In [15]:

```
import cv2
import operator
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers.experimental import preprocessing
print(tf.__version__)
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
2.5.0-dev20201024
```

In [7]:

```
url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data'
column_names = ['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
                'Acceleration', 'Model Year', 'Origin']

raw_dataset = pd.read_csv(url, names=column_names,
                          na_values='?', comment='\t',
                          sep=' ', skipinitialspace=True)
```

In [8]:

```
dataset=raw_dataset.copy()
dataset.tail()
```

Out[8]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

In [10]:

```
dataset.isna().sum()
```

Out[10]:

```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

In [11]:

```
dataset=dataset.dropna()
```

In [12]:

```
dataset['Origin'] = dataset['Origin'].map({1: 'USA', 2: 'Europe', 3: 'Japan'})
```

In [13]:

```
dataset = pd.get_dummies(dataset, prefix='', prefix_sep='')
dataset.tail()
```

Out[13]:

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Europe	Japan	USA
393	27.0	4	140.0	86.0	2790.0	15.6	82	0	0	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	1	0	0
395	32.0	4	135.0	84.0	2295.0	11.6	82	0	0	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	0	0	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	0	0	1

In [19]:

```
X=dataset[['Cylinders','Displacement','Horsepower','Weight','Acceleration','Model Year','Europe','Japan','USA']]
y=dataset[['MPG']]
```

In [21]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [22]:

```
train_dataset.describe().transpose()[['mean', 'std']]
```

Out[22]:

	mean	std
MPG	23.599361	7.956255
Cylinders	5.482428	1.700446
Displacement	195.517572	103.766567
Horsepower	104.594249	38.283669
Weight	2986.124601	841.133957
Acceleration	15.544089	2.817864
Model Year	76.207668	3.630136
Europe	0.153355	0.360906
Japan	0.201278	0.401597
USA	0.645367	0.479168

In [23]:

```
normalizer = preprocessing.Normalization()
```

In [25]:

```
normalizer.adapt(np.array(X_train))
```

In [26]:

```
print(normalizer.mean.numpy())
```

```
[5.48242807e+00 1.95517578e+02 1.04594246e+02 2.98612451e+03
 1.55440893e+01 7.62076645e+01 1.53354630e-01 2.01277956e-01
 6.45367384e-01]
```

In [28]:

```
horsepower = np.array(X_train['Horsepower'])

horsepower_normalizer = preprocessing.Normalization(input_shape=[1,])
horsepower_normalizer.adapt(horsepower)
```

In [30]:

```
horsepower_model=tf.keras.Sequential([horsepower_normalizer,
                                      layers.Dense(units=1)])
horsepower_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
normalization_1 (Normalizati (None, 1)                 3
_________________________________________________________________
dense (Dense)                (None, 1)                 2
=================================================================
Total params: 5
Trainable params: 2
Non-trainable params: 3
_________________________________________________________________
```

In [ ]:

```
This model will predict MPG from Horsepower
```

In [31]:

```
horsepower_model.predict(horsepower[:10])
```

Out[31]:

```
array([[ 0.11686753],
       [-0.27227628],
       [-0.16418077],
       [ 0.9816316 ],
       [-0.8776111 ],
       [ 1.8463956 ],
       [-0.92084926],
       [ 0.11686753],
       [ 0.22496304],
       [ 0.11686753]], dtype=float32)
```

In [32]:

```
horsepower_model.compile(
    optimizer=tf.optimizers.Adam(learning_rate=0.1),
    loss='mean_absolute_error')
```

In [34]:

```
%%time
history = horsepower_model.fit(
    X_train['Horsepower'], y_train,
    epochs=100,
    # suppress logging
    verbose=0,
    # Calculate validation results on 20% of the training data
    validation_split = 0.2)
```

```
CPU times: user 22.3 s, sys: 531 ms, total: 22.8 s
Wall time: 22.6 s
```

In [35]:

```
hist = pd.DataFrame(history.history)
hist['epoch'] = history.epoch
hist.tail()
```

Out[35]:

	loss	val_loss	epoch
95	3.755007	4.136340	95
96	3.757320	4.125999	96
97	3.753460	4.137080	97
98	3.754762	4.158894	98
99	3.754327	4.159886	99

In [37]:

```
test_results = {}

test_results['horsepower_model'] = horsepower_model.evaluate(
    X_test['Horsepower'],
    y_test, verbose=0)
```

In [38]:

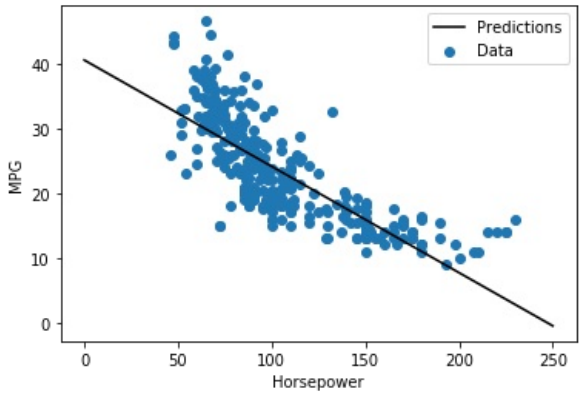
```
x = tf.linspace(0.0, 250, 251)
y = horsepower_model.predict(x)
```

In [48]:

```
def plot_horsepower(x, y):
  plt.scatter(X_train['Horsepower'], y_train, label='Data')
  plt.plot(x, y, color='k', label='Predictions')
  plt.xlabel('Horsepower')
  plt.ylabel('MPG')
  plt.legend()
```

In [49]:

```
plot_horsepower(x,y)
```



In [ ]:

```

```