In [40]:
```
# import package

import numpy as np
import random
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
from sklearn.metrics import r2_score
```

1. Function generation

To make the code efficiently, here I use functions to prevent similar codes

In [41]:
```
# in the following sections, several scatter plot figures are need
# draw the scatter plot
def scatter_plot(a, b, title):
    # a is the x value, b is the y value
    plt.scatter(a, b, alpha=0.8)
    plt.xlabel("x") # since the labels keep same for the following parts
    plt.ylabel("y")
    plt.title(title)
    plt.show()

# fit the linear model
def linear_fit(a, b):
    # a is the input variable(s), b is the output variable
    model = LinearRegression().fit(a, b)
    return model

# darw the scatter plot and the model line/curve
def scatter_fit_plot(a, b, c, d, new_label, cus_title):
    # a: vector x; b: true y; c: x values for predcting; d: predicted y on c
    plt.scatter(a, b, alpha=0.8, label='Data point')
    plt.plot(c, d, '-', linewidth=1, color='red', label=new_label)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.legend()
    plt.title(cus_title)
    plt.show()

# calculate the r2 statistics
def cal_r2(model, a, b):
    # a: input data; b:true y
    y_pre = model.predict(a)
    return r2_score(b, y_pre)
```

2. Data Generation and Model Fitting

2.1 First Data Generation

In [42]:
```
# create a vector x containing 100 observations (mean 0, variance 1)

# set the random seed to ensure the same experimental results
np.random.seed(42)

mean_x = 0; var_x = 1
```

```
x0 = np.random.normal(mean_x, var_x**0.5, 100)

# create a vector eps containing 100 observations (mean 0, variance 0.25)
mean_eps = 0; var_eps = 0.25
eps = np.random.normal(mean_eps, var_eps**0.5, 100)

# generate a vector y according to the model: y = -0.5 + 0.75x + eps
y0 = -0.5 + 0.75*x0 + eps
```

In [43]:
```
np.var(x0), np.var(y0)
```

Out[43]:
```
(0.8165221946938584, 0.5966616593549943)
```

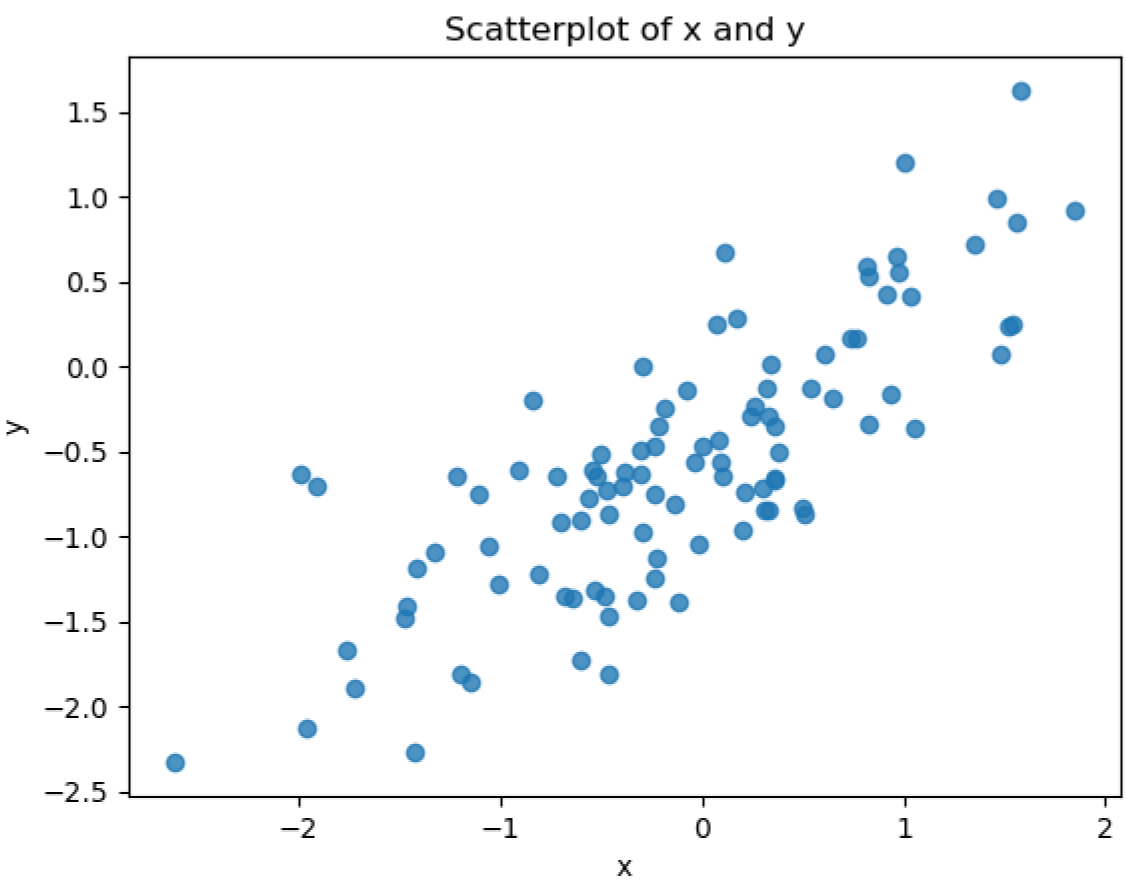
Question a: What is the length of the vector y?

In [44]:
```
print('The length of the vector y is', len(y0))
```

```
The length of the vector y is 100
```

2.2 First Data Visualization

In [45]:
```
scatter_plot(x0, y0, 'Scatterplot of x and y')
```



2.3 Fitting First Linear Regression

In [46]:
```
# transform the data shape to fit the function
x = x0.reshape(-1,1)
y = y0.reshape(-1,1)

# fit the data into the linear regression model
model1 = linear_fit(x, y)
```

(a) How do the estimations of β̂0 and β̂1 compare to β0 and β1 ?

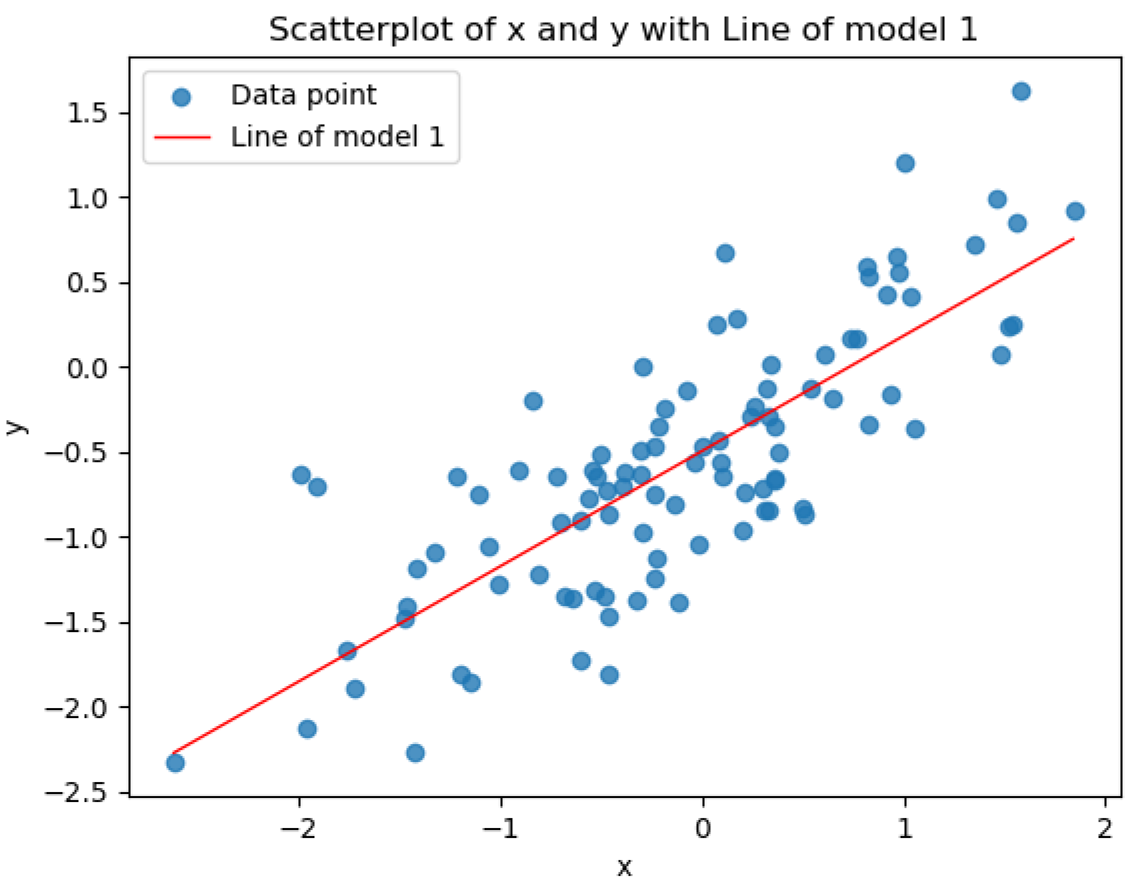
In [47]:
```
# calculate beta_0_hat and beta_1_hat
m1_beta_0_hat = model1.intercept_[0]
m1_beta_1_hat = model1.coef_[0][0]

m1_beta_0_hat, m1_beta_1_hat
## output: (-0.4962860850680164, 0.6783714198642783)
```

Out[47]:
```
(-0.4962860850680164, 0.6783714198642783)
```

(b) Display the least squares line on the scatterplot obtained in Subsection 2.2.

In [48]:
```
# generate x values and predict on x_step to draw the line of model 1
x_step = np.arange(min(x), max(x), 0.02).reshape(-1,1)
y_m1 = model1.predict(x_step)
```

In [49]:
```
# draw the figure
scatter_fit_plot(x, y, x_step, y_m1,
                 'Line of model 1',
                 "Scatterplot of x and y with Line of model 1")
```



(c) Compute R2 statistics

In [50]:
```
m1_r2 = cal_r2(model1, x, y)
m1_r2
## output: 0.6297598193059208
```

Out[50]:
```
0.6297598193059208
```

2.4 Fitting Second Linear Regression

In [51]:
```
## generate x^2 and the input data

import pandas as pd

x_2 = np.square(x0)# x square, the quadratic term

# convert array to dataframe to prepare for the regression
df_quad_x = pd.DataFrame({'x': list(x0), 'x^2': list(x_2)}, columns=['x', 'x^2'])
```

In [52]:
```
# fit the data into the model
model2 = linear_fit(df_quad_x, y)
```

(a) What is the estimated value for β̂2?

In [53]:
```
m2_beta_2_hat = model2.coef_[0][1]

m2_beta_2_hat
## output: 0.09221497437831445
```

Out[53]:
```
0.09221497437831445
```

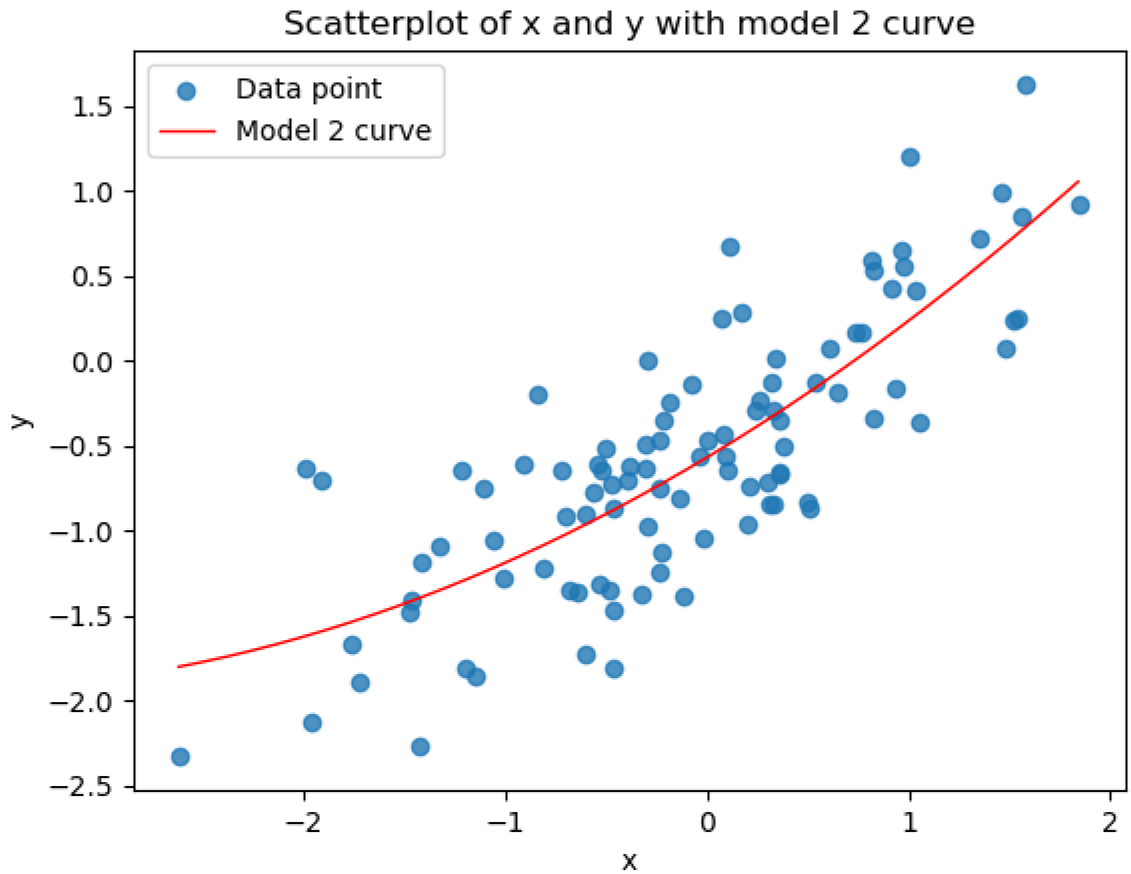
(b) How do the estimations of β̂0 and β̂1 compare to β0 and β1 ?

In [54]:
```
## calculate beta_0_hat and beta_1_hat
m2_beta_0_hat = model2.intercept_[0]
m2_beta_1_hat = model2.coef_[0][0]

m2_beta_0_hat, m2_beta_1_hat
## output: (-0.5690705740231348, 0.7121283510062474)
```

Out[54]:
```
(-0.5690705740231348, 0.7121283510062474)
```

(c) Display the least squares line on the scatterplot obtained in Subsection 2.2.

In [55]:
```
# generate input data to draw the curve
df_x_step = pd.DataFrame({'x': list(x_step), 'x^2': list(np.square(x_step))},
                         columns=['x', 'x^2'])
y_step_m2 = model2.predict(df_x_step)

scatter_fit_plot(x, y, x_step, y_step_m2,
                 'Model 2 curve',
                 "Scatterplot of x and y with model 2 curve")
```



(d) Compute R2 statistics.

In [56]:
```
m2_r2 = cal_r2(model2, df_quad_x, y)
m2_r2
## output: 0.6469951045504286
```

Out[56]:
```
0.6469951045504286
```

(e) Is there evidence that the quadratic term improves the model fit? Explain your answer.

In [57]:
```
# from the perspective of increasement of r2
print('r2 of model 1:', m1_r2)
print('r2 of model 2:', m2_r2)
print('Increasement of r^2:', m2_r2 - m1_r2)
```

```
r2 of model 1: 0.6297598193059208
r2 of model 2: 0.6469951045504286
Increasement of r^2: 0.017235285244507792
```

In [58]:
```
# from the perspective of decreasement of mse
from sklearn.metrics import mean_squared_error

mse_m1 = mean_squared_error(y_true=y, y_pred=model1.predict(x))
mse_m2 = mean_squared_error(y_true=y, y_pred=model2.predict(df_quad_x))

print('Decreasement of MSE:', mse_m1 - mse_m2)
```

```
Decreasement of MSE: 0.010283633893444638
```

In [65]:
```
# from the perspective of predction accuracy
from sklearn.metrics import mean_absolute_error
import math
```

```
def mae_2model_compare(x_m1, x_m2, y_value, model1, model2):
    # set the cut-line of the train and test data
    cut_number = math.ceil(len(x_m1)*0.7)

    # prepare the train and test data for model 1 and model 2
    x_train_m1 = x_m1[:cut_number]
    x_test_m1 = x_m1[cut_number:]

    x_train_m2 = x_m2[:cut_number]
    x_test_m2 = x_m2[cut_number:]

    y_train = y_value[:cut_number]
    y_test = y_value[cut_number:]

    # train the model and obtain the MAE
    m1_train = linear_fit(x_train_m1.reshape(-1,1), y_train.reshape(-1,1))
    y_m1_train_pred = m1_train.predict(x_test_m1.reshape(-1,1))
    test_m1_mae = mean_absolute_error(y_test, y_m1_train_pred)
    print('MAE of', model1, test_m1_mae)

    m2_train = linear_fit(x_train_m2, y_train.reshape(-1,1))
    y_m2_train_pred = m2_train.predict(x_test_m2)
    test_m2_mae = mean_absolute_error(y_test, y_m2_train_pred)
    print('MAE of', model2, test_m2_mae)

    # compare the MAE of the two models on the test data
    print('Çhange of MAE from', str(model1), 'to', str(model2), ':', test_m2_mae - test_

mae_2model_compare(x_m1=x0, x_m2=df_quad_x, y_value=y0,
                   model1='model 1', model2='model 2')
```

```
MAE of model 1 0.32898662296457126
MAE of model 2 0.3359233106987609
Çhange of MAE from model 1 to model 2 : 0.006936687734189650 5
```

2.5 Second Data Generation

In [66]:

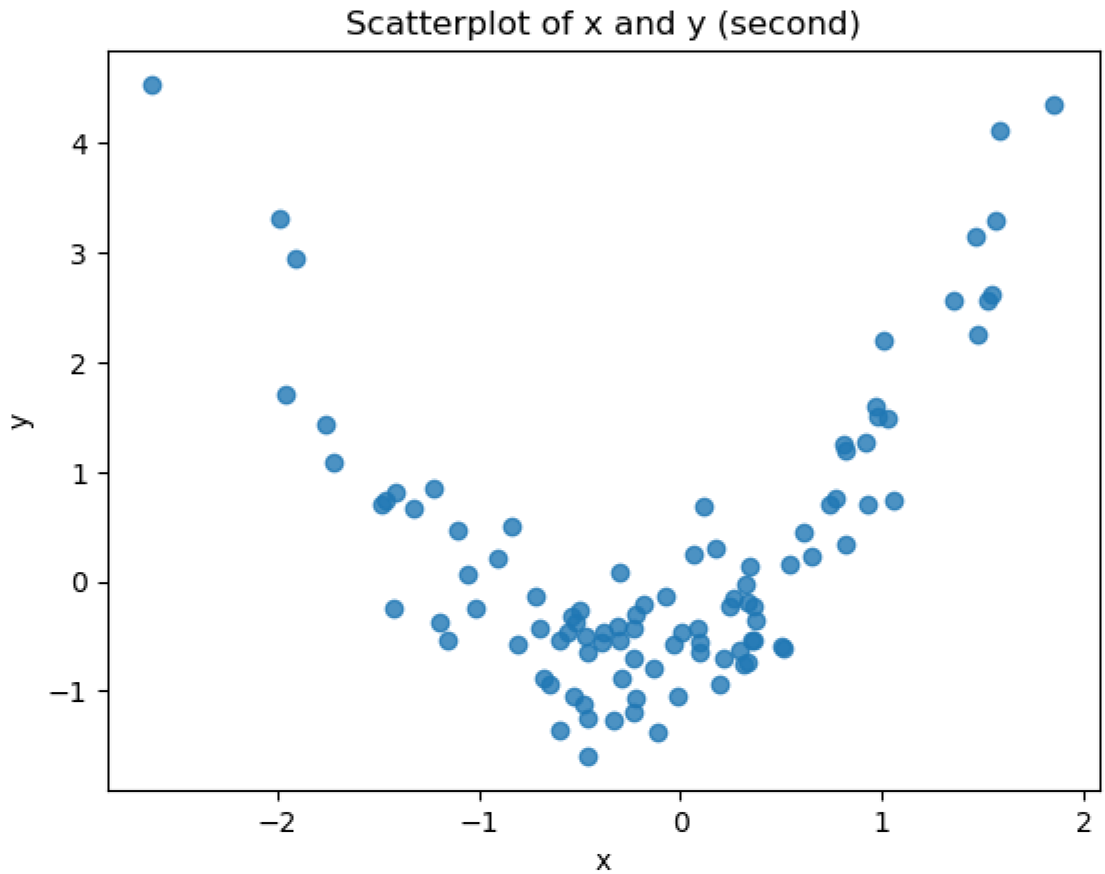
```
# Using x and eps, generate a vector y according to the model:
# y = -0.5 + 0.75x + x2 + eps

y = -0.5 + 0.75 * x0 + x_2 + eps
```

2.6 Second Data Visualization

Create a new scatterplot displaying the relationship between x and y. Comment on what you observe.

In [67]:

```
scatter_plot(x0, y, "Scatterplot of x and y (second)")
```



2.7 Fitting Third Linear Regression

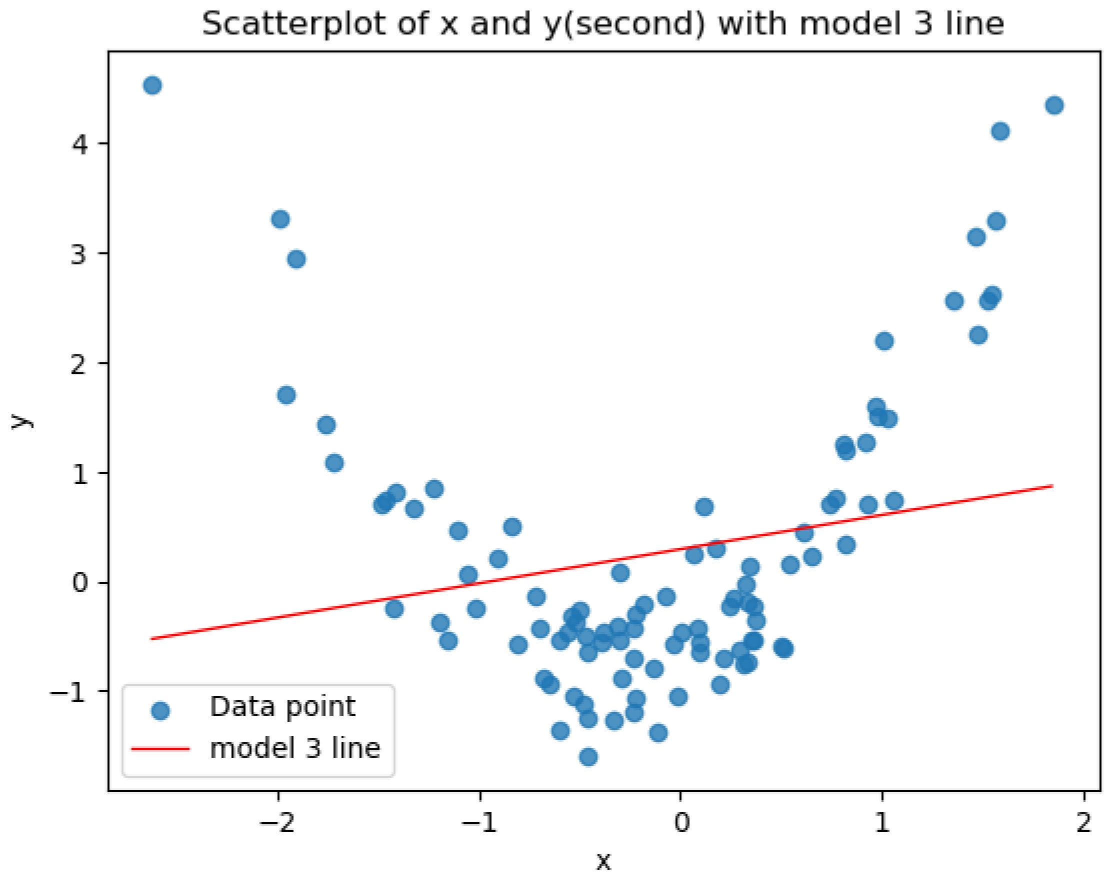
(a) How do the estimations of β̂0 and β̂1 compare to β0 and β1 ?

In [68]:
```
model3 = linear_fit(x, y.reshape(-1,1))

m3_beta_0_hat = model3.intercept_[0]
m3_beta_1_hat = model3.coef_[0][0]

m3_beta_0_hat, m3_beta_1_hat
## output: (0.29300534450521215, 0.31230363781968207)
```

Out[68]:
```
(0.29300534450521215, 0.31230363781968207)
```

(b) Display the least squares line on the scatterplot obtained in Subsection 2.6.

In [69]:
```
m3_y = model3.predict(x_step)

scatter_fit_plot(x, y, x_step, m3_y,
                 'model 3 line',
                 "Scatterplot of x and y(second) with model 3 line")
```



(c) Compute R2 statistics.

In [70]:
```
m3_r2 = cal_r2(model3, x, y)
m3_r2
## output: 0.045956423052825435
```

Out[70]:
```
0.045956423052825435
```

2.8 Fitting Fourth Linear Regression

Now fit a polynomial regression model that predicts y using x and x2. Comment on the model obtained:

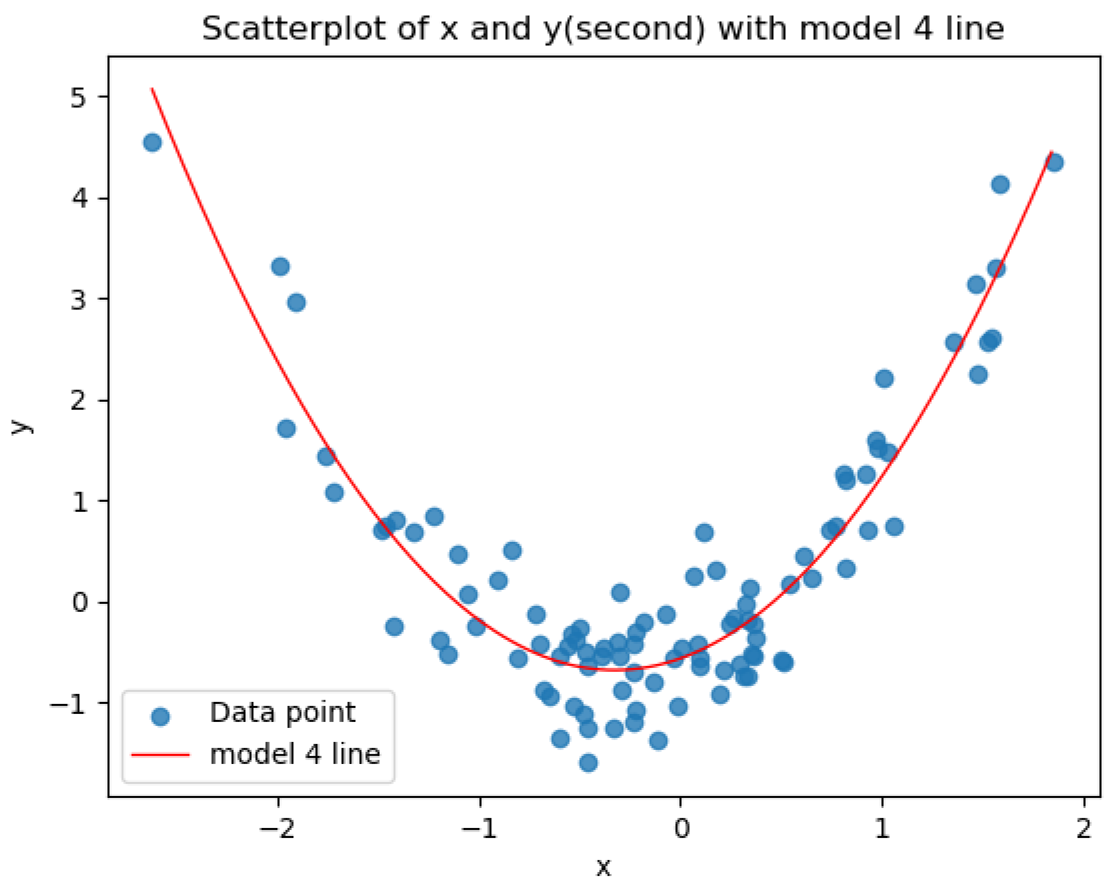
(a) How do the estimations of β^0, β^1, amd β^2 compare to β0, β1, and β2 ?

In [71]:
```
model4 = linear_fit(df_quad_x, y.reshape(-1,1))

m4_beta_0_hat = model4.intercept_[0]
m4_beta_1_hat = model4.coef_[0][0]
m4_beta_2_hat = model4.coef_[0][1]

m4_beta_0_hat, m4_beta_1_hat, m4_beta_2_hat
## output: (-0.5690705740231352, 0.7121283510062474, 1.092214974378315)
```

Out[71]:
```
(-0.5690705740231352, 0.7121283510062474, 1.092214974378315)
```

(b) Display the least squares line on the scatterplot obtained in Subsection 2.6.

In [72]:
```python
m4_y = model4.predict(df_x_step)

scatter_fit_plot(x, y, x_step, m4_y,
                 'model 4 line',
                 "Scatterplot of x and y(second) with model 4 line")
```



(c) Compute R2 statistics.

In [73]:
```python
m4_r2 = cal_r2(model4, df_quad_x, y)
m4_r2
## output: 0.8784561474099986
```

Out[73]:
```
0.8784561474099986
```

(d) Is there evidence that the quadratic term improves the model fit? Explain your answer.

In [74]:
```python
# from the perspective of increasement of r2
print('Increasement of r^2 from model 3 to model 4:', m4_r2 - m3_r2)
```

```
Increasement of r^2 from model 3 to model 4: 0.8324997243571731
```

In [75]:
```python
# from the perspective of decreasement of mse
from sklearn.metrics import mean_squared_error

mse_m3 = mean_squared_error(y_true=y, y_pred=model3.predict(x))
mse_m4 = mean_squared_error(y_true=y, y_pred=model4.predict(df_quad_x))

print('Decreasement of MSE from model 3 to model 4:', mse_m3 - mse_m4)
```

```
Decreasement of MSE from model 3 to model 4: 1.4426466116302548
```

In [76]:
```python
# from the perspective of predction accuracy
mae_2model_compare(x_m1=x0, x_m2=df_quad_x, y_value=y,
```

```
                    model1='model 3', model2='model 4')
```

```
MAE of model 3 1.1327708525281477
MAE of model 4 0.3359233106987608
Çhange of MAE from model 3 to model 4 : -0.7968475418293869
```

In [77]:
```
print('|Change of MAE|/range of x:', abs(-0.7968475418293869) / (max(x0)-min(x0)))
```

```
|Change of MAE|/range of x: 0.17818501613373322
```