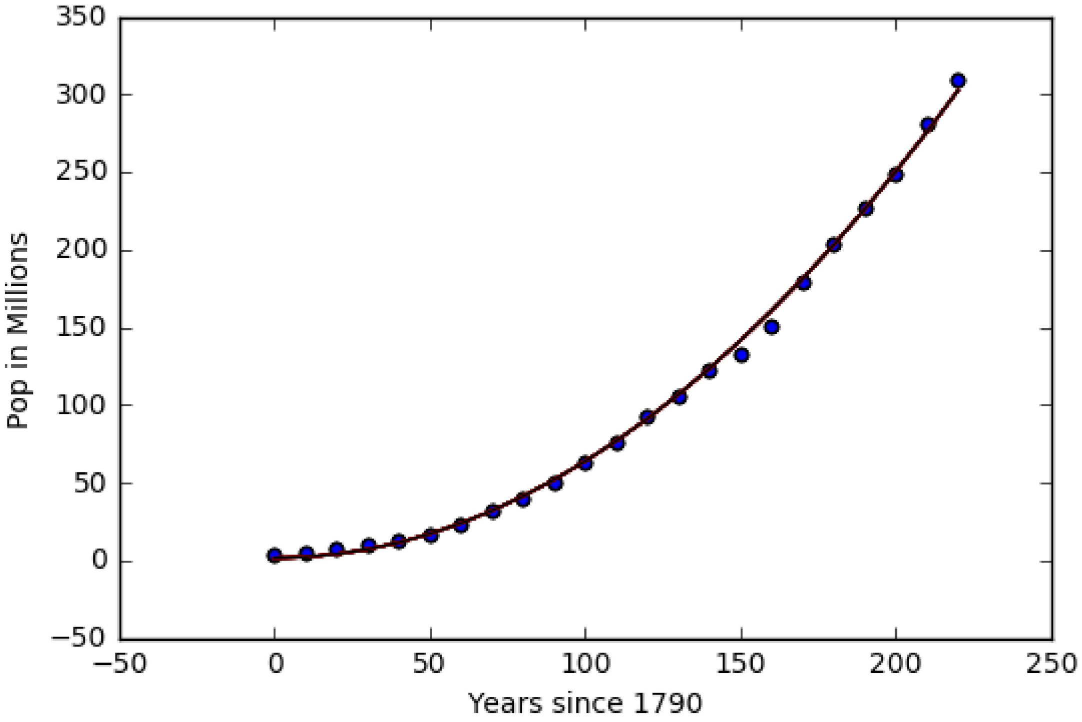
```
0.99849156949866458
```

US Population Growth (f) Plot the models you built on top of the data. Which one fits the data better? Is this apparent in your R2-values. Explain.

In [27]:

```
plt.plot(x,y_pred, c = 'r')
plt.plot(x, y2_pred, c = 'k')
plot

# the above squared model from question "e" is 99.8% where as 91.9% from question "c"
```

Out[27]:

