```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Example 2.2
Polynomial regression
Machine Learning for Engineering Problem Solving
@author: Austin Downey
"""

import IPython as IP
IP.get_ipython().run_line_magic('reset', '-sf')

import numpy as np
import scipy as sp
import matplotlib as mpl
import matplotlib.pyplot as plt
import sklearn as sk
from sklearn import linear_model

plt.close('all')

#%% build the data sets
np.random.seed(2) # 2 and 6 are pretty good
m = 100
X = 6 * np.random.rand(m,1) - 3
Y = 0.5 * X**2 + X + 2 + np.random.randn(m,1)

# plot the data
plt.figure()
plt.grid(True)
plt.scatter(X,Y)
plt.xlabel('x')
plt.ylabel('y')

#%% perform  polynomial regression

# generate x^2 as we use the model y = a*x^2* + b*x + c
X_poly_manual = np.hstack((X,X**2))

# or use the code as this does lots of features for multi-feature data sets.
poly_features = sk.preprocessing.PolynomialFeatures(degree=2, include_bias=False)
X_poly_sk = poly_features.fit_transform(X)

# they do do same thing as shown below, so select one to carry forward.
print(X_poly_manual == X_poly_sk)
X_poly = X_poly_manual

# In essence, we now have two data sets. We can plot that here
plt.figure()
plt.grid(True)
plt.scatter(X_poly[:,0],Y,label = 'data for x')
plt.scatter(X_poly[:,1],Y,marker='s',label = 'data for $x^2$')
plt.legend()
plt.xlabel('x')
plt.ylabel('y')

# and fit linear models to these data sets
model = sk.linear_model.LinearRegression() # Select a linear model
model.fit(X_poly,Y) # Train the model
X_model_1 = np.linspace(-3,3)
X_model_2 = np.linspace(0,9)

# the model parameters are:
model_coefficients = model.coef_
model_intercept = model.intercept_
print(model_coefficients)
print(model_intercept)
```

```python

Y_X1 = model_coefficients[0][0]*X_model_1 + model_intercept
Y_X2 = model_coefficients[0][1]*X_model_2 + model_intercept

# now if we plot the linear models on the extended set of features.
plt.figure()
plt.grid(True)
plt.scatter(X_poly[:,0],Y,label = 'data for x')
plt.scatter(X_poly[:,1],Y,marker='s',label = 'data for $x^2$')
plt.plot(X_model_1,Y_X1,'--',label='inear fit $x$')
plt.plot(X_model_2,Y_X2,':',label='linear fit for $x^2$',)
plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.savefig('example_6_fig_1',dpi=300)

# now that we have a parameter for x and x^2, these can be recombined into a single
# model, y = x^2*a + x*b + c.
Y_polynominal = model_coefficients[0][1]*X_model_1**2 + model_coefficients[0][0]*\
    X_model_1 + model_intercept

plt.figure()
plt.grid(True)
plt.scatter(X,Y,label='data')
plt.plot(X_model_1,Y_polynominal,'r--',label='polynominal fit')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.savefig('example_6_fig_2',dpi=300)
```