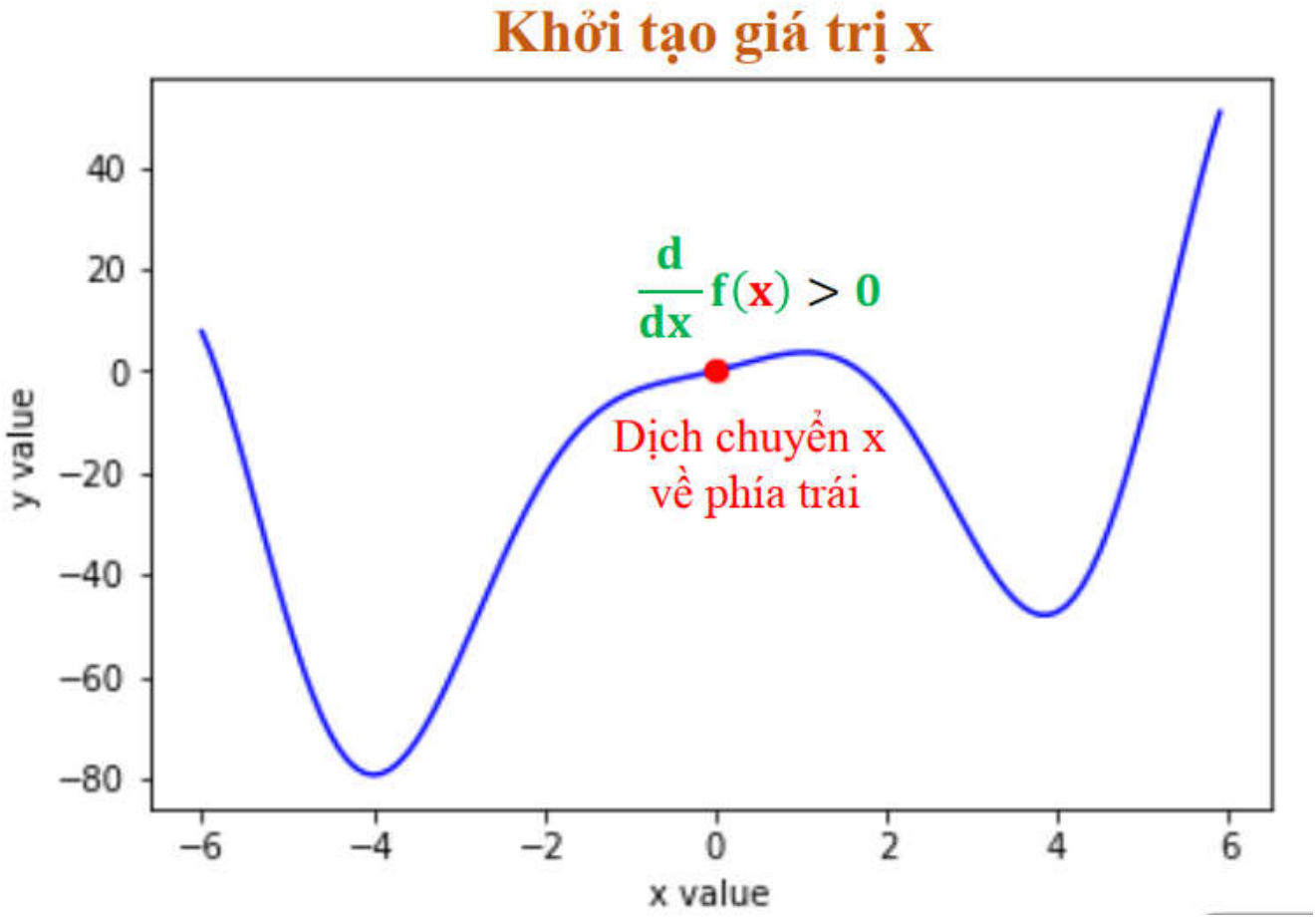