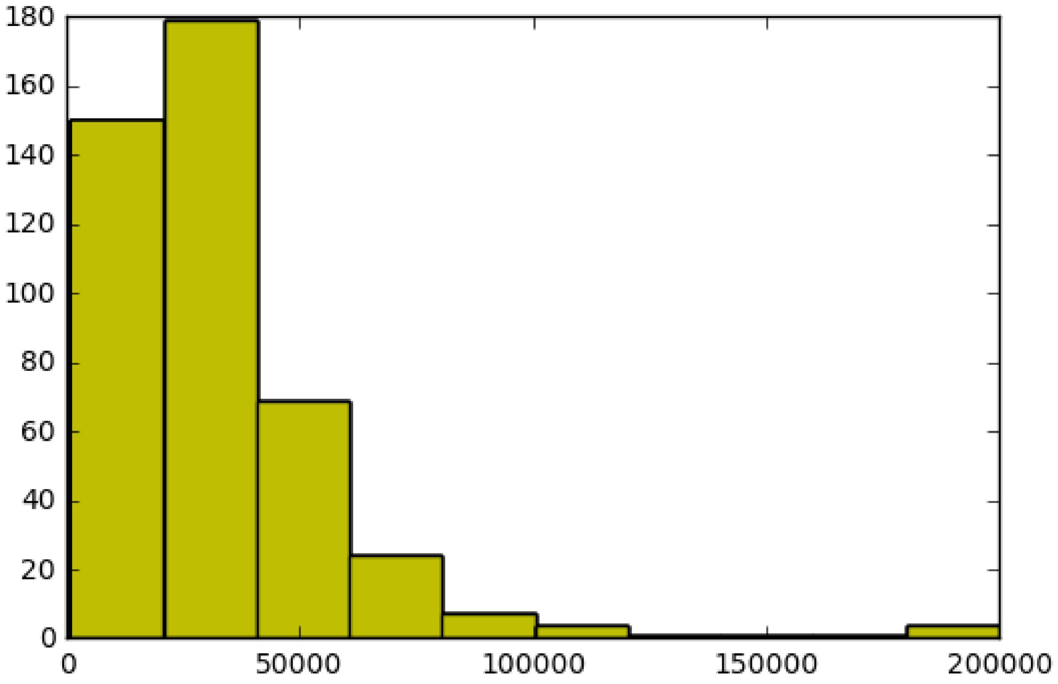


Customer Spending Data (a) Make a histogram of the customer spending amounts.

In [35]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import colors

# import packages and csv data
df_spend = pd.read_csv('customer_spending.csv')
plot_2 = plt.figure(2)
plt.hist(df_spend['ann_spending'])
plot_2
```

Out[35]:



Customer Spending Data (b) Make a new data set that is a log transformation of the customer spending amounts.

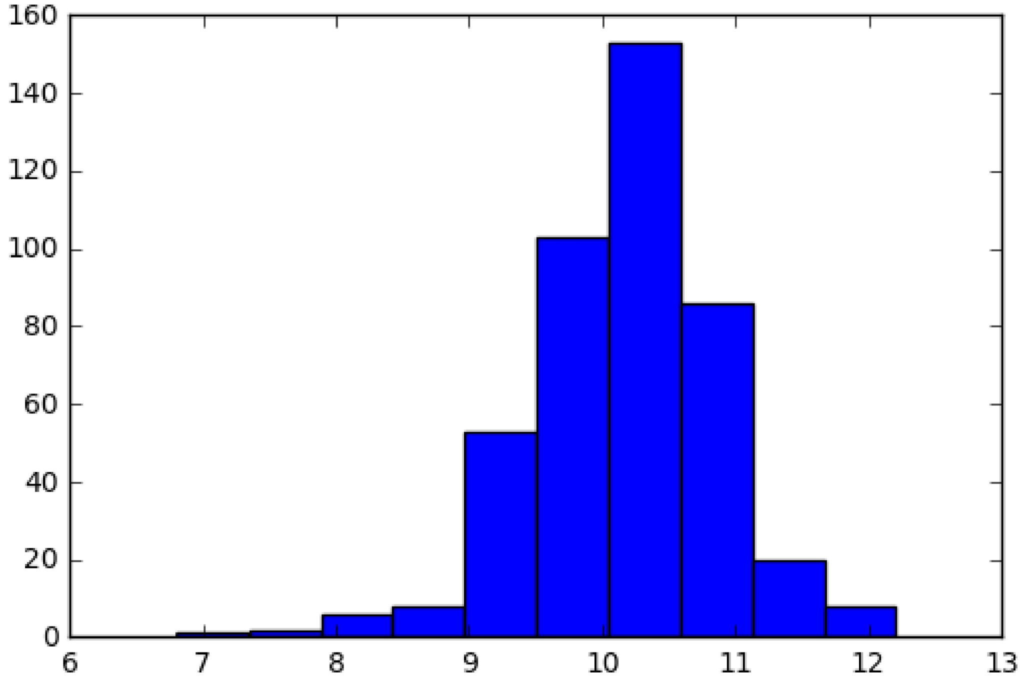
```
In [38]: # create new dataset of spending amounts

         customer_spending = np.array(df_spend['ann_spending'])
         data = np.log(customer_spending)
         data
```

```
Out[38]: array([ 10.43740451, 10.41229113, 10.50807671, 10.21760462,
                 10.73856823, 10.19279331, 10.18357838, 10.29272165,
                  9.8359579 , 10.75068541, 10.43010778,  9.96453561,
                 11.04912655, 10.87447478, 10.89170817,  9.73867187,
                 10.22842928,  9.93561587, 10.6731327 , 10.06164448,
                 10.37710963,  9.45524542, 10.8927499 , 11.61574375,
                 10.99188114, 10.38130433,  9.79512253,  9.86407082,
                 11.09014124, 10.83814851, 10.59104413,  9.5451684 ,
                 10.21921029, 10.8080093 ,  8.86021536, 10.03355044,
                 10.75709404, 10.76513201, 10.70214274, 11.16685128,
                 10.81241121, 10.48029753, 10.60903256, 10.73880681,
                 10.38566676, 11.04825161, 10.80037118, 12.1689625 ,
                 10.48346603, 11.18225238,  9.67878028,  9.70941725,
                 10.91345076, 10.27763398, 10.45443733,  9.80714184,
                 11.32118368, 10.45533035,  9.83392309, 10.10041032,
                 10.08184292, 12.01902594, 10.30239789, 10.79220152,
                  9.63906617, 11.43563397,  9.44383022, 10.76563884,
                 10.06271106,  9.54995054, 10.43611381, 11.07623281,
                 10.02543966, 10.68024032, 10.28510386, 10.20073622,
                  9.89288151, 11.11771757,  9.7154698 ,  9.41548287,
                  9.42641903, 10.38649995, 10.52457466, 10.23501903,
                 10.13233481, 12.2055275 , 11.92169843, 11.27275019,
                 10.01654834, 10.62485769,  9.76508718, 10.2570627 ,
                 11.47181176, 10.90124767, 10.453255  ,  9.26359674,
                  9.6153387 ,  7.67693715,  7.81439963,  9.89978069,
                 10.73275963, 10.56343983, 10.14987877, 11.41308303,
                 10.04550781, 10.04316253, 10.21423897, 10.76229747,
                  9.94736089, 10.79205759,  9.86537023, 10.82063791,
                 10.5003716 , 10.07878449, 10.12914793,  9.8263905 ,
                  9.8573389 ,  9.87179047, 10.34138758,  9.77258128,
                 10.14639451,  9.36443391,  9.65078645, 10.46498728,
                 10.76976838, 11.56215363, 10.36545901, 10.72344358,
                  9.54802616, 10.79333163,  9.79244425,  8.22416351,
                  9.96519394,  9.45712232,  9.45312968,  9.38219063,
                 10.26806122,  9.94592424, 10.18501321,  9.68657455,
                 10.46490172, 10.83253655, 10.92339929, 10.00960278,
                 10.1396263 , 11.31785904,  9.50308461,  9.80791234,
                  9.37746369, 10.52538038,  9.9944702 ,  9.12685006,
                 10.1992494 ,  9.78633557,  6.80682936, 10.73861161,
                 10.43907408, 10.17427776, 10.15288347, 10.35834582,
                 10.15046481,  9.71625395, 10.10716282, 11.07699084,
                 10.23505492, 11.03550238, 10.3194309 ,  9.40648265,
                  9.13216259,  9.35045008, 10.33588666, 11.02763762,
                  9.73210594, 10.58367613,  9.29145942, 10.24020976,
                 11.13113782, 10.27591281, 10.11411308,  9.76766783,
                 10.36901242, 12.15566843, 10.2128455 , 12.1317962 ,
                  8.75998249,  9.81153672,  9.1622    , 10.30861927,
                 10.52854324, 10.3119814 , 10.39454915,  9.69996279,
                  9.24609354, 10.34817337,  9.50271143, 10.52848972,
                 11.08337256, 10.19705229, 10.14564911,  9.6376322 ,
                 10.89794231, 11.0517785 , 10.97478003,  8.50207955,
                  9.28628238, 10.76519537,  9.16471519,  9.98732306,
                  9.75086071, 10.80216406, 10.16585182, 11.58203166,
                  9.52303208, 10.05556466, 10.23774305, 10.56356904,
                 11.15307405, 10.14191373, 10.46826144,  9.04002643,
                  9.88359092, 10.24145867,  9.74911185,  9.81694879,
                  9.30063801, 10.01175883, 10.64053199,  9.30991418,
        9.09750756,  10.0177979 ,  10.43673003,   9.92059046,
       10.41439317,   9.27021177,  10.28489901,   9.53032021,
        9.64562315,  10.27505111,   9.19248185,  11.03751525,
       10.75877484,  10.40061985,  10.19298048,  10.10830419,
       10.3446415 ,  10.46535794,   9.67451454,   9.751734  ,
       10.12858919,   9.97366639,   9.17543832,  11.44372543,
       10.03469124,  10.94445307,  10.65953911,  10.53385466,
       10.02424394,   9.99300828,  11.20153796,  11.1604844 ,
        9.79199718,   9.75359446,  10.25315802,   9.77457405,
       10.41939027,  10.96398252,  10.64977258,  10.40520179,
       10.77789323,   9.88669738,   9.43188264,   9.34749021,
        9.76238471,  10.78292756,   8.86700464,   8.25582843,
       10.74071343,  10.24891953,  10.08917843,  10.59383081,
        9.29917509,  10.34435198,  11.0927781 ,  10.67807646,
       11.49088433,  10.73617926,   9.51863357,  10.31105046,
        9.88842467,  10.71408438,   9.25903529,   9.68296556,
        9.67884288,  10.46774967,  10.41661054,   9.77554054,
       10.26419934,  10.08468348,  10.23250349,   8.34188697,
       10.44993125,  10.81438376,  10.13173804,  10.33318989,
       10.76223391,  10.20184989,  10.85603307,  10.12535028,
        9.42084438,  10.69776883,  10.12326545,  10.68702373,
       10.6823995 ,   9.80040222,  10.070357  ,  10.50358699,
        9.58355773,   9.5338721 ,   9.94984619,  11.12189566,
        9.14558849,   9.91709456,  10.13899436,  10.27725557,
       10.48091519,  11.78201323,   9.40705781,   8.32579053,
       10.37735869,   9.77223927,   9.97487755,  11.08037175,
       10.3726158 ,  11.69766909,  10.48768445,  10.86482813,
        9.81929039,  10.03403324,  10.06573394,   9.96345313,
       10.0478476 ,  10.21859022,   9.80961634,  10.85767119,
        9.08341568,   9.72214552,  10.48175685,  10.92101631,
        9.3080114 ,  10.92972626,   9.16753725,  10.84083503,
        8.38068595,  10.5348923 ,  10.43529158,   8.15622332,
       10.19436452,  10.32803608,  10.36825877,   9.22335531,
       10.2469354 ,   8.98406693,   8.93931874,   9.38446173,
        8.86742744,  10.2429553 ,   9.63456181,   8.84735988,
       10.26262945,   9.06612352,  10.87237073,  10.42231111,
        9.64393932,  10.47061789,   9.45336498,   9.16910162,
       10.52096785,  10.75720051,   9.19877374,   9.93909598,
       10.49399274,  10.45400493,  10.95957501,   9.34128091,
       10.95042051,   9.79890451,   9.30091207,  10.22408444,
        9.55307865,   9.69689379,   9.44730793,   9.26388114,
        9.18327745,  10.69258107,   9.82254861,   9.69480149,
       10.43664202,  10.01117536,   9.83985529,   9.51554306,
        9.3354741 ,  10.76454038,  10.51740211,  10.65166728,
       10.37608157,   9.40401392,  10.75705145,  10.82335189,
       10.05466238,  10.24177956,   9.82189782,   9.60447486,
        9.86433084,  10.53709702,   9.82444435,  10.2936698 ,
       10.45754506,  10.18606914,  10.47517306,   9.83097062,
       10.34769249,  10.52387585,  10.46153068,  10.29004164,
       10.50320279,  10.42186459,  10.75600747,  11.15356119,
        9.75799925,   9.71528876,  10.19264354,  10.47897669,
       10.27591281,   9.21423279,  10.3903483 ,  11.20234317,
       10.78667641,  11.25683249,   9.78886203,   8.93445511])
```

Customer Spending Data (c) Make a histogram of the log transformed dataset.

In [40]:

```
# plot the new data
plot_3 = plt.figure(3)
plt.hist(data)
plt.show(plot_3)
```



Customer Spending Data (d) Compare the two histograms. Discuss why it might be useful to apply a log transformation to this data for modeling purposes.

In [41]:

```
# the logged data removes the outliers. the first histogram is extremely skewed
# where as the second is far more symmetrical and easier to interpret.
```