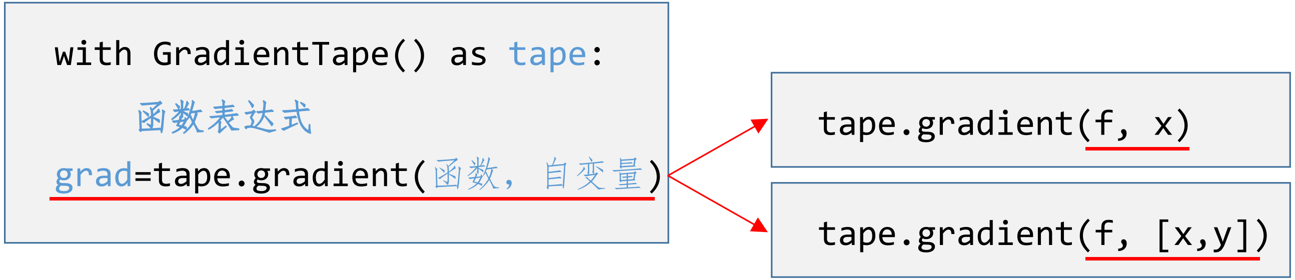
10.4 TensorFlow实现梯度下降法

中国大学MOOC

课程回顾

- **可训练**变量
  - □ **Variable**对象
  - □ 自动记录梯度信息
  - □ 由算法自动优化
- **GradientTape**——自动求导



```
with GradientTape() as tape:
     函数表达式
grad=tape.gradient(函数，自变量)
```

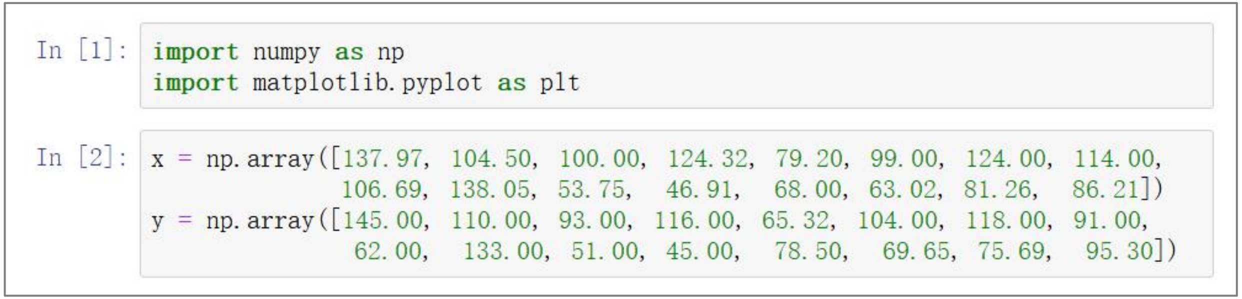
```
tape.gradient(f, x)
```

```
tape.gradient(f, [x,y])
```

10.4 TensorFlow实现梯度下降法

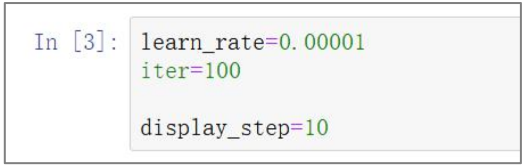
■ NumPy实现一元线性回归

加载数据



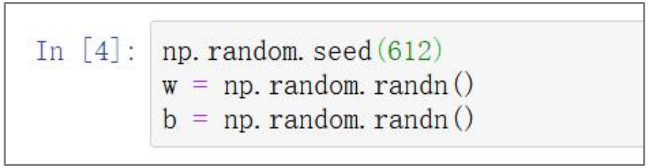
```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

In [2]: x = np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
                      106.69, 138.05, 53.75,  46.91,  68.00, 63.02, 81.26,  86.21])
        y = np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
                       62.00,  133.00, 51.00, 45.00,  78.50,  69.65, 75.69,  95.30])
```

设置超参数



```
In [3]: learn_rate=0.00001
        iter=100

        display_step=10
```

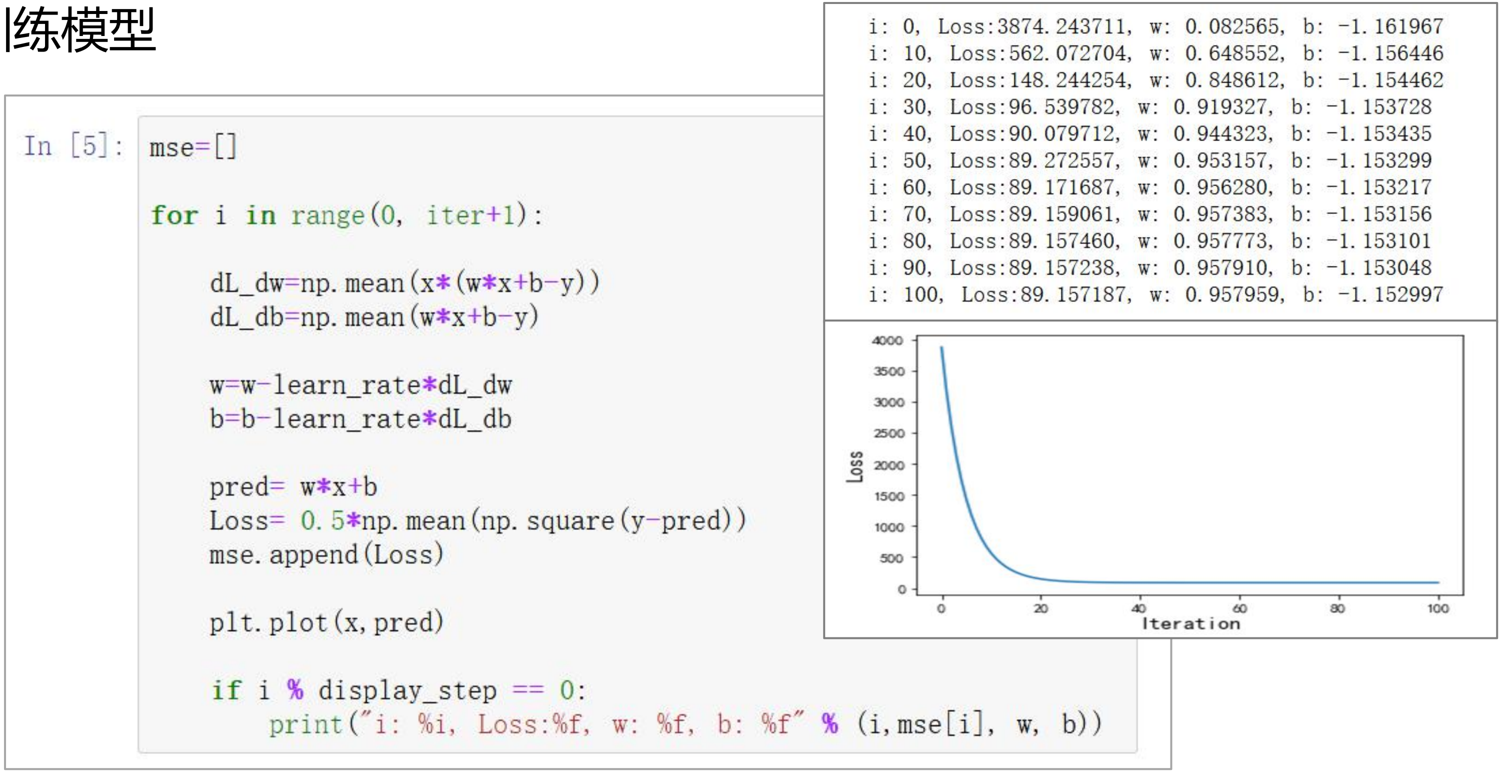
设置模型参数初值



```
In [4]: np.random.seed(612)
        w = np.random.randn()
        b = np.random.randn()
```

10.4 TensorFlow实现梯度下降法

训练模型

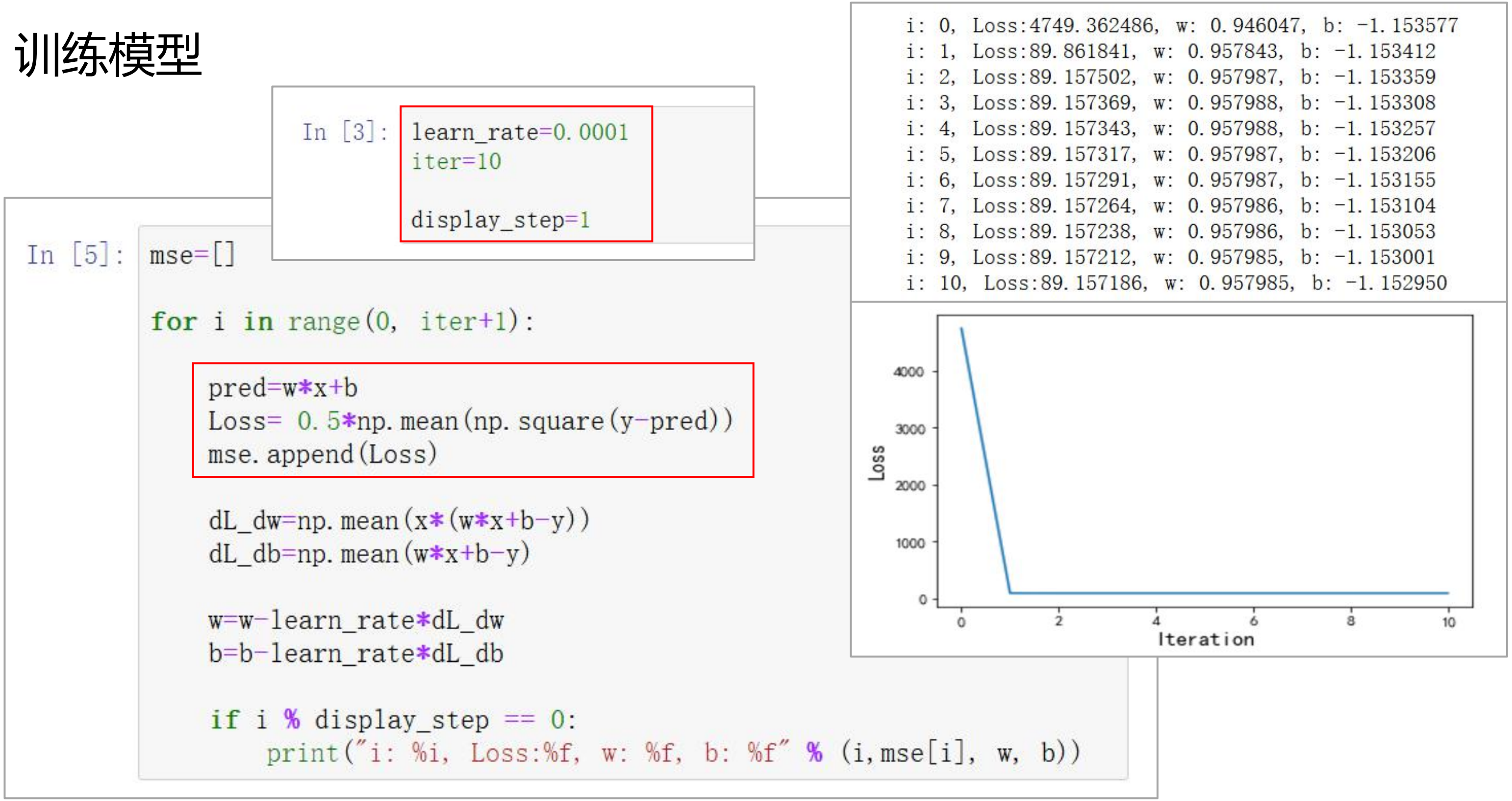


```
In [5]: mse=[]

        for i in range(0, iter+1):

            dL_dw=np.mean(x*(w*x+b-y))
            dL_db=np.mean(w*x+b-y)

            w=w-learn_rate*dL_dw
            b=b-learn_rate*dL_db

            pred= w*x+b
            Loss= 0.5*np.mean(np.square(y-pred))
            mse.append(Loss)

            plt.plot(x,pred)

            if i % display_step == 0:
                print("i: %i, Loss:%f, w: %f, b: %f" % (i,mse[i], w, b))
```

```
i: 0, Loss:3874.243711, w: 0.082565, b: -1.161967
i: 10, Loss:562.072704, w: 0.648552, b: -1.156446
i: 20, Loss:148.244254, w: 0.848612, b: -1.154462
i: 30, Loss:96.539782, w: 0.919327, b: -1.153728
i: 40, Loss:90.079712, w: 0.944323, b: -1.153435
i: 50, Loss:89.272557, w: 0.953157, b: -1.153299
i: 60, Loss:89.171687, w: 0.956280, b: -1.153217
i: 70, Loss:89.159061, w: 0.957383, b: -1.153156
i: 80, Loss:89.157460, w: 0.957773, b: -1.153101
i: 90, Loss:89.157238, w: 0.957910, b: -1.153048
i: 100, Loss:89.157187, w: 0.957959, b: -1.152997
```

10.4 TensorFlow实现梯度下降法

训练模型



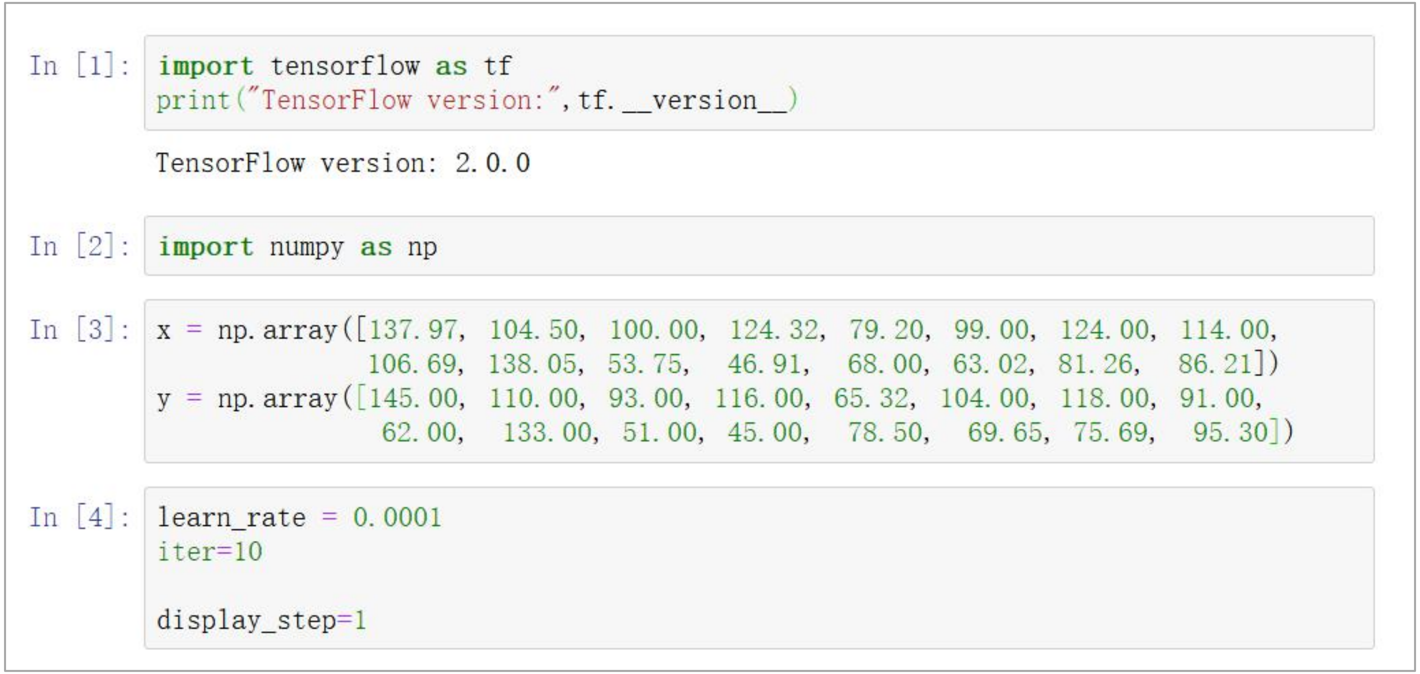
```
In [3]: learn_rate=0. 0001
        iter=10

        display_step=1
```

```
In [5]: mse=[]

        for i in range(0, iter+1):

            pred=w*x+b
            Loss= 0. 5*np. mean(np. square(y-pred))
            mse. append(Loss)

            dL_dw=np. mean(x*(w*x+b-y))
            dL_db=np. mean(w*x+b-y)

            w=w-learn_rate*dL_dw
            b=b-learn_rate*dL_db

            if i % display_step == 0:
                print(”i: %i, Loss:%f, w: %f, b: %f” % (i,mse[i], w, b))
```

```
i: 0, Loss:4749. 362486, w: 0. 946047, b: -1. 153577
i: 1, Loss:89. 861841, w: 0. 957843, b: -1. 153412
i: 2, Loss:89. 157502, w: 0. 957987, b: -1. 153359
i: 3, Loss:89. 157369, w: 0. 957988, b: -1. 153308
i: 4, Loss:89. 157343, w: 0. 957988, b: -1. 153257
i: 5, Loss:89. 157317, w: 0. 957987, b: -1. 153206
i: 6, Loss:89. 157291, w: 0. 957987, b: -1. 153155
i: 7, Loss:89. 157264, w: 0. 957986, b: -1. 153104
i: 8, Loss:89. 157238, w: 0. 957986, b: -1. 153053
i: 9, Loss:89. 157212, w: 0. 957985, b: -1. 153001
i: 10, Loss:89. 157186, w: 0. 957985, b: -1. 152950
```

10.4 TensorFlow实现梯度下降法

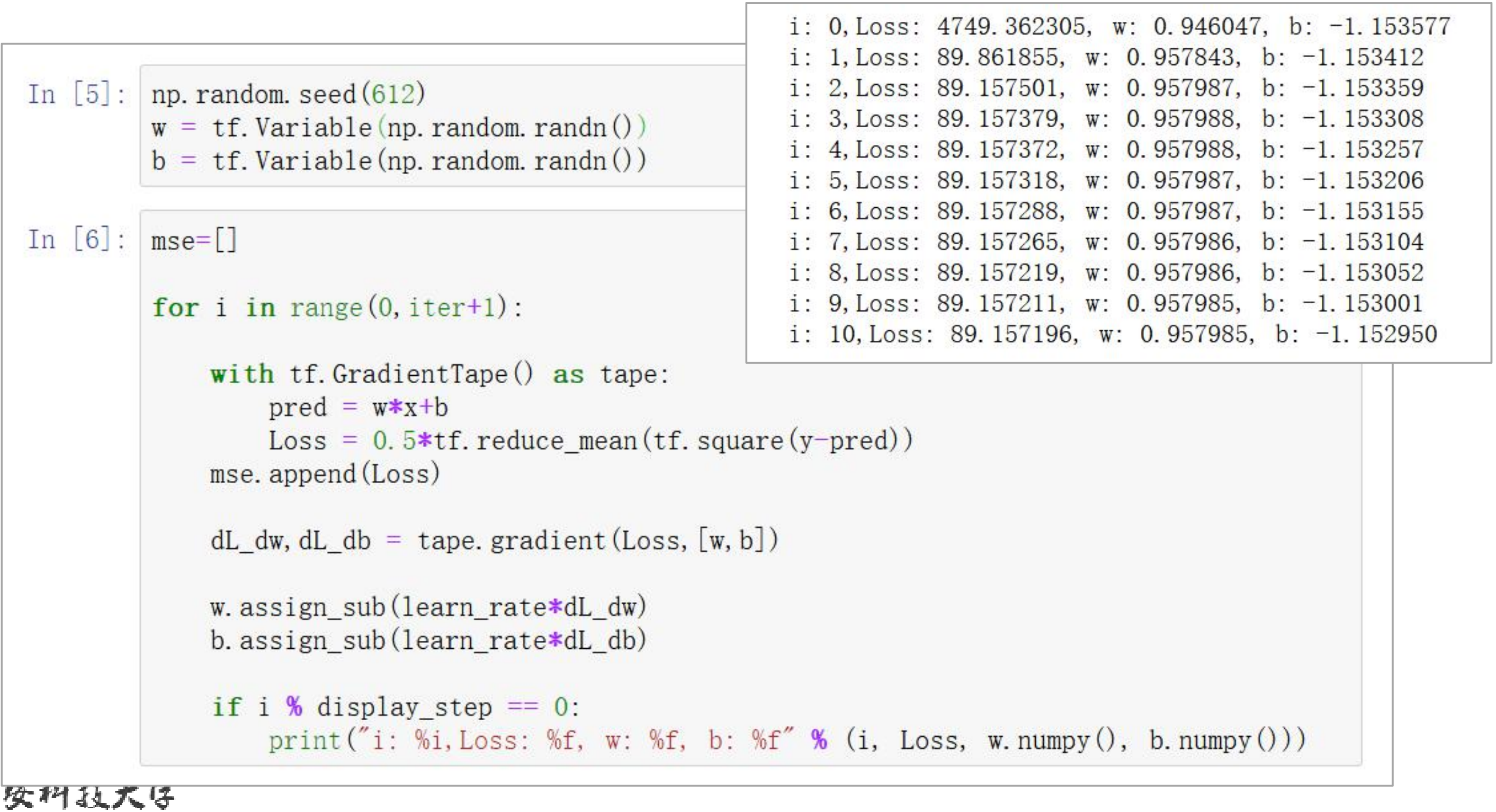
TensorFlow实现一元线性回归



```
In [1]: import tensorflow as tf
        print("TensorFlow version:",tf.__version__)

        TensorFlow version: 2.0.0

In [2]: import numpy as np

In [3]: x = np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
                      106.69, 138.05, 53.75,  46.91,  68.00, 63.02, 81.26,  86.21])
        y = np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
                       62.00,  133.00, 51.00, 45.00,  78.50,  69.65, 75.69,  95.30])

In [4]: learn_rate = 0.0001
        iter=10

        display_step=1
```

西安科技大学
计算机科学与技术学院

10.4 TensorFlow实现梯度下降法



```
In [5]: np.random.seed(612)
        w = tf.Variable(np.random.randn())
        b = tf.Variable(np.random.randn())
```

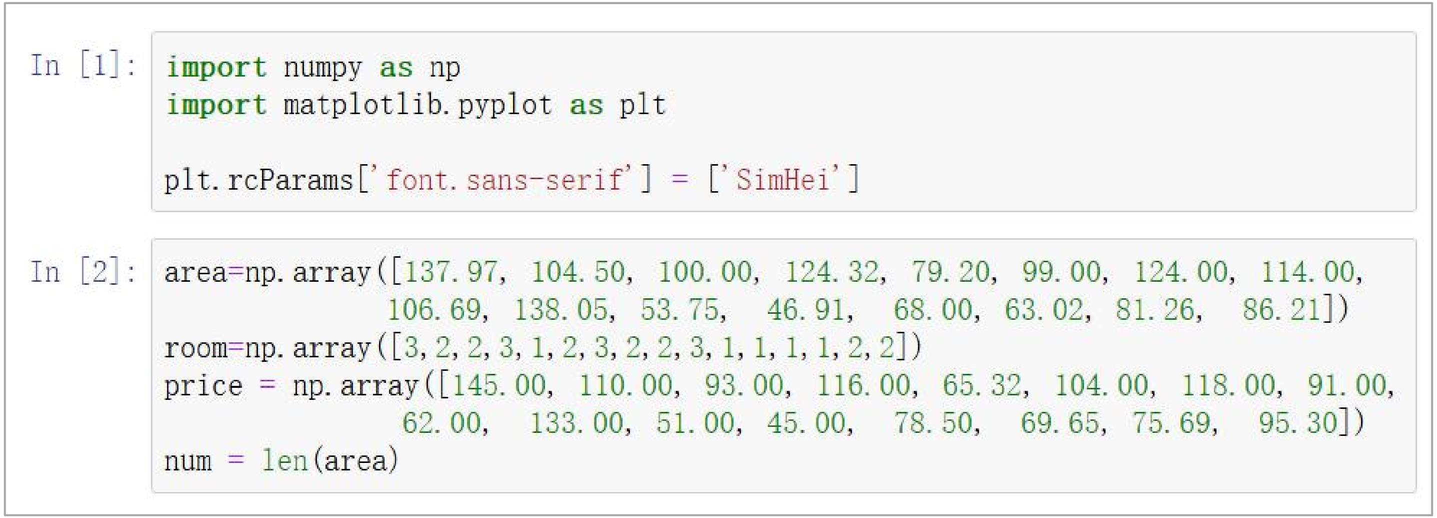
```
In [6]: mse=[]

        for i in range(0,iter+1):

            with tf.GradientTape() as tape:
                pred = w*x+b
                Loss = 0.5*tf.reduce_mean(tf.square(y-pred))
            mse.append(Loss)

            dL_dw,dL_db = tape.gradient(Loss,[w,b])

            w.assign_sub(learn_rate*dL_dw)
            b.assign_sub(learn_rate*dL_db)

            if i % display_step == 0:
                print("i: %i,Loss: %f, w: %f, b: %f" % (i, Loss, w.numpy(), b.numpy()))
```

```
i: 0,Loss: 4749.362305, w: 0.946047, b: -1.153577
i: 1,Loss: 89.861855, w: 0.957843, b: -1.153412
i: 2,Loss: 89.157501, w: 0.957987, b: -1.153359
i: 3,Loss: 89.157379, w: 0.957988, b: -1.153308
i: 4,Loss: 89.157372, w: 0.957988, b: -1.153257
i: 5,Loss: 89.157318, w: 0.957987, b: -1.153206
i: 6,Loss: 89.157288, w: 0.957987, b: -1.153155
i: 7,Loss: 89.157265, w: 0.957986, b: -1.153104
i: 8,Loss: 89.157219, w: 0.957986, b: -1.153052
i: 9,Loss: 89.157211, w: 0.957985, b: -1.153001
i: 10,Loss: 89.157196, w: 0.957985, b: -1.152950
```

10.4 TensorFlow实现梯度下降法

NumPy实现多元线性回归

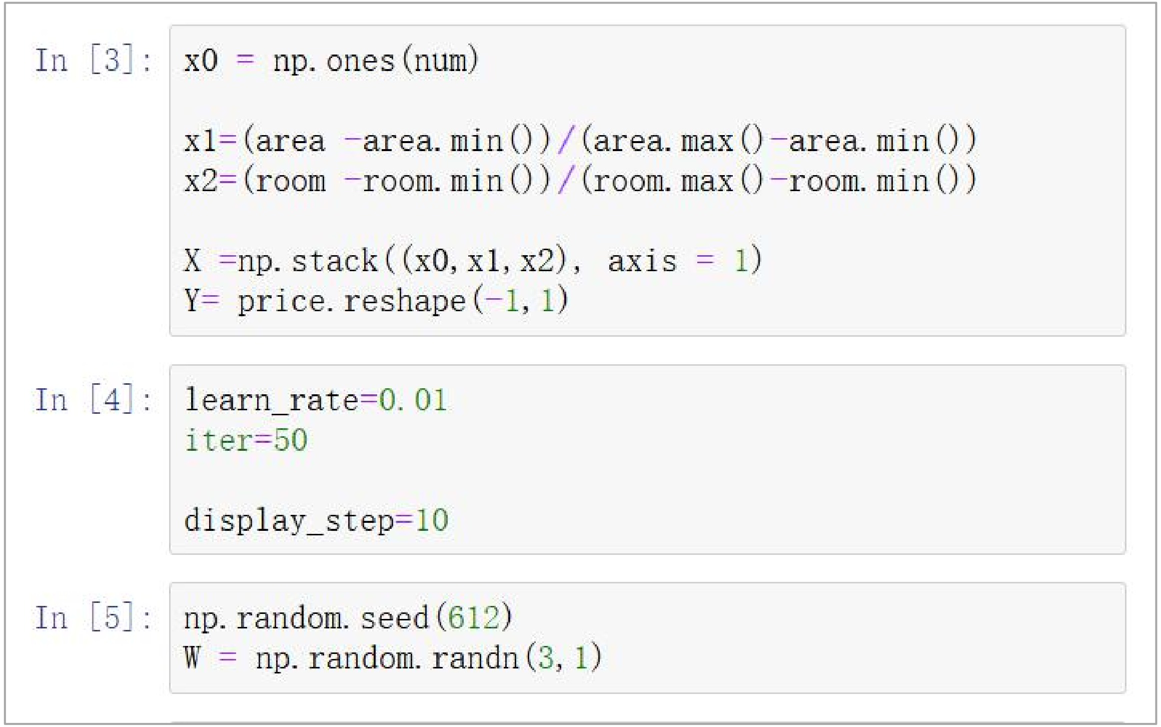
加载样本数据



```
In [1]: import numpy as np
        import matplotlib.pyplot as plt

        plt.rcParams['font.sans-serif'] = ['SimHei']
```

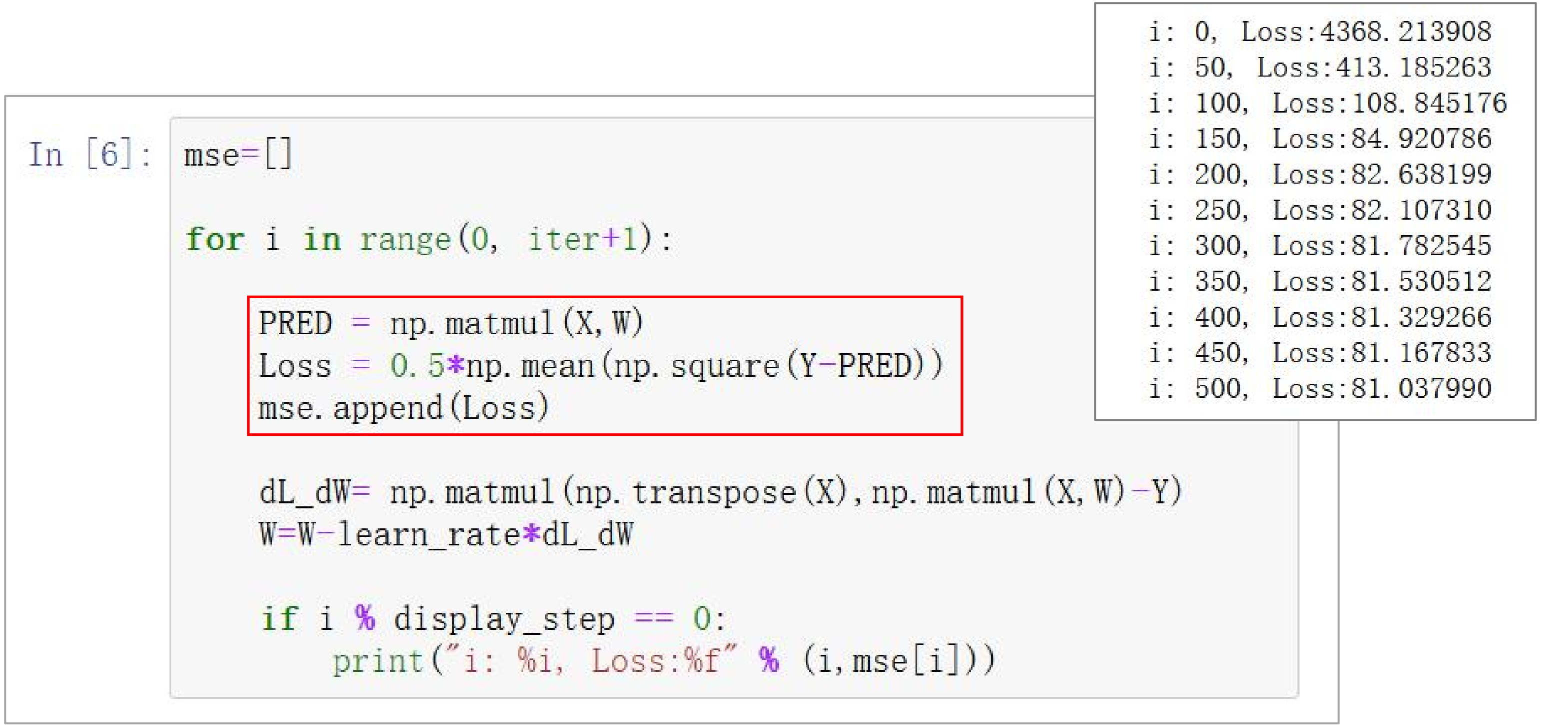
```
In [2]: area=np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
                       106.69, 138.05, 53.75, 46.91, 68.00, 63.02, 81.26, 86.21])
        room=np.array([3,2,2,3,1,2,3,2,2,3,1,1,1,1,2,2])
        price = np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
                          62.00, 133.00, 51.00, 45.00, 78.50, 69.65, 75.69, 95.30])
        num = len(area)
```

10.4 TensorFlow实现梯度下降法



```
In [3]: x0 = np.ones(num)

        x1=(area -area.min())/(area.max()-area.min())
        x2=(room -room.min())/(room.max()-room.min())

        X =np.stack((x0,x1,x2), axis = 1)
        Y= price.reshape(-1,1)

In [4]: learn_rate=0.01
        iter=50

        display_step=10

In [5]: np.random.seed(612)
        W = np.random.randn(3,1)
```

10.4 TensorFlow实现梯度下降法

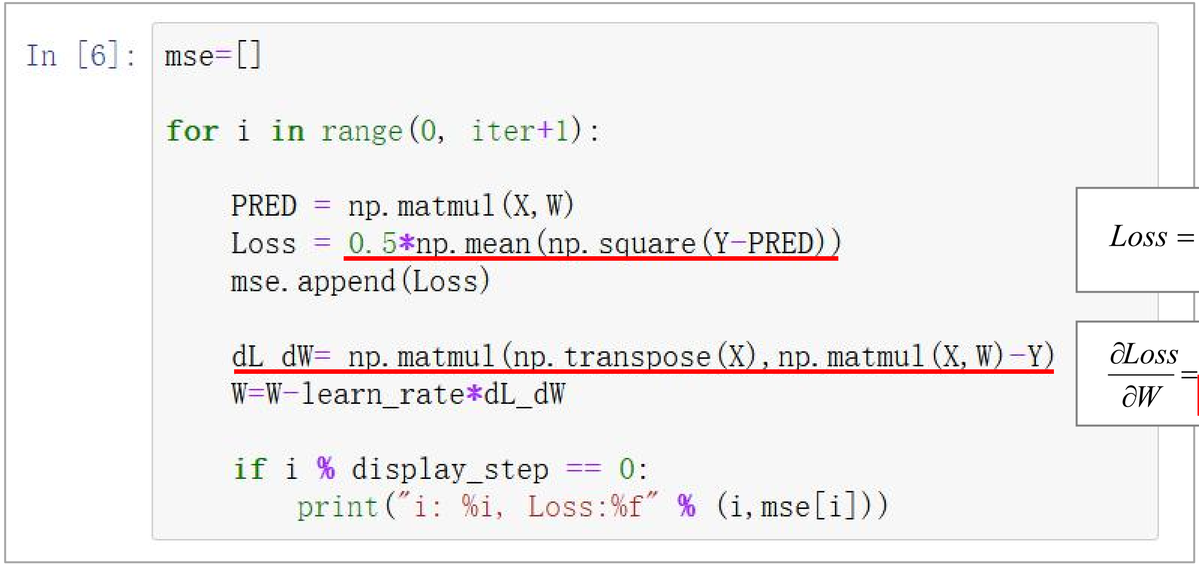


```
In [6]: mse=[]

        for i in range(0, iter+1):

            PRED = np.matmul(X,W)
            Loss = 0.5*np.mean(np.square(Y-PRED))
            mse.append(Loss)

            dL_dW= np.matmul(np.transpose(X),np.matmul(X,W)-Y)
            W=W-learn_rate*dL_dW

            if i % display_step == 0:
                print("i: %i, Loss:%f" % (i,mse[i]))
```

```
i: 0, Loss:4368.213908
i: 50, Loss:413.185263
i: 100, Loss:108.845176
i: 150, Loss:84.920786
i: 200, Loss:82.638199
i: 250, Loss:82.107310
i: 300, Loss:81.782545
i: 350, Loss:81.530512
i: 400, Loss:81.329266
i: 450, Loss:81.167833
i: 500, Loss:81.037990
```

10.4 TensorFlow实现梯度下降法

训练模型

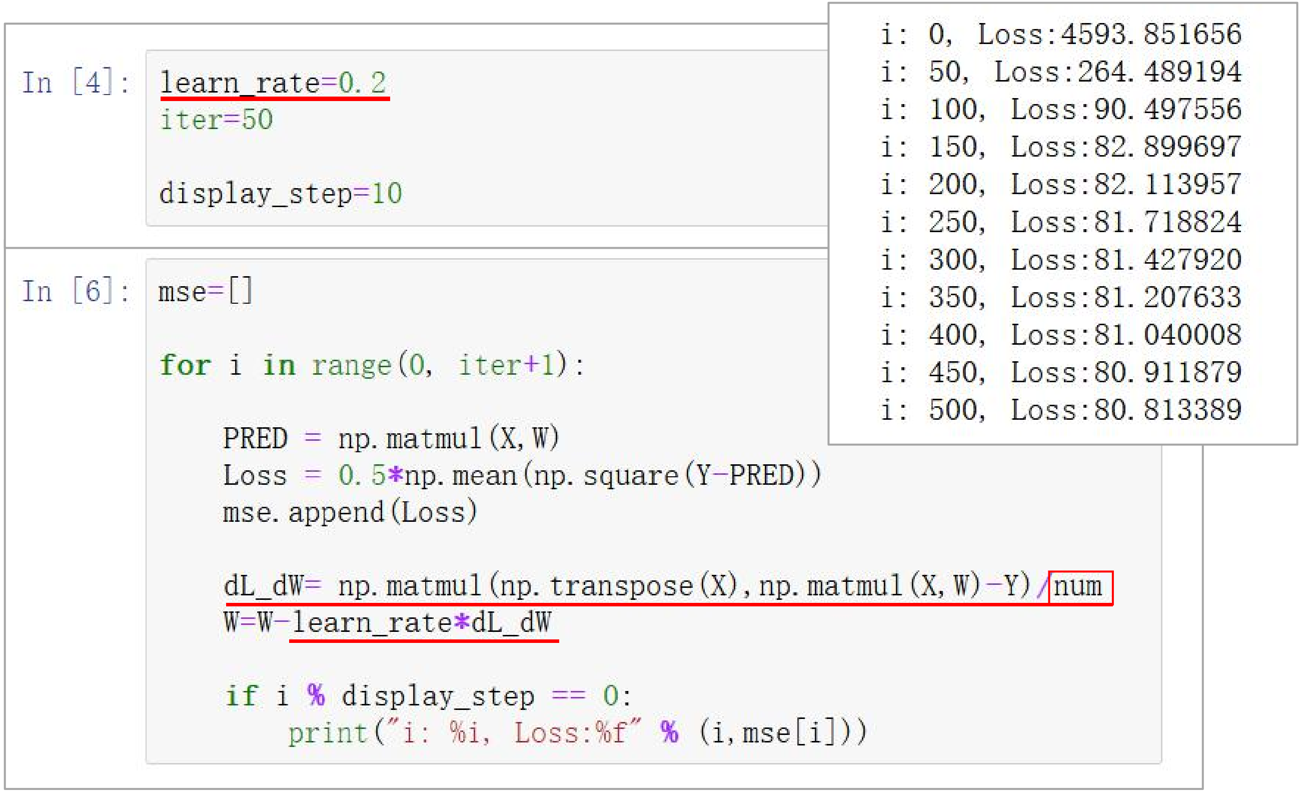


```
In [6]: mse=[]

        for i in range(0, iter+1):

            PRED = np.matmul(X,W)
            Loss = 0.5*np.mean(np.square(Y-PRED))
            mse.append(Loss)

            dL_dW= np.matmul(np.transpose(X),np.matmul(X,W)-Y)
            W=W-learn_rate*dL_dW

            if i % display_step == 0:
                print("i: %i, Loss:%f" % (i,mse[i]))
```

$$Loss=\frac{1}{2\boxed{n}}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2$$

$$\frac{\partial Loss}{\partial W}=\frac{1}{\boxed{n}}X^T(XW-Y)=0$$

10.4 TensorFlow实现梯度下降法



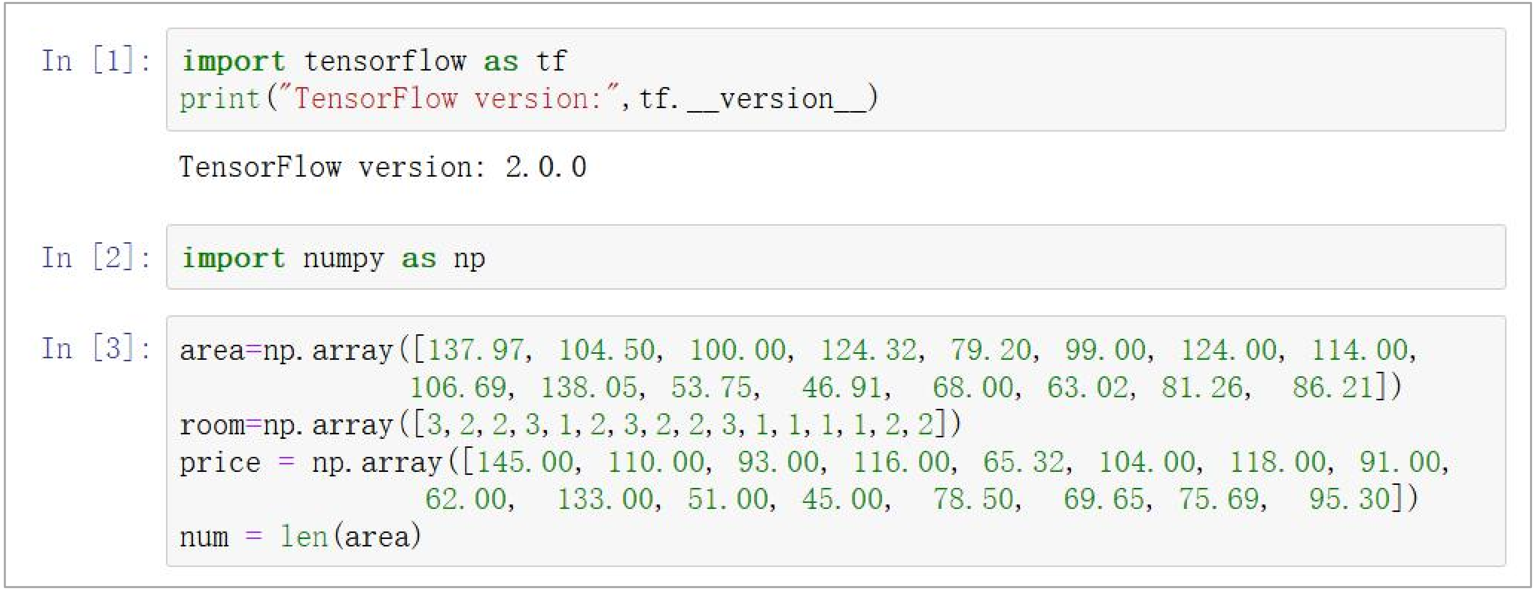
```
In [4]: learn_rate=0.2
        iter=50

        display_step=10
```

```
In [6]: mse=[]

        for i in range(0, iter+1):

            PRED = np.matmul(X,W)
            Loss = 0.5*np.mean(np.square(Y-PRED))
            mse.append(Loss)

            dL_dW= np.matmul(np.transpose(X),np.matmul(X,W)-Y)/num
            W=W-learn_rate*dL_dW

            if i % display_step == 0:
                print("i: %i, Loss:%f" % (i,mse[i]))
```

```
i: 0, Loss:4593.851656
i: 50, Loss:264.489194
i: 100, Loss:90.497556
i: 150, Loss:82.899697
i: 200, Loss:82.113957
i: 250, Loss:81.718824
i: 300, Loss:81.427920
i: 350, Loss:81.207633
i: 400, Loss:81.040008
i: 450, Loss:80.911879
i: 500, Loss:80.813389
```

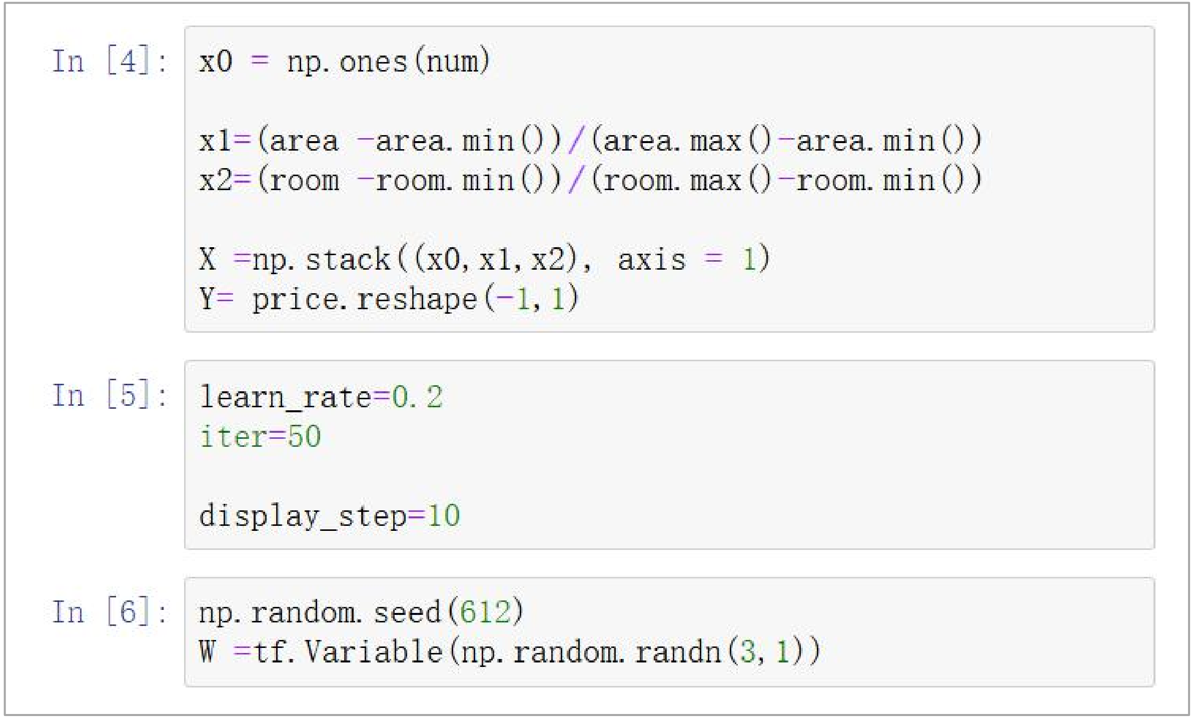
10.4 TensorFlow实现梯度下降法

TensorFlow实现多元线性回归



```
In [1]: import tensorflow as tf
        print("TensorFlow version:",tf.__version__)

        TensorFlow version: 2.0.0

In [2]: import numpy as np

In [3]: area=np.array([137.97, 104.50, 100.00, 124.32, 79.20, 99.00, 124.00, 114.00,
                       106.69, 138.05, 53.75,  46.91,  68.00, 63.02, 81.26,  86.21])
        room=np.array([3,2,2,3,1,2,3,2,2,3,1,1,1,1,2,2])
        price = np.array([145.00, 110.00, 93.00, 116.00, 65.32, 104.00, 118.00, 91.00,
                           62.00,  133.00, 51.00, 45.00,  78.50, 69.65,  75.69,  95.30])
        num = len(area)
```

10.4 TensorFlow实现梯度下降法

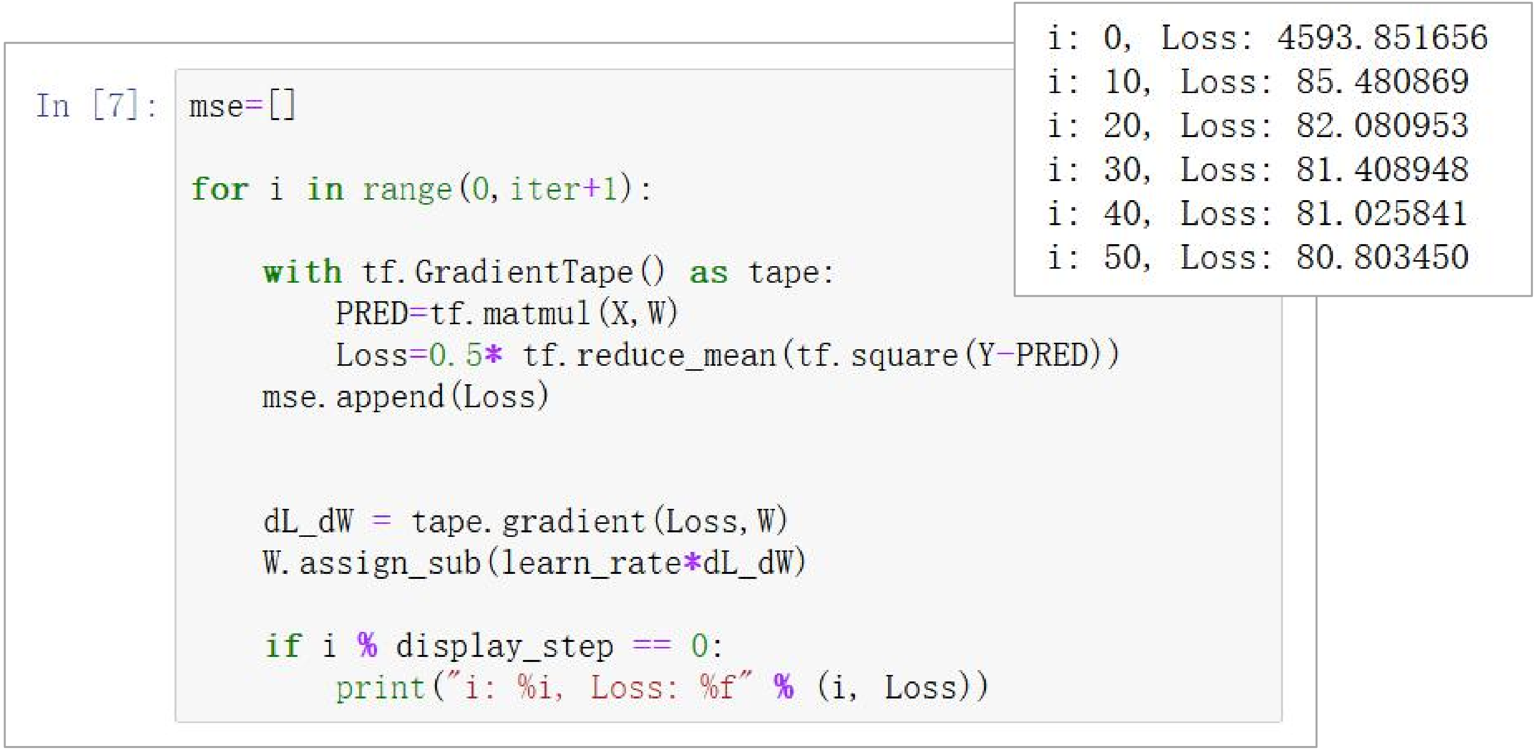


In [4]:
```
x0 = np.ones(num)

x1=(area -area.min())/(area.max()-area.min())
x2=(room -room.min())/(room.max()-room.min())

X =np.stack((x0,x1,x2), axis = 1)
Y= price.reshape(-1,1)
```

In [5]:
```
learn_rate=0.2
iter=50

display_step=10
```

In [6]:
```
np.random.seed(612)
W =tf.Variable(np.random.randn(3,1))
```

10.4 TensorFlow实现梯度下降法



```
In [7]: mse=[]

        for i in range(0,iter+1):

            with tf.GradientTape() as tape:
                PRED=tf.matmul(X,W)
                Loss=0.5* tf.reduce_mean(tf.square(Y-PRED))
            mse.append(Loss)


            dL_dW = tape.gradient(Loss,W)
            W.assign_sub(learn_rate*dL_dW)

            if i % display_step == 0:
                print("i: %i, Loss: %f" % (i, Loss))
```

```
i: 0, Loss: 4593.851656
i: 10, Loss: 85.480869
i: 20, Loss: 82.080953
i: 30, Loss: 81.408948
i: 40, Loss: 81.025841
i: 50, Loss: 80.803450
```