Lab 2

Using Python for Exploratory Data Analysis (EDA)

The first part of Lab2 is to go through a demo for EDA of a relatively clean tabular data called *Auto MPG data*. This data set comes from the UCI Machine Learning Data Repository (http://archive.ics.uci.edu/ml/ (http://archive.ics.uci.edu/ml/)) and can be found at http://archive.ics.uci.edu/ml/datasets/Auto+MPG (http://archive.ics.uci.edu/ml/datasets/Auto+MPG). We provide a slightly modified version of this data set as file `cars.csv` .

To load this data set and pursue EDA, it is a good idea to use several popular Python packages, which come preloaded with the Anaconda software:

- pandas (http://pandas.pydata.org/ (http://pandas.pydata.org/)) -- a library for data science
- numpy (http://www.numpy.org/ (http://www.numpy.org/)) -- a library for data computations
- matplotlib (http://matplotlib.org/ (http://matplotlib.org/)) -- a library for data visualization

The main feature of *pandas* is its `DataFrame` data structure that provides an intuitive way of handling tabular data. The main feature of *numpy* is its `array` data structure that represents matrices and allows us to perform matrix algebra operations. *matplotlib* allows visualizing data stored in `DataFrame` or `array` objects.

As you will soon realize, each new Python library requires spending some time to learn about it. There are several nice tutorials that you can find on the web that get you started with the 3 libraries. A particularly great resource for learning about those libraries is your textbook *Python for Data Analysis*, so please take some time to browse its contents and try to run the code provided in it. When learning about the new libraries, you are best advised to jump in and immediatelly start tinkering with the code. The more time you spend using the library, the more you will uncover about all the great features and posibilities the library offers you.

Let us start by loading the 3 libraries in a particular way that many data scientists prefer.

In [42]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
# the following line allows ipython to display plots
%matplotlib inline
```

Question 1: What are we accomplishing with `as` reserved word?

'as' is used to create an alias while importing a module. Meaning imported modules can be given user-defined names.

`cars.csv` is in an easy-to-read comma separated format and the following *pandas* functionality makes it easy to read it into a `DataFrame` object.

In [43]:

```
# read this csv file, remember to put the full path to
# the directory where you saved the data
df = pd.read_csv('cars.csv')  # df is DataFrame object
print (df.head())    # see the first 5 rows of the loaded table
```

```
                       Car   MPG  Cylinders  Displacement  Horsepowe
r  \
0  Chevrolet Chevelle Malibu  18.0          8         307.0         13
0
1          Buick Skylark 320  15.0          8         350.0         16
5
2         Plymouth Satellite  18.0          8         318.0         15
0
3              AMC Rebel SST  16.0          8         304.0         15
0
4                Ford Torino  17.0          8         302.0         14
0

   Weight  Acceleration  Model Origin
0    3504          12.0     70     US
1    3693          11.5     70     US
2    3436          11.0     70     US
3    3433          12.0     70     US
4    3449          10.5     70     US
```

Question 2: How can you display the first 10 rows using method `head` ? What are the types of each of the columns in `df` ?

In [44]:

```
print (df.head(10))  # see the first 10 rows of the loaded table
```

```
                        Car   MPG  Cylinders  Displacement  Horsepowe
r  \
0  Chevrolet Chevelle Malibu  18.0          8         307.0         13
0
1          Buick Skylark 320  15.0          8         350.0         16
5
2         Plymouth Satellite  18.0          8         318.0         15
0
3              AMC Rebel SST  16.0          8         304.0         15
0
4                Ford Torino  17.0          8         302.0         14
0
5          Ford Galaxie 500  15.0          8         429.0         19
8
6          Chevrolet Impala  14.0          8         454.0         22
0
7         Plymouth Fury iii  14.0          8         440.0         21
5
8          Pontiac Catalina  14.0          8         455.0         22
5
9         AMC Ambassador DPL  15.0          8         390.0         19
0

   Weight  Acceleration  Model Origin
0    3504          12.0     70     US
1    3693          11.5     70     US
2    3436          11.0     70     US
3    3433          12.0     70     US
4    3449          10.5     70     US
5    4341          10.0     70     US
6    4354           9.0     70     US
7    4312           8.5     70     US
8    4425          10.0     70     US
9    3850           8.5     70     US
```

Types of columns: Car, MPG, Cylinders, Displacement, Horsepower, Weight, Accerleration, Model, Origin

There are different ways of exploring and indexing the table. Here are some examples.

In [45]:

```
print (list(df.columns));
print (df[0:5]);          # print the first 5 rows, same outcome as df.head()
print (df[['Car', 'MPG']][:10]);  # print the first 10 rows for selected columns
print (df[df['MPG'] > 40]);   # using Boolean condition, print only cars with MPG >
print (df.iloc[[0,1,5],0:5]);   # uses 'ix' indexing, selects rows and columns base
```

```
['Car', 'MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight', 'A
cceleration', 'Model', 'Origin']
                       Car   MPG  Cylinders  Displacement  Horsepowe
r  \
0  Chevrolet Chevelle Malibu  18.0          8         307.0         13
0
1          Buick Skylark 320  15.0          8         350.0         16
5
2         Plymouth Satellite  18.0          8         318.0         15
0
3              AMC Rebel SST  16.0          8         304.0         15
0
4                Ford Torino  17.0          8         302.0         14
0

   Weight  Acceleration  Model Origin
0    3504          12.0     70     US
1    3693          11.5     70     US
2    3436          11.0     70     US
3    3433          12.0     70     US
4    3449          10.5     70     US
                       Car   MPG
0  Chevrolet Chevelle Malibu  18.0
1          Buick Skylark 320  15.0
2         Plymouth Satellite  18.0
3              AMC Rebel SST  16.0
4                Ford Torino  17.0
5           Ford Galaxie 500  15.0
6           Chevrolet Impala  14.0
7          Plymouth Fury iii  14.0
8           Pontiac Catalina  14.0
9         AMC Ambassador DPL  15.0
                               Car   MPG  Cylinders  Displacement  \
251  Volkswagen Rabbit Custom Diesel  43.1          4          90.0
316                Volkswagen Rabbit  41.5          4          98.0
329                        Mazda GLC  46.6          4          86.0
331                      Datsun 210  40.8          4          85.0
332     Volkswagen Rabbit C (Diesel)  44.3          4          90.0
333       Volkswagen Dasher (diesel)  43.4          4          90.0
336              Honda Civic 1500 gl  44.6          4          91.0
337            Renault Lecar Deluxe  40.9          4          85.0
402                Volkswagen Pickup  44.0          4          97.0

     Horsepower  Weight  Acceleration  Model  Origin
251          48    1985          21.5     78  Europe
316          76    2144          14.7     80  Europe
329          65    2110          17.9     80   Japan
331          65    2110          19.2     80   Japan
332          48    2085          21.7     80  Europe
333          48    2335          23.7     80  Europe
336          67    1850          13.8     80   Japan
337           0    1835          17.3     80  Europe
402          52    2130          24.6     82  Europe
                      Car   MPG  Cylinders  Displacement  Horsepowe
```

```
r
0  Chevrolet Chevelle Malibu  18.0          8         307.0          13
0
1           Buick Skylark 320  15.0          8         350.0          16
5
5            Ford Galaxie 500  15.0          8         429.0          19
8
```

Question 3: Show two ways of printing the last 5 rows of `df` . Print the names of the cars with 3 cilinders.

In [46]:

```
print df.tail(5)
```

```
                   Car   MPG  Cylinders  Displacement  Horsepower  Wei
ght  \
401    Ford Mustang GL  27.0          4         140.0          86    2
790
402  Volkswagen Pickup  44.0          4          97.0          52    2
130
403      Dodge Rampage  32.0          4         135.0          84    2
295
404        Ford Ranger  28.0          4         120.0          79    2
625
405         Chevy S-10  31.0          4         119.0          82    2
720

     Acceleration  Model  Origin
401          15.6     82      US
402          24.6     82  Europe
403          11.6     82      US
404          18.6     82      US
405          19.4     82      US
```

In [47]:

```
print df.tail()
```

```
                 Car   MPG  Cylinders  Displacement  Horsepower  Wei
ght  \
401    Ford Mustang GL  27.0          4         140.0          86    2
790
402  Volkswagen Pickup  44.0          4          97.0          52    2
130
403      Dodge Rampage  32.0          4         135.0          84    2
295
404        Ford Ranger  28.0          4         120.0          79    2
625
405         Chevy S-10  31.0          4         119.0          82    2
720

     Acceleration  Model  Origin
401          15.6     82      US
402          24.6     82  Europe
403          11.6     82      US
404          18.6     82      US
405          19.4     82      US
```

In [48]:

```
print (df[df['Cylinders'] == 3])
```

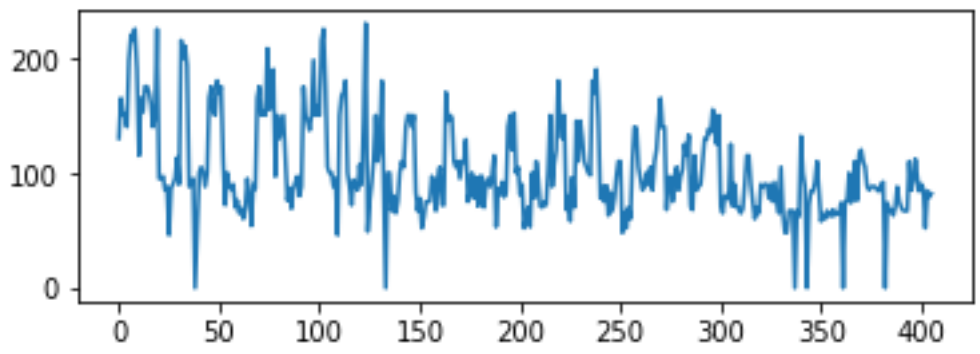
```
                Car   MPG  Cylinders  Displacement  Horsepower  Weigh
t  \
78   Mazda RX2 Coupe  19.0          3          70.0          97    233
0
118        Mazda RX3  18.0          3          70.0          90    212
4
250       Mazda RX-4  21.5          3          80.0         110    272
0
341    Mazda RX-7 GS  23.7          3          70.0         100    242
0

     Acceleration  Model Origin
78           13.5     72  Japan
118          13.5     73  Japan
250          13.5     77  Japan
341          12.5     80  Japan
```

Now, we are ready to start plotting the data.

In [49]:

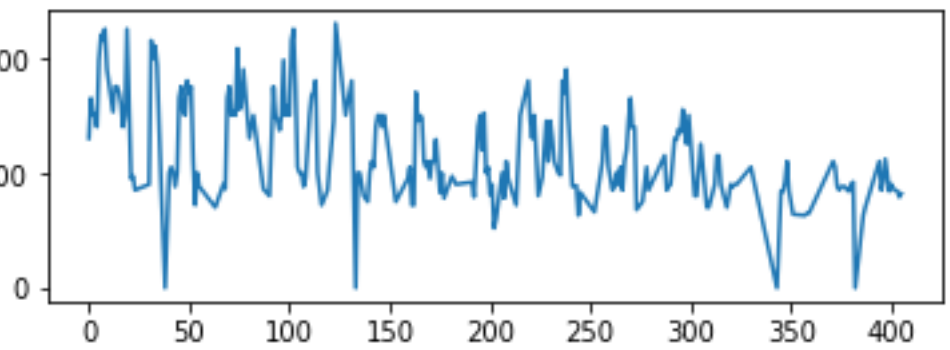
```
plt.figure(figsize=(6,2))  # can control the size of the display
plt.plot(df['Horsepower']);      # display 'Model' attribute
```



Question 4: Plot *Horsepower* attribute, but only for the US cars.

In [50]:

```
plt.figure(figsize=(6,2))
x = df[df['Origin'] == 'US']
plt.plot(x['Horsepower']);
```

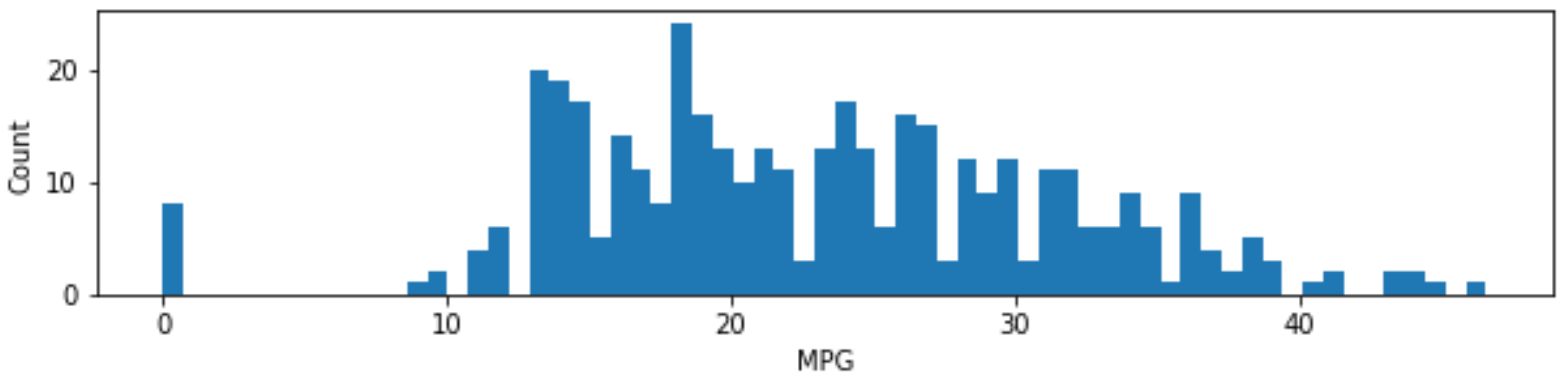


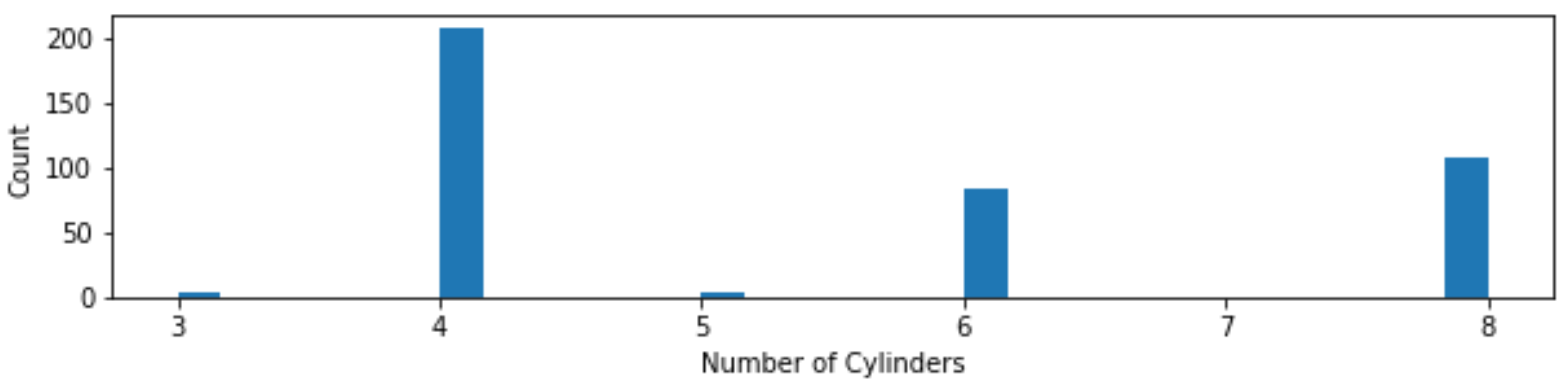
To plot the histogram of a selected attribute:

In [51]:

```
fig = plt.figure(figsize=(10,2))
plt.hist(df['MPG'], bins=65)    # ; suppresses Out
plt.xlabel('MPG')
plt.ylabel('Count');

fig1 = plt.figure(figsize=(10,2))
plt.hist(df['Cylinders'], bins=30)
plt.xlabel('Number of Cylinders')
plt.ylabel('Count');
```





Question 5: What can we conclude by looking at the histogram? Explain in one-two sentences. Figure out what is your preferred number of bins for *MPG* and *Cylinders* and argue why.

We can conclude the majority of cars available in 1970, fell between ten (10) and forty (40) MPG. Secondly, we can determine that five different cylinder options were available for the cars in our data set.
For MPG, I believe that sixty-five (65) bins is the best option because it allows us to see the individual groupings of MPG much easier than the lower bin-count options. With the lower options the bars connect to eachother due to lack of space. Above sixty-five bins the bars don't separate any further, illustrating that additional bins wouldn't improve the depiction of our data.

For Cylinders, I believe thirty (30) bins represents the data as clear as possible, before thirty the bars are so thick, they could be representing multiple numbers (for example, 4.0 and 4.1). Additionally, I chose thirty because it aligned the left side of the bars on the graph with the right side of the number they represent, with the exception of 8, which I thought looked better.

The following code provides statistics for number of cylinders.

In [52]:

```
t = pd.crosstab(index=df["Cylinders"],  # Make a crosstab
                columns="count")      # Name the count column
t['percentage']= (t/t.sum())*100
print (t);
```

```
col_0       count  percentage
Cylinders
3               4    0.985222
4             207   50.985222
5               3    0.738916
6              84   20.689655
8             108   26.600985
```

Question 6: Try to learn more about `crosstab` method (by doing Google search) and write a line of code that uses it in a different way on `df` data.

In [53]:

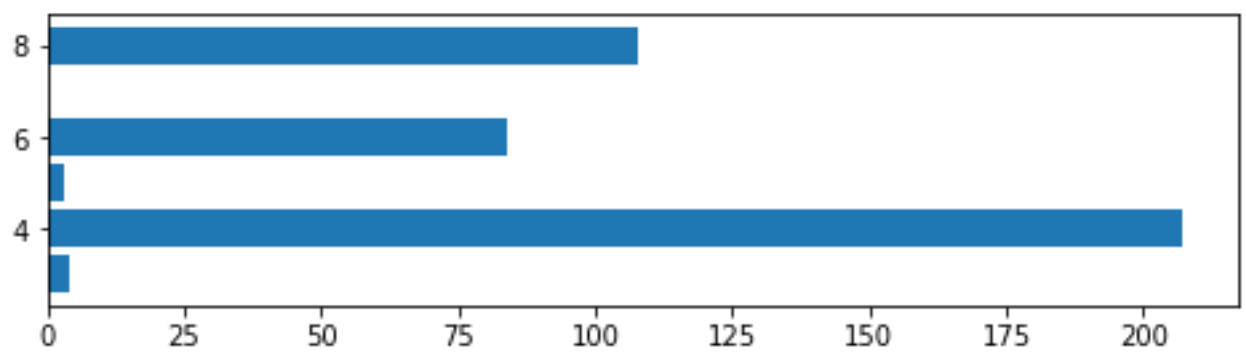
```
t1 = pd.crosstab(df["Model"], df["Cylinders"], rownames=['model'], colnames=['Cylinc
print (t1);
```

```
Cylinders  3   4  5   6   8
model
70         0   8  0   4  23
71         0  14  0   8   7
72         1  14  0   0  13
73         1  11  0   8  20
74         0  15  0   7   5
75         0  12  0  12   6
76         0  15  0  10   9
77         1  14  0   5   8
78         0  17  1  12   6
79         0  12  1   6  10
80         1  25  1   2   0
81         0  22  0   7   1
82         0  28  0   3   0
```

Horizontal bar plot:

In [54]:

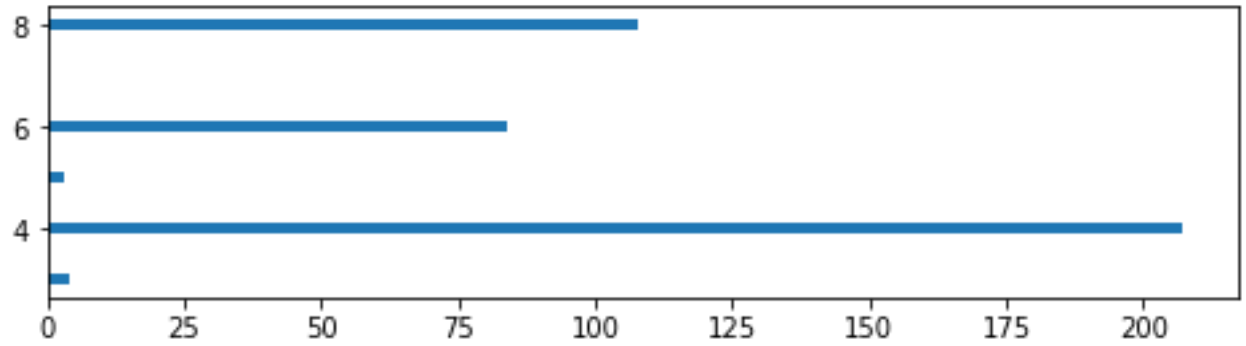
```
plt.figure(figsize=(8,2))
plt.barh(t.index, t["count"]);
```



Question 7: How about a horizontal bar plot? Can you learn how to control the width of bars and make a plot that has thinner bars?

In [55]:

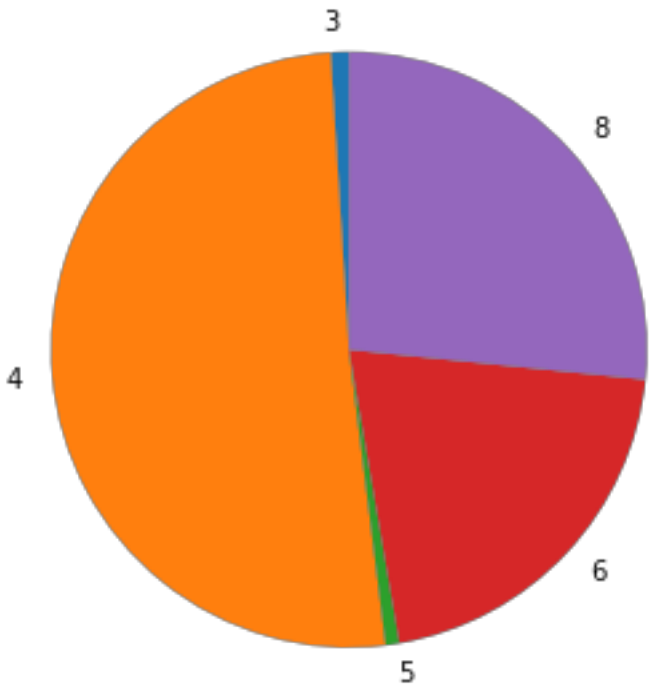
```
plt.figure(figsize=(8,2))
width = 1/4.9
plt.barh(t.index, t["count"], width);
```



Pie chart:

In [56]:

```
plt.figure(figsize=(5,5))
plt.axis("equal")
plt.pie(t["count"],labels=t.index,startangle=90);
```



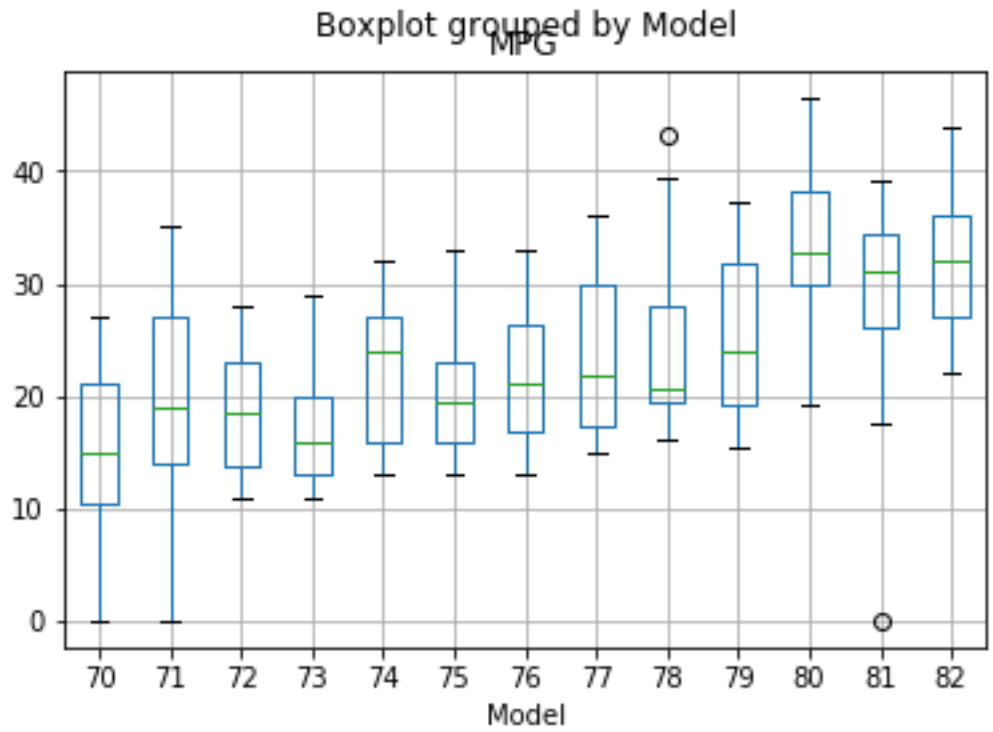
Question 8: Do you prefer bar or pie displayed chart and why?

I prefer bar charts for a couple reasons. For one, as the horizontal-bar chart above shows, it is much easier to roughly guess how many Cylinders exist in the current dataframe. Secondly, information on data is difficult to accurately collect from the pie chart, but it is helpful for seeing what attributes may be dominating the dataset.

The following is a boxplot of MPG values for each of the model years. Pay attention that matplotlib is not used here. Instead, we called a panda `boxplot` method

In [57]:

```
df.boxplot(column='MPG',by='Model');
```

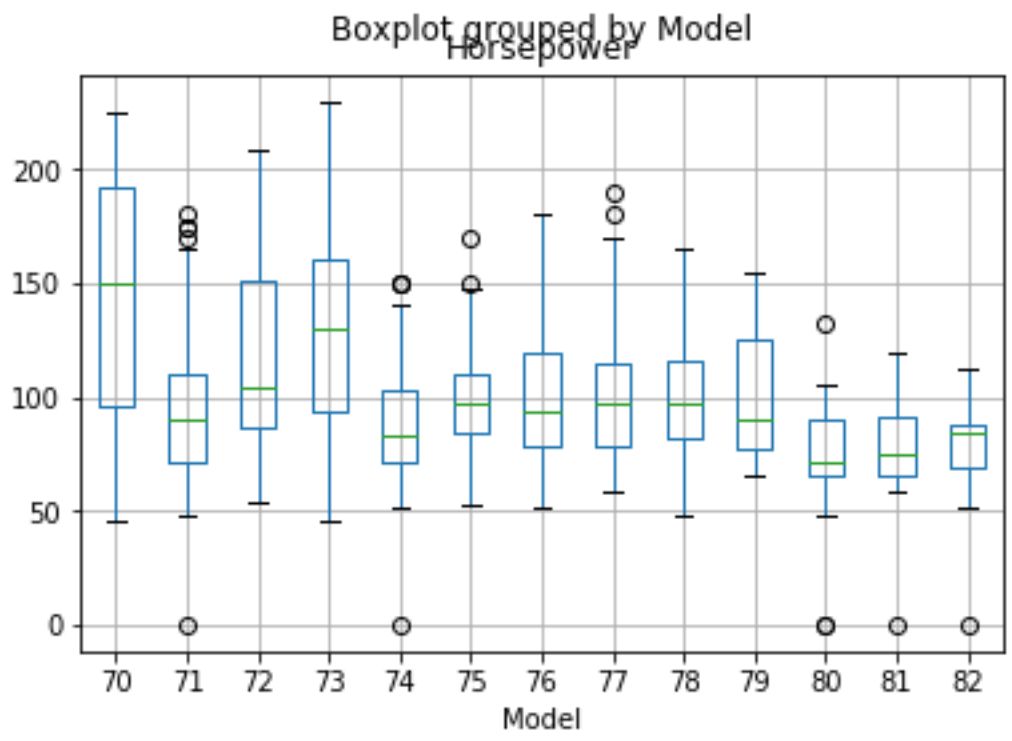


Question 9: Discuss what can you learn from the displayed boxplot. Plot another boxplot using `df` data that you think is very useful and explain what can we learn from it?

I learned that the median MPG grew from roughly fifteen to over thirty in a decade, that's a pretty impressive growth rate. It makes me wonder why cars have such low gas mileage today, I think the answer lies somewhere in the oil company-car company connections.

In [58]:

```
df.boxplot(column='Horsepower',by='Model');
```

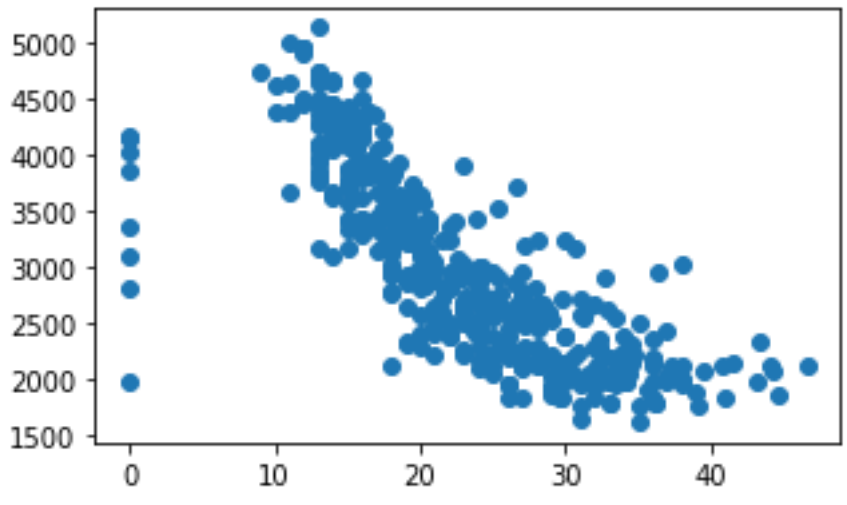


I learned that the Horsepower over the span of the dataset model years reduced greatly with even the highest outlier in 1982 being roughly 100 less than the highest outlier in 1970.

Scatterplot between MPG and Weight attributes:

In [59]:

```
plt.figure(figsize=(5,3))
plt.scatter(df['MPG'],df['Weight']);
```

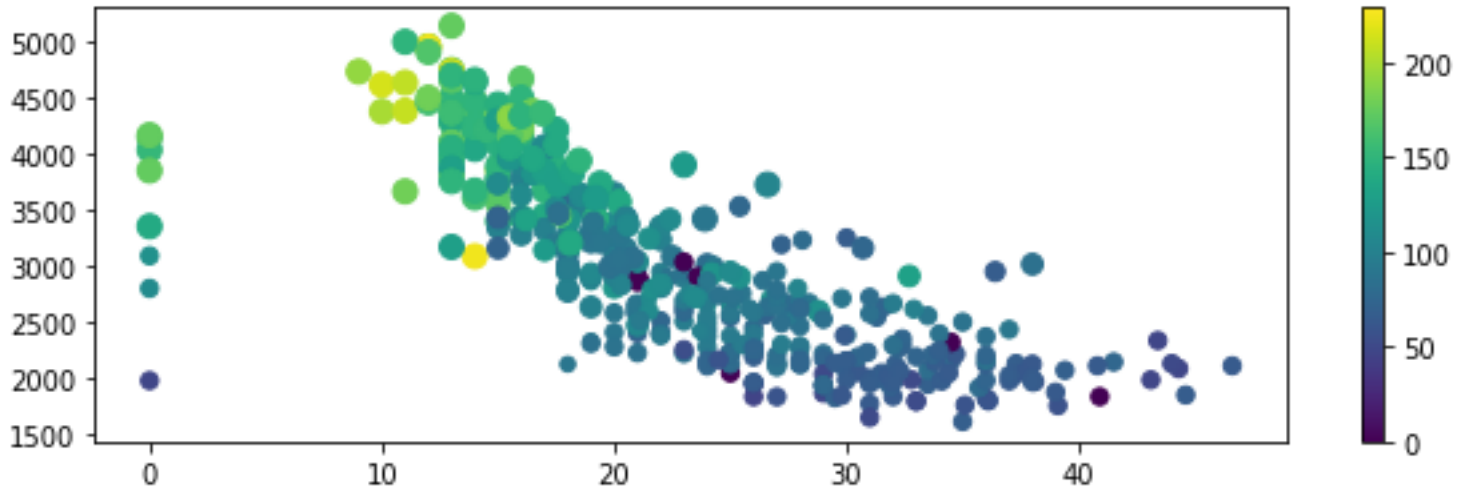


Question 10: Discuss what can we learn from the displayed scatterplot.

From the above scatterplot we can determine that cars with lower weight also had higher MPG.

In [60]:

```
plt.figure(figsize=(10,3))
plt.scatter(df['MPG'],df['Weight'],df['Cylinders']*10,df['Horsepower']);
plt.colorbar();
```

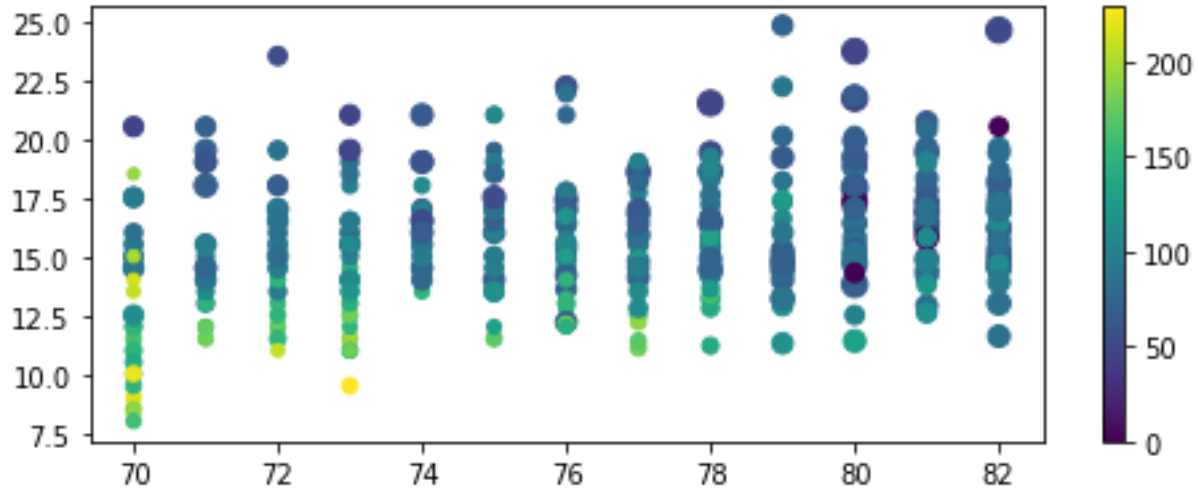


Question 11: Discuss what additional insight we can get from this scatterplot as compared to the previous scatterplot. Plot another scatterplot by picking a different set of attributes that you think is providing a useful view at the data. Discuss what can be concluded from that plot.

From the scatterplot we can clearly identify the 'Horsepower' based on the color of each point. Additionally, we can determine the number of cylinders by the size of the points on the graph. The number of 'Cylinders' is multipled by ten (10) to make identifying differences in size easier, also, without the multiplication the points are very small and not as pleasant to look at.

In [61]:

```
plt.figure(figsize=(8,3))
plt.scatter(df['Model'],df['Acceleration'],df['MPG']*2,df['Horsepower']);
plt.colorbar();
```

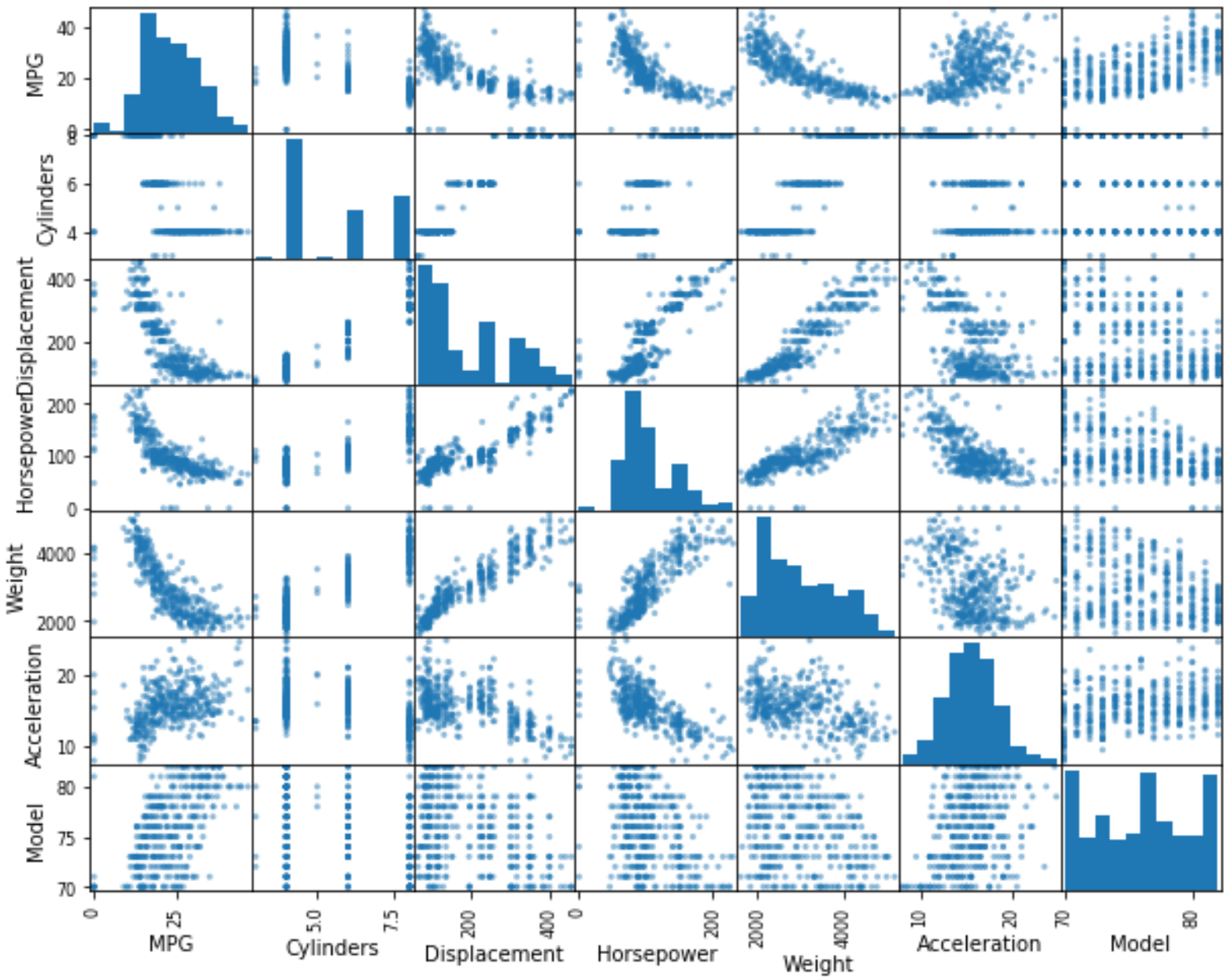


We can conclude while Horsepower dropped over the 12 years of our dataset, Acceleration continued to steadily increase ever so slightly.

Pandas `scatter_matrix` method allows us to plot all scatterplots for a data set (it would take a few seconds to display):

In [62]:

```
pd.plotting.scatter_matrix(df, figsize=(10, 8));
```



Question 12: Explain what are we seeing from this plot and discuss about the insights you obtained from it.

When you are done with running and playing with the code provided in this file and answering Questions 1-12, **submit** `.ipynb` file containing modifications of this file together with your answers and comments. Feel free to modify the provided code or produce new lines of code.

Question 13: Produce a 2 page word document titled "Exploratory Analysis of MPG Data Set". In this document you should combine your own discussion and figures produced by Python to provide a coherent story about the properties of the MPG data set and the most important and interesting insights about the data. You can feel free to frame your story around some known historical facts about the cars and U.S. and World economy during the 1970-1982 period. **Submit** the document as `.pdf` file.

Each square has an x and y axis representing different attributes in our dataset. It seems that we can see the relationships between each attribute and every other attribute.

We can determine that the relationship between Weight and MPG is negatively correlated. That is also true for MPG and Horsepower. Additionally, we can see that the relationship between Horsepower and Weight is positively correlated.