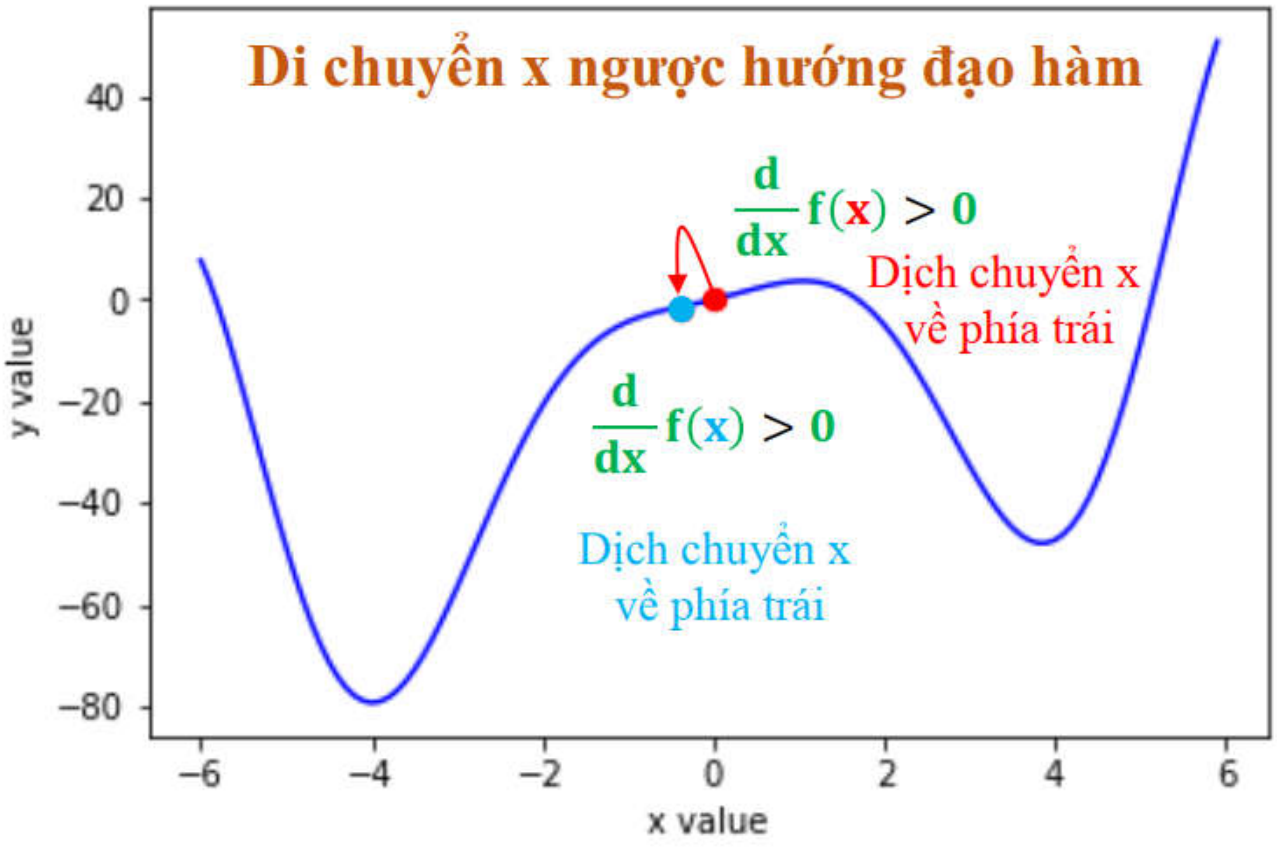
Gradient descent

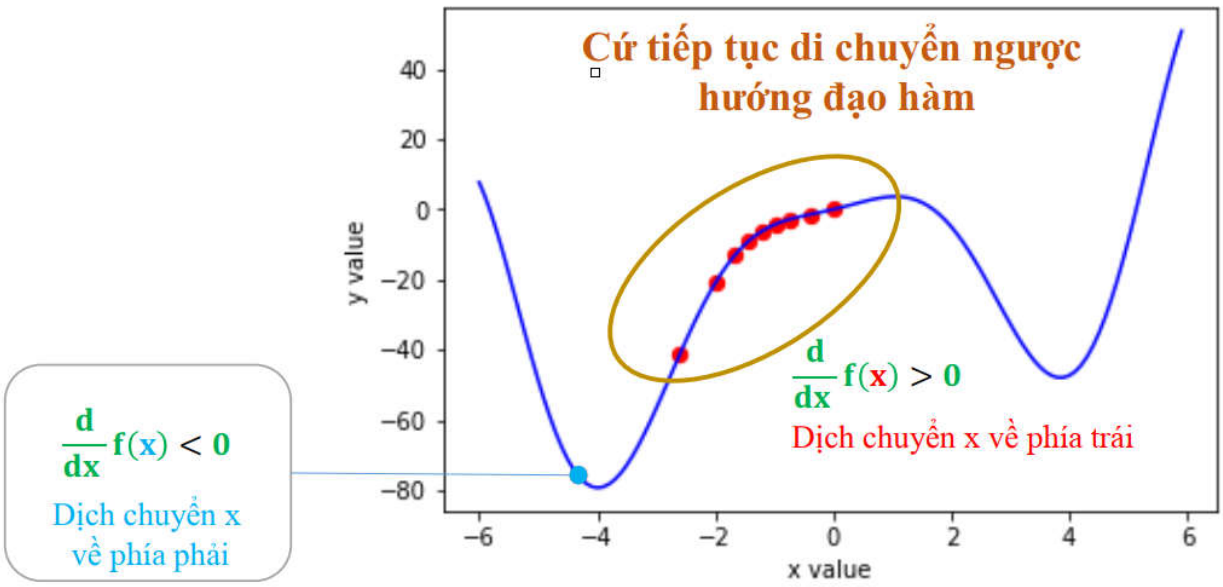
1. Khởi tạo giá trị x = x0 tùy ý



2. Gán x = x - a* f'(x) (a là hằng số không âm ví dụ a = 0.01)



3. Tính lại f(x): Nếu f(x) đủ nhỏ thì dừng lại, ngược lại tiếp tục bước 2



giá trị dự đoán: O = w1*x1* + *w*2x2 + ... wn*xn + b

viết lại: O = W*X + b

trong đó W là ma trân trọng số, X là vector đặc trưng

loss funtion L(w, b) = $(o - y)$**2

Tính đạo hàm

$$\frac{\partial L}{\partial w_j} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial w_j} = 2x_j(o - y)$$

$$\frac{\partial L}{\partial b} = \frac{\partial L}{\partial o}\frac{\partial o}{\partial w_j} = 2(o - y)$$

mục đính là đi tìm giá trị nhỏ nhất của hàm loss funtion

khi đó ta cập nhận trọng số W bằng công thức trên

w = w - a* f'(w)

với f'(w) là đạo hàm riêng theo w

Cập nhật tham số

$$w_j = w_j - \eta L'_{w_j}$$

$$b = b - \eta L'_b$$

η is learning rate

ví dụ

In [87]:

```
#data
import numpy as np
X = np.array([6.7, 4.6, 3.5, 5.5])
y = np.array([9.1, 5.9, 4.6, 6.7]) #label
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)
print(X)
print(y)
```

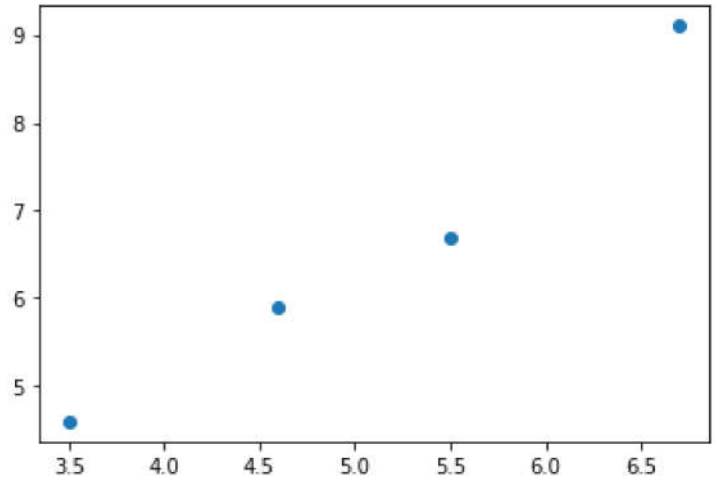
```
[[6.7]
 [4.6]
 [3.5]
 [5.5]]
[[9.1]
 [5.9]
 [4.6]
 [6.7]]
```

In [88]:

```
from matplotlib import pyplot as plt
plt.scatter(X, y)
```

Out[88]:

```
<matplotlib.collections.PathCollection at 0xb771f0e80>
```



O = w*x + b

loss funtion: Loss = (O - y)2 = (w*x + b - y)2

-> Loss'(w) = 2(w*x* + *b)*x

-> Loss'(b) = 2(w*x + b)

cập nhật hệ số: x = x - a* f'(x)

In [89]:

```
X = np.hstack((np.ones((4, 1)), X))
print(X)
w = np.array([0.,1.]).reshape(-1,1)
print(w)
```

```
[[1.  6.7]
 [1.  4.6]
 [1.  3.5]
 [1.  5.5]]
[[0.]
 [1.]]
```

In [90]:

```
learning_rate = 0.01
#cập nhật hệ số 100 lần
numOfIteration = 100
cost = np.zeros((numOfIteration,1)) #lưu trữ dữ liệu sau mỗi lần cập nhật

for i in range(1, numOfIteration):
    r = np.dot(X, w) - y
    cost[i] = 0.5*np.sum(r*r)
    w[0] -= learning_rate*np.sum(r)
    # correct the shape dimension
    w[1] -= learning_rate*np.sum(np.multiply(r, X[:,1].reshape(-1,1)))
    print(cost[i])
```

```
[5.05]
[0.26049609]
[0.18732943]
[0.18613523]
[0.18603965]
[0.18596114]
[0.18588319]
[0.18580555]
[0.18572822]
[0.18565119]
[0.18557446]
[0.18549804]
[0.18542191]
[0.18534609]
[0.18527056]
[0.18519533]
[0.1851204]
[0.18504576]
[0.18497141]
[0.18489736]
[0.1848236]
[0.18475012]
[0.18467694]
[0.18460404]
[0.18453143]
[0.18445911]
[0.18438707]
[0.18431532]
[0.18424384]
[0.18417265]
[0.18410174]
[0.1840311]
[0.18396075]
[0.18389067]
[0.18382086]
[0.18375133]
[0.18368208]
[0.18361309]
[0.18354438]
[0.18347594]
[0.18340776]
[0.18333986]
[0.18327222]
[0.18320485]
[0.18313774]
```

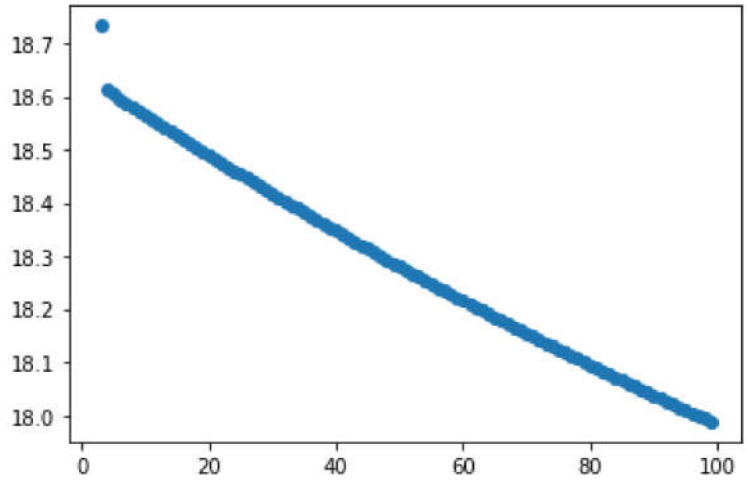
```
[0.18307089]
[0.18300431]
[0.18293799]
[0.18287194]
[0.18280614]
[0.1827406]
[0.18267532]
[0.18261029]
[0.18254552]
[0.182481]
[0.18241674]
[0.18235273]
[0.18228898]
[0.18222547]
[0.18216221]
[0.18209921]
[0.18203645]
[0.18197393]
[0.18191166]
[0.18184964]
[0.18178786]
[0.18172633]
[0.18166503]
[0.18160398]
[0.18154317]
[0.18148259]
[0.18142226]
[0.18136216]
[0.18130229]
[0.18124267]
[0.18118328]
[0.18112412]
[0.18106519]
[0.1810065]
[0.18094803]
[0.1808898]
[0.18083179]
[0.18077402]
[0.18071647]
[0.18065914]
[0.18060204]
[0.18054517]
[0.18048852]
[0.18043209]
[0.18037589]
[0.1803199]
[0.18026414]
[0.18020859]
[0.18015327]
[0.18009816]
[0.18004327]
[0.17998859]
[0.17993413]
[0.17987988]
```

In [105]:

```
# biểu đồ làm loss đang giảm
plt.scatter(range(3, 100), 100*cost[3:])
```

Out[105]:

```
<matplotlib.collections.PathCollection at 0xb78512f60>
```



In [92]:

```
w.reshape(2)
```

Out[92]:

```
array([-0.02475688,  1.3039986 ])
```

In [93]:

```
x = np.array(range(1,10))
y_predict = x*w[1] + w[0]
print(x, y_predict)
```

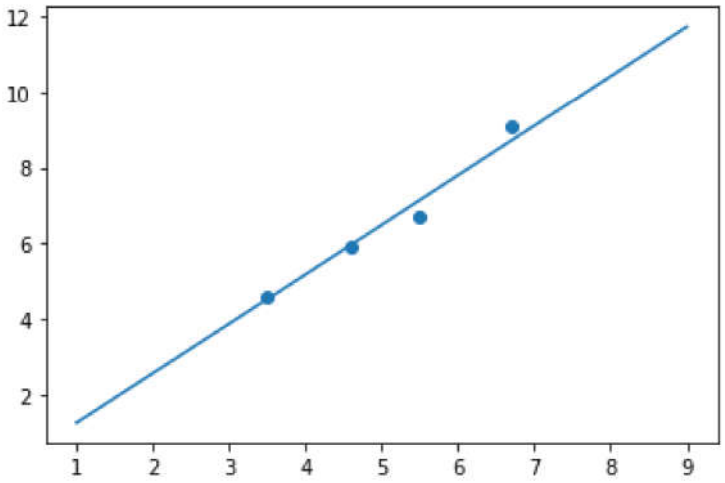
```
[1 2 3 4 5 6 7 8 9] [ 1.27924172  2.58324033  3.88723893  5.19123753  6.4952
3614  7.79923474
  9.10323335 10.40723195 11.71123056]
```

In [95]:

```
X = np.array([6.7, 4.6, 3.5, 5.5])
y = np.array([9.1, 5.9, 4.6, 6.7])
plt.scatter(X, y)
plt.plot(x,y_predict)
```

Out[95]:

```
[<matplotlib.lines.Line2D at 0xb77272cc0>]
```



In [106]:

```
x = 10
y_predict = x*w[1] + w[0]
print(y_predict)
```

```
[13.01522916]
```

In [ ]: