In [1]:
```
import numpy as np
import pandas as pd
```

Reading the file into a dataframe:

In [2]:
```
pop_df = pd.read_csv("http://facweb.cs.depaul.edu/mobasher/classes/csc478/da
pop_df
```

Out[2]:

	year	hare	lynx	carrot
0	1900	30000.0	4000.0	48300
1	1901	47200.0	6100.0	48200
2	1902	70200.0	9800.0	41500
3	1903	77400.0	35200.0	38200
4	1904	36300.0	59400.0	40600
5	1905	20600.0	41700.0	39800
6	1906	18100.0	19000.0	38600
7	1907	21400.0	13000.0	42300
8	1908	22000.0	8300.0	44500
9	1909	25400.0	9100.0	42100
10	1910	27100.0	7400.0	46000
11	1911	40300.0	8000.0	46800
12	1912	57000.0	12300.0	43800
13	1913	76600.0	19500.0	40900
14	1914	52300.0	45700.0	39400
15	1915	19500.0	51100.0	39000
16	1916	11200.0	29700.0	36700
17	1917	7600.0	15800.0	41800
18	1918	14600.0	9700.0	43300
19	1919	16200.0	10100.0	41300
20	1920	24700.0	8600.0	47300

In [3]:
```
pop_df.head(5)
```

Out[3]:

	year	hare	lynx	carrot
0	1900	30000.0	4000.0	48300
1	1901	47200.0	6100.0	48200
2	1902	70200.0	9800.0	41500
3	1903	77400.0	35200.0	38200
4	1904	36300.0	59400.0	40600

In [4]:
```
pop_df.columns
```

Out[4]:
```
Index(['year', 'hare', 'lynx', 'carrot'], dtype='object')
```

In [5]:
```
pop_df.values
```

Out[5]:
```
array([[ 1900., 30000.,  4000., 48300.],
       [ 1901., 47200.,  6100., 48200.],
       [ 1902., 70200.,  9800., 41500.],
       [ 1903., 77400., 35200., 38200.],
       [ 1904., 36300., 59400., 40600.],
       [ 1905., 20600., 41700., 39800.],
       [ 1906., 18100., 19000., 38600.],
       [ 1907., 21400., 13000., 42300.],
       [ 1908., 22000.,  8300., 44500.],
       [ 1909., 25400.,  9100., 42100.],
       [ 1910., 27100.,  7400., 46000.],
       [ 1911., 40300.,  8000., 46800.],
       [ 1912., 57000., 12300., 43800.],
       [ 1913., 76600., 19500., 40900.],
       [ 1914., 52300., 45700., 39400.],
       [ 1915., 19500., 51100., 39000.],
       [ 1916., 11200., 29700., 36700.],
       [ 1917.,  7600., 15800., 41800.],
       [ 1918., 14600.,  9700., 43300.],
       [ 1919., 16200., 10100., 41300.],
       [ 1920., 24700.,  8600., 47300.]])
```

In [6]:
```
pop_df.dtypes
```

Out[6]:
```
year       int64
hare     float64
lynx     float64
carrot     int64
dtype: object
```

We can access columns (Pandas series) using their labels:

In [7]:
```
hare_df = pop_df["hare"]
hare_df
```

Out[7]:
```
0     30000.0
1     47200.0
2     70200.0
3     77400.0
4     36300.0
5     20600.0
6     18100.0
7     21400.0
8     22000.0
9     25400.0
10    27100.0
11    40300.0
12    57000.0
13    76600.0
14    52300.0
15    19500.0
16    11200.0
17     7600.0
18    14600.0
19    16200.0
20    24700.0
Name: hare, dtype: float64
```

Or alternatively using the label as a property of the dataframe:

In [8]:
```
pop_df.hare
```

Out[8]:
```
0     30000.0
1     47200.0
2     70200.0
3     77400.0
4     36300.0
5     20600.0
6     18100.0
7     21400.0
8     22000.0
9     25400.0
10    27100.0
11    40300.0
12    57000.0
13    76600.0
14    52300.0
15    19500.0
16    11200.0
17     7600.0
18    14600.0
19    16200.0
20    24700.0
Name: hare, dtype: float64
```

The usual numeric operations are available for dataframes or series:

In [9]:
```
print("Mean Hare Population: ", hare_df.mean())
```

```
Mean Hare Population:  34080.95238095238
```

In [13]:
```
print("Mean Populations: \n")
print(pop_df[["hare","lynx","carrot"]].mean())
print("\n")
print("Standard Deviations: \n")
print(pop_df[["hare","lynx","carrot"]].std())
```

```
Mean Populations:

hare      34080.952381
lynx      20166.666667
carrot    42400.000000
dtype: float64


Standard Deviations:

hare      21413.981859
lynx      16655.999920
carrot     3404.555771
dtype: float64
```

The describe() method provides a detailed description of variables:

In [14]:
```
pop_df[["hare","lynx","carrot"]].describe()
```

Out[14]:

	hare	lynx	carrot
count	21.000000	21.000000	21.000000
mean	34080.952381	20166.666667	42400.000000
std	21413.981859	16655.999920	3404.555771
min	7600.000000	4000.000000	36700.000000
25%	19500.000000	8600.000000	39800.000000
50%	25400.000000	12300.000000	41800.000000
75%	47200.000000	29700.000000	44500.000000
max	77400.000000	59400.000000	48300.000000

In [15]:
```
pop_df.describe()
```

Out[15]:

	year	hare	lynx	carrot
count	21.000000	21.000000	21.000000	21.000000
mean	1910.000000	34080.952381	20166.666667	42400.000000
std	6.204837	21413.981859	16655.999920	3404.555771
min	1900.000000	7600.000000	4000.000000	36700.000000
25%	1905.000000	19500.000000	8600.000000	39800.000000
50%	1910.000000	25400.000000	12300.000000	41800.000000
75%	1915.000000	47200.000000	29700.000000	44500.000000
max	1920.000000	77400.000000	59400.000000	48300.000000

A better way to do correlation analysis:

In [16]:
```
pop_df[["hare","lynx","carrot"]].corr()
```

Out[16]:

	hare	lynx	carrot
hare	1.000000	0.071892	-0.016604
lynx	0.071892	1.000000	-0.680577
carrot	-0.016604	-0.680577	1.000000

In [17]:
```
pop_df
```

Out[17]:

	year	hare	lynx	carrot
0	1900	30000.0	4000.0	48300
1	1901	47200.0	6100.0	48200
2	1902	70200.0	9800.0	41500
3	1903	77400.0	35200.0	38200
4	1904	36300.0	59400.0	40600
5	1905	20600.0	41700.0	39800
6	1906	18100.0	19000.0	38600
7	1907	21400.0	13000.0	42300
8	1908	22000.0	8300.0	44500
9	1909	25400.0	9100.0	42100
10	1910	27100.0	7400.0	46000
11	1911	40300.0	8000.0	46800
12	1912	57000.0	12300.0	43800
13	1913	76600.0	19500.0	40900
14	1914	52300.0	45700.0	39400
15	1915	19500.0	51100.0	39000
16	1916	11200.0	29700.0	36700
17	1917	7600.0	15800.0	41800
18	1918	14600.0	9700.0	43300
19	1919	16200.0	10100.0	41300
20	1920	24700.0	8600.0	47300

Also sorting is done easily:

In [18]:
```
pop_df.sort_values(by=['hare'])
```

Out[18]:

	year	hare	lynx	carrot
17	1917	7600.0	15800.0	41800
16	1916	11200.0	29700.0	36700
18	1918	14600.0	9700.0	43300
19	1919	16200.0	10100.0	41300
6	1906	18100.0	19000.0	38600
15	1915	19500.0	51100.0	39000
5	1905	20600.0	41700.0	39800
7	1907	21400.0	13000.0	42300
8	1908	22000.0	8300.0	44500
20	1920	24700.0	8600.0	47300
9	1909	25400.0	9100.0	42100
10	1910	27100.0	7400.0	46000
0	1900	30000.0	4000.0	48300
4	1904	36300.0	59400.0	40600
11	1911	40300.0	8000.0	46800
1	1901	47200.0	6100.0	48200
14	1914	52300.0	45700.0	39400
12	1912	57000.0	12300.0	43800
2	1902	70200.0	9800.0	41500
13	1913	76600.0	19500.0	40900
3	1903	77400.0	35200.0	38200

More examples of accessing and manipulating data in dataframes:

In [25]:
```
# finding all instances when the population of hares is above 50k
hare_above_50K = pop_df.hare>50000
print(hare_above_50K)
print("\n")
print(pop_df[hare_above_50K])
print("\n")
print(pop_df[hare_above_50K].year)
```

```
0     False
1     False
2      True
3      True
4     False
5     False
6     False
7     False
8     False
9     False
10    False
11    False
12     True
13     True
14     True
15    False
16    False
17    False
18    False
19    False
20    False
Name: hare, dtype: bool


    year     hare     lynx  carrot
2   1902  70200.0   9800.0   41500
3   1903  77400.0  35200.0   38200
12  1912  57000.0  12300.0   43800
13  1913  76600.0  19500.0   40900
14  1914  52300.0  45700.0   39400


2     1902
3     1903
12    1912
13    1913
14    1914
Name: year, dtype: int64
```

In [26]:
```
# finding all instances when the population of one of the animal species is
above_50K = (pop_df["hare"]>50000) | (pop_df["lynx"]>50000)
pop_df[above_50K]
```

Out[26]:

	year	hare	lynx	carrot
2	1902	70200.0	9800.0	41500
3	1903	77400.0	35200.0	38200
4	1904	36300.0	59400.0	40600
12	1912	57000.0	12300.0	43800
13	1913	76600.0	19500.0	40900
14	1914	52300.0	45700.0	39400
15	1915	19500.0	51100.0	39000

In [27]:
```
pop2 = pop_df.drop("year", axis=1)
pop2
```

Out[27]:

	hare	lynx	carrot
0	30000.0	4000.0	48300
1	47200.0	6100.0	48200
2	70200.0	9800.0	41500
3	77400.0	35200.0	38200
4	36300.0	59400.0	40600
5	20600.0	41700.0	39800
6	18100.0	19000.0	38600
7	21400.0	13000.0	42300
8	22000.0	8300.0	44500
9	25400.0	9100.0	42100
10	27100.0	7400.0	46000
11	40300.0	8000.0	46800
12	57000.0	12300.0	43800
13	76600.0	19500.0	40900
14	52300.0	45700.0	39400
15	19500.0	51100.0	39000
16	11200.0	29700.0	36700
17	7600.0	15800.0	41800
18	14600.0	9700.0	43300
19	16200.0	10100.0	41300
20	24700.0	8600.0	47300

When necessary, we can convert a dataframe (or a series) into a Numpy array:

In [28]:
```
poptable = np.array(pop2)
poptable
```

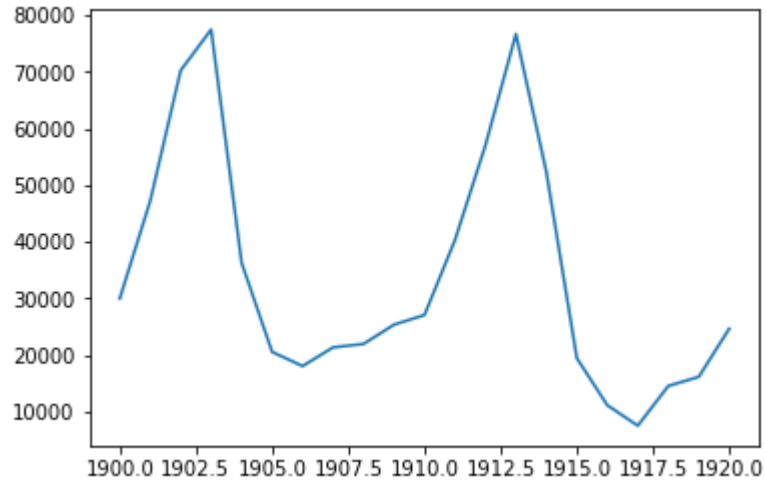
Out[28]:
```
array([[30000.,  4000., 48300.],
       [47200.,  6100., 48200.],
       [70200.,  9800., 41500.],
       [77400., 35200., 38200.],
       [36300., 59400., 40600.],
       [20600., 41700., 39800.],
       [18100., 19000., 38600.],
       [21400., 13000., 42300.],
       [22000.,  8300., 44500.],
       [25400.,  9100., 42100.],
       [27100.,  7400., 46000.],
       [40300.,  8000., 46800.],
       [57000., 12300., 43800.],
       [76600., 19500., 40900.],
       [52300., 45700., 39400.],
       [19500., 51100., 39000.],
       [11200., 29700., 36700.],
       [ 7600., 15800., 41800.],
       [14600.,  9700., 43300.],
       [16200., 10100., 41300.],
       [24700.,  8600., 47300.]])
```

Example of basic visualization using Pandas and with Matplotlib:

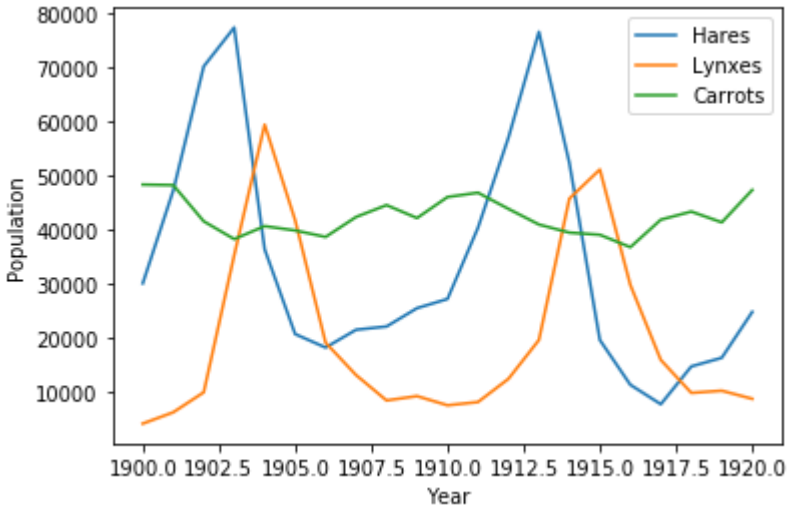
In [29]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [30]:
```
plt.plot(pop_df["year"], pop_df["hare"])
```

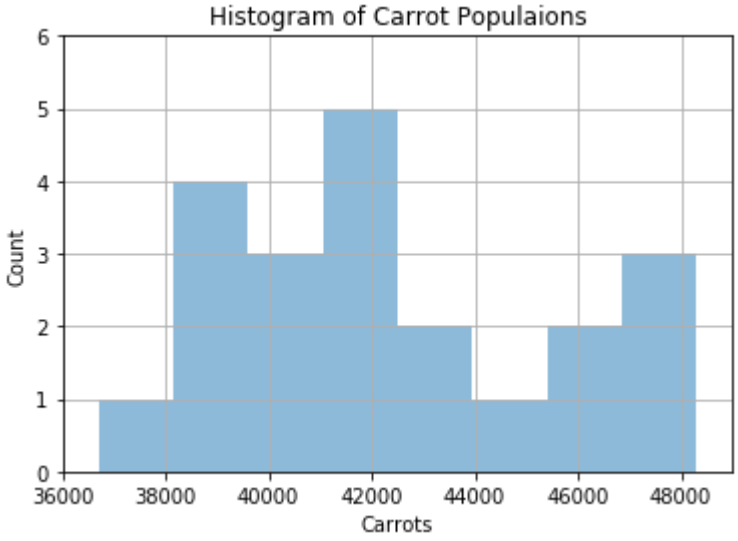
Out[30]:
```
[<matplotlib.lines.Line2D at 0x1b2be21bfc8>]
```



In [31]:
```
plt.plot(pop_df["year"], pop2, label=['Hares','Lynxes','Carrots'])
plt.legend( ('Hares','Lynxes','Carrots') )
plt.ylabel('Population')
plt.xlabel('Year')
plt.show()
```



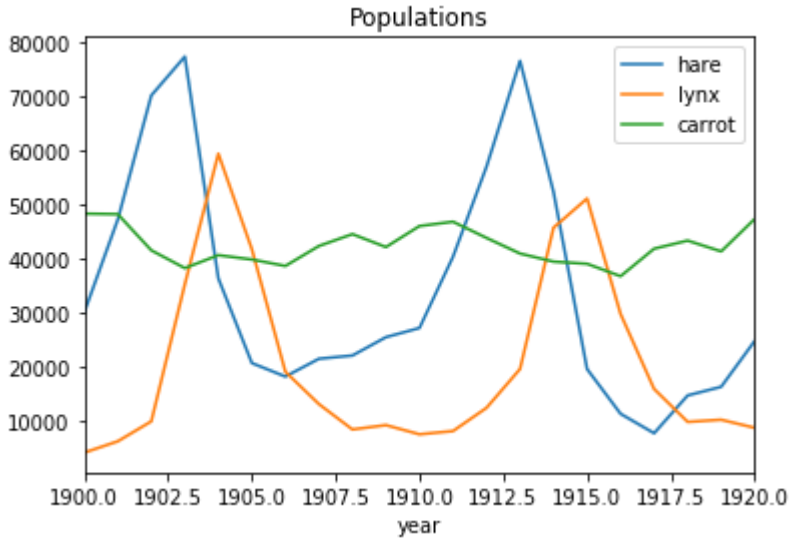
In [32]:
```
plt.hist(pop_df["carrot"], bins=8, alpha=0.5)
plt.xlabel('Carrots')
plt.ylabel('Count')
plt.title('Histogram of Carrot Populaions')
plt.axis([36000, 49000, 0, 6])
plt.grid(True)
```



Pandas has its own versatile "plot" method that can handle most types of charts:

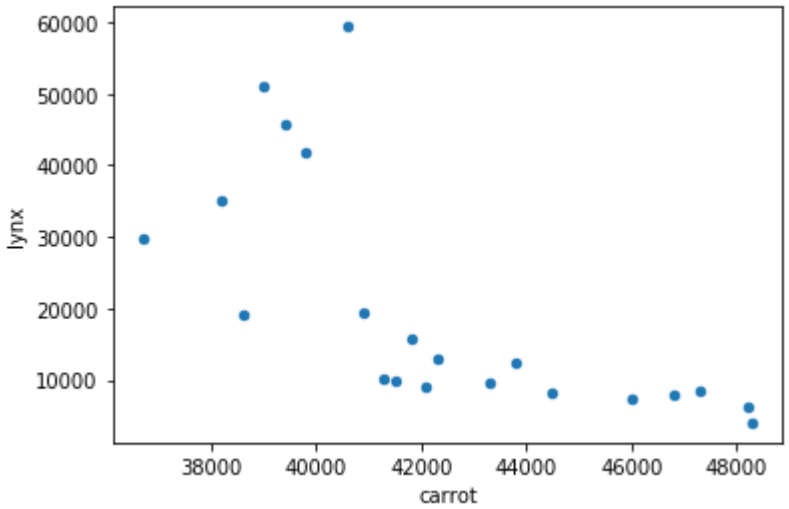
In [33]:
```
pop_df.plot(x="year", title="Populations")
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b2be3caa88>
```



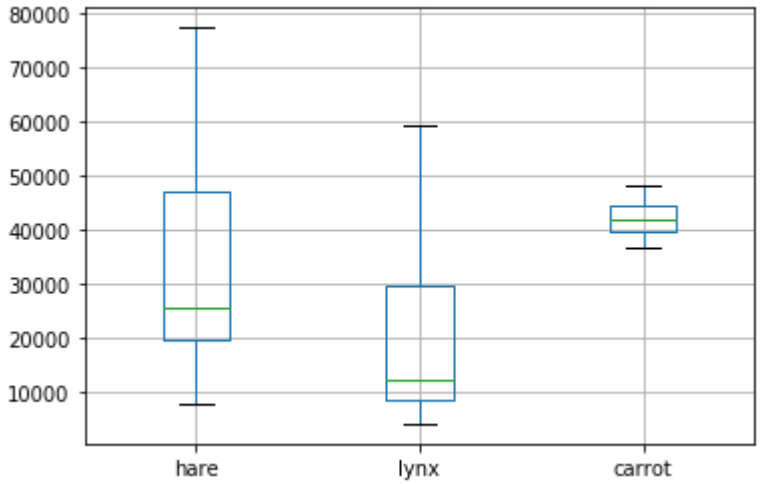
In [34]:
```
pop_df.plot(x="carrot", y="lynx", kind="scatter")
```

Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b2be482648>
```



In [35]:
```
pop_df.boxplot(column=["hare","lynx","carrot"], return_type='axes')
```

Out[35]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b2be4ee588>
```



In [36]:
```
fox_col = np.random.randint(low=5000, high=20000, size=21)
fox_col
```

Out[36]:
```
array([19167, 10516,  8879, 14442,  7983,  6410, 10962, 16344, 12533,
       18666, 10546,  5221, 18250,  5286, 16096, 12953, 11412, 16448,
       10245,  8105, 16696])
```

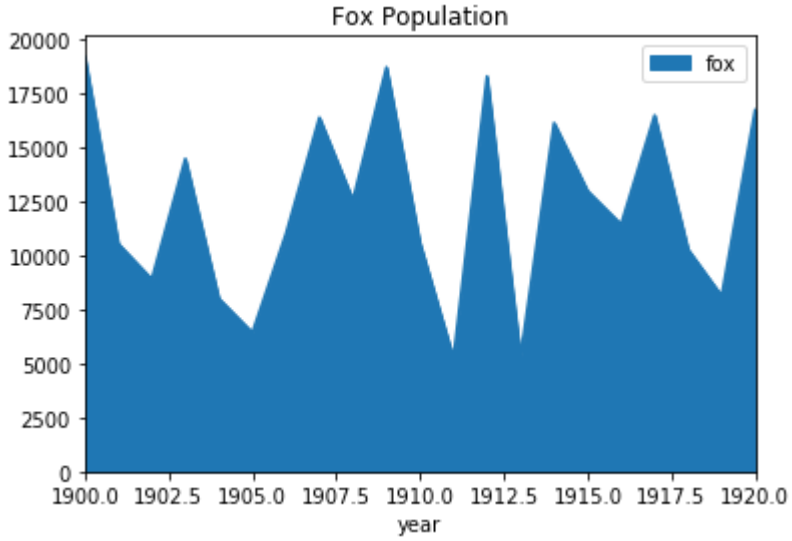
In [37]:
```
pop_df["fox"] = pd.Series(fox_col, index=pop_df.index)
pop_df
```

Out[37]:

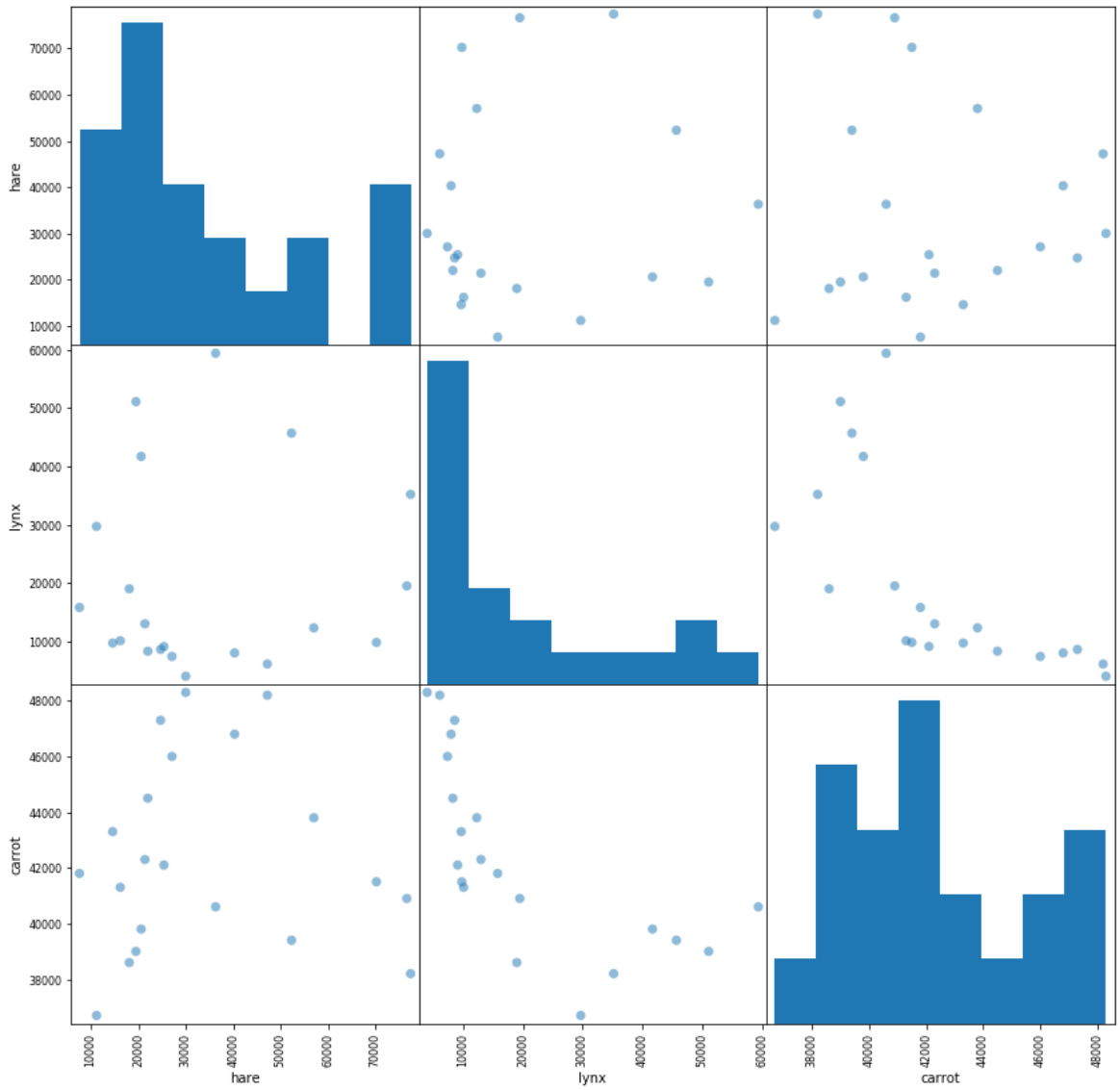
	year	hare	lynx	carrot	fox
0	1900	30000.0	4000.0	48300	19167
1	1901	47200.0	6100.0	48200	10516
2	1902	70200.0	9800.0	41500	8879
3	1903	77400.0	35200.0	38200	14442
4	1904	36300.0	59400.0	40600	7983
5	1905	20600.0	41700.0	39800	6410
6	1906	18100.0	19000.0	38600	10962
7	1907	21400.0	13000.0	42300	16344
8	1908	22000.0	8300.0	44500	12533
9	1909	25400.0	9100.0	42100	18666
10	1910	27100.0	7400.0	46000	10546
11	1911	40300.0	8000.0	46800	5221
12	1912	57000.0	12300.0	43800	18250
13	1913	76600.0	19500.0	40900	5286
14	1914	52300.0	45700.0	39400	16096
15	1915	19500.0	51100.0	39000	12953
16	1916	11200.0	29700.0	36700	11412
17	1917	7600.0	15800.0	41800	16448
18	1918	14600.0	9700.0	43300	10245
19	1919	16200.0	10100.0	41300	8105
20	1920	24700.0	8600.0	47300	16696

In [38]:
```
pop_df.plot(x="year", y="fox", kind="area", title="Fox Population")
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b2be47e548>
```



In [41]:
```
pd.plotting.scatter_matrix(pop_df[["hare","lynx","carrot"]], figsize=(14,14)
```



In [ ]: