# Homework2: Wooseok Kim

In [3]:
```
import matplotlib.pyplot as plt
import numpy as np
from numpy.linalg import inv
```

I will use matplotlib, numpy and numpy.linalg for this homework.

In [2]:
```
data = np.loadtxt('hw2.dat', delimiter=' ')
```

Let's load data file using loadtxt

In [4]:
```
x, y = data[:,0], data[:,1]
data_len = np.size(x)
```

A first column of the data is x. A second column of the data is y. data_len is the size of the first column, which is 300.

In [5]:
```
x_np, y_np = np.array([x]), np.array([y])
ones_arr = np.ones(data_len)
ones_arr_np = np.array([ones_arr])
```

All the data have 1 in an array

I used basis functions and normal equation in lecture note.

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$p = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix}$$

$$F = \begin{bmatrix} g_1(x_1) & g_2(x_1) & g_3(x_1) & \dots & g_d(x_1) \\ g_1(x_2) & g_2(x_2) & x_{23} & \dots & g_d(x_2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ g_1(x_n) & g_2(x_n) & x_{d3} & \dots & g_d(x_n) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^k \\ 1 & x_2 & x_2^2 & \dots & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^k \end{bmatrix}$$

$$X^T X p = X^T y$$

$$p = (X^T X)^{-1} X^T y$$

## Degree - 1

In [8]:
```
A = np.concatenate((ones_arr_np.T, x_np.T), axis=1) # Degree-1
```

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}$$

In [9]:
```
C = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y)
```

$$p = (X^T X)^{-1} X^T y$$

In [20]:
```
x_draw = np.linspace(-10, 10, 10000)
y_draw, sum_x = 0, 0
```

In [11]:
```
for i in range(2):
        y_draw += C[i]*(x_draw**i)
        sum_x += C[i]*(x**i)
```

$$y\_draw = C[0] + C[1] * (x\_draw^1)$$
$$sum\_x = C[0] + C[1] * (x^1)$$

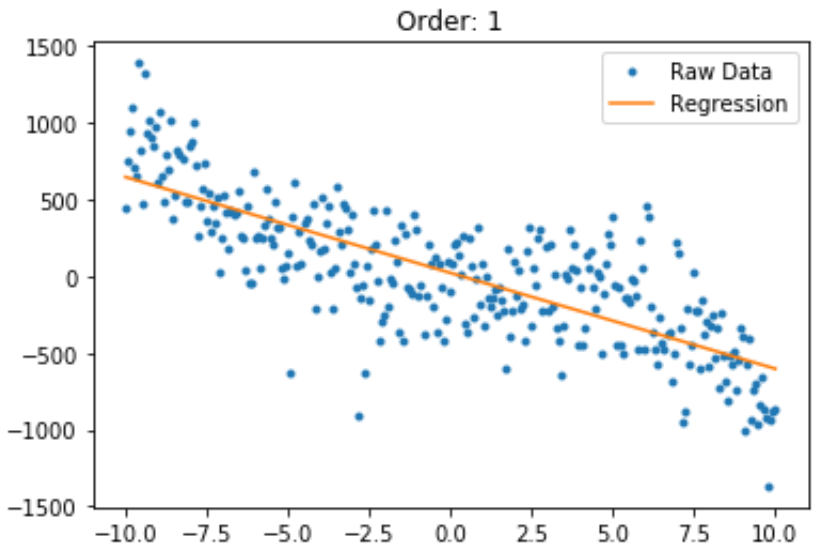
In [12]:
```
sum_square = np.sum((y-sum_x)**2)
```

$$sum\_square = \sum_{i=0}^{n} (y_i - sum\_x_i)^2$$

In [15]:
```
print("\nCoefficient(order-1): ", C)
print("Residual: ", sum_square, "\nOptimal fit error: ", sum_square/np
.size(x))
```

```
Coefficient(order-1):  [ 21.463462   -62.50802079]
Residual:  24428690.735389516
Optimal fit error:  81428.96911796505
```

In [16]:
```
plt.title("Order: 1")
plt.plot(x, y, '.', label='Raw Data')
plt.plot(x_draw, y_draw, '-', label='Regression')
plt.legend(loc='best')
plt.show()
```




## Degree - 3

In [18]:
```
A = np.concatenate((ones_arr_np.T, x_np.T, (x_np**2).T, (x_np**3).T),
axis=1)
```

$$\begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}$$

In [19]:
```
C = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y)
```

$$p = (X^T X)^{-1} X^T y$$

In [21]:
```
x_draw = np.linspace(-10, 10, 10000)
y_draw, sum_x = 0, 0
```

In [22]:
```
y_draw, sum_x = 0, 0
for i in range(4):
        y_draw += C[i]*(x_draw**i)
        sum_x += C[i]*(x**i)
```

$$y\_draw = C[0] + C[1] * (x\_draw^1) + C[2] * (x\_draw^2) + C[3] * (x\_draw^3)$$
$$sum\_x = C[0] + C[1] * (x^1) + C[2] * (x^2) + C[3] * (x^3)$$

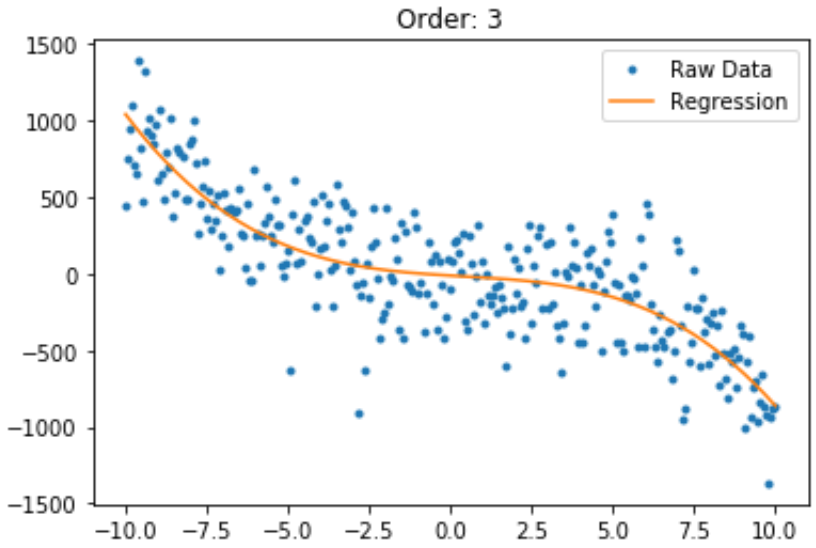
In [24]:
```
sum_square = np.sum((y-sum_x)**2)
```

$$sum\_square = \sum_{i=0}^{n} (y_i - sum\_x_i)^2$$

In [25]:
```
print("\nCoefficient(order-3): ", C)
print("Residual: ", sum_square, "\nOptimal fit error: ", sum_square/np
.size(x))
```

```
Coefficient(order-3):  [-12.52936915 -12.62612795   1.01300895  -0.8
258531 ]
Residual:  19380682.10461033
Optimal fit error:  64602.273682034436
```

In [26]:
```
plt.title("Order: 3")
plt.plot(x, y, '.', label='Raw Data')
plt.plot(x_draw, y_draw, '-', label='Regression')
plt.legend(loc='best')
plt.show()
```




# Degree - 5

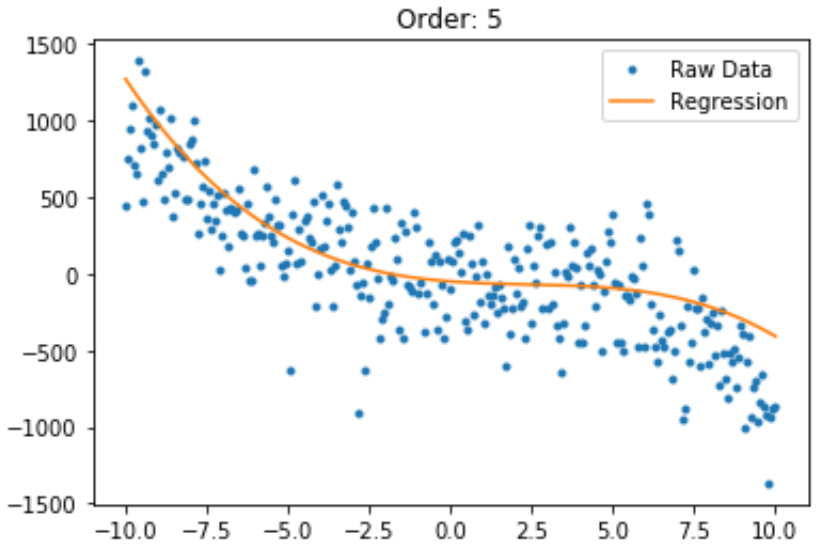
**Degree-5 and Degree-7 are the same as above.**

In [29]:
```
A = np.concatenate((ones_arr_np.T, x_np.T, (x_np**2).T, (x_np**3).T, (
x_np**4).T, (x_np**5).T), axis=1)
C = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y)
x_draw = np.linspace(-10, 10, 10000)
y_draw, sum_x = 0, 0
for i in range(4):
        y_draw += C[i]*(x_draw**i)
        sum_x += C[i]*(x**i)

sum_square = np.sum((y-sum_x)**2)
print("\nCoefficient(order-5): ", C)
print("Residual: ", sum_square, "\nOptimal fit error: ", sum_square/np
.size(x))
```

```
Coefficient(order-5):  [-5.08531140e+01 -1.57128463e+01  4.82011657e
+00 -6.82751700e-01
 -4.41227651e-02 -1.27944031e-03]
Residual:  26150834.583551634
Optimal fit error:  87169.44861183879
```

In [30]:
```
plt.title("Order: 5")
plt.plot(x, y, '.', label='Raw Data')
plt.plot(x_draw, y_draw, '-', label='Regression')
plt.legend(loc='best')
plt.show()
```

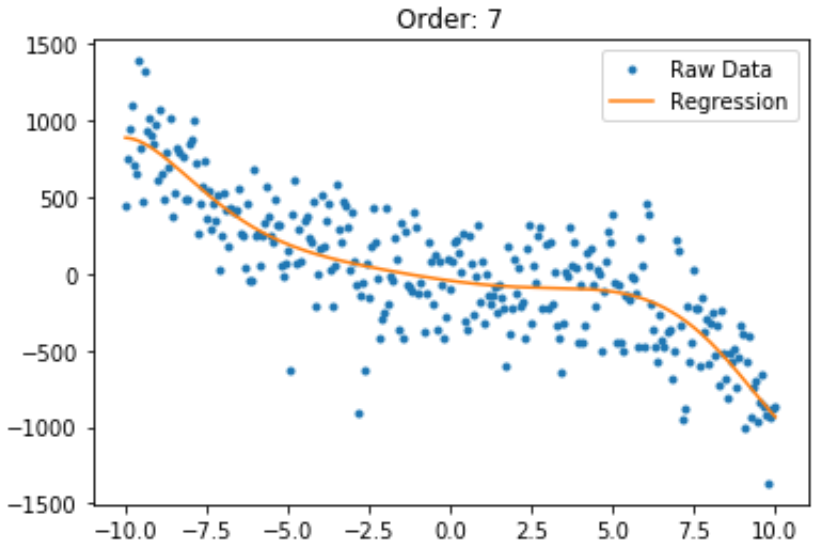



## Degree - 7

In [31]:
```
A = np.concatenate((ones_arr_np.T, x_np.T, (x_np**2).T, (x_np**3).T, (
x_np**4).T, (x_np**5).T, (x_np**6).T, (x_np**7).T), axis=1)
C = np.dot(np.dot(np.linalg.inv(np.dot(A.T, A)), A.T), y)
x_draw = np.linspace(-10, 10, 10000)
y_draw, sum_x = 0, 0
for i in range(8):
        y_draw += C[i]*(x_draw**i)
        sum_x += C[i]*(x**i)
sum_square = np.sum((y-sum_x)**2)
print("\nCoefficient(order-7): ", C)
print("Residual: ", sum_square, "\nOptimal fit error: ", sum_square/np
.size(x))
```

```
Coefficient(order-7):  [-4.61622370e+01 -2.91366769e+01  3.84144730e
+00  5.17578931e-01
 -1.49543134e-02 -2.75155264e-02 -2.12502741e-04  1.61358610e-04]
Residual:  18944163.087041147
Optimal fit error:  63147.210290137155
```

In [33]:
```
plt.title("Order: 7")
plt.plot(x, y, '.', label='Raw Data')
plt.plot(x_draw, y_draw, '-', label='Regression')
plt.legend(loc='best')
plt.show()
```




In [ ]: