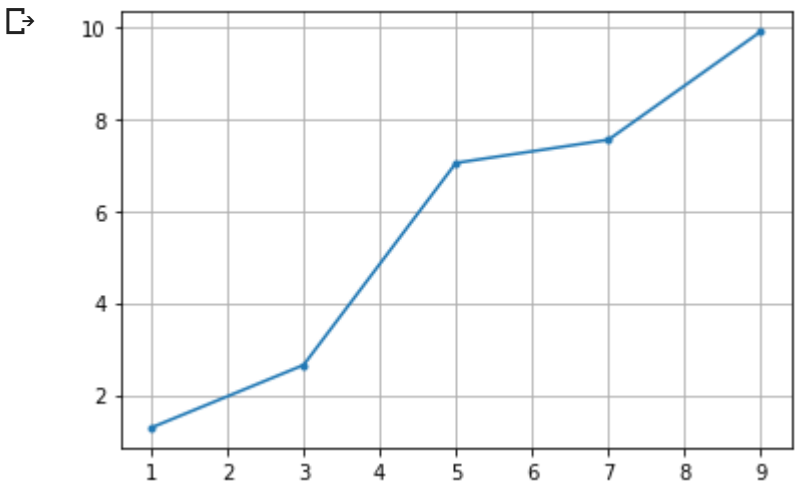
```
#import tensorflow as tf
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

import numpy as np
import matplotlib.pyplot as plt


x_train = [1, 3, 5,7, 9]

y_train = [2, 4, 6, 8, 10]

signal_length = len(x_train)
y_noise = np.random.normal(0, 1, signal_length)

y_train = y_train + y_noise


plt.plot(x_train, y_train,'.-')
plt.grid()
```




```
    w0 = 10.0;
    b0 = 1.0;

    W = tf.Variable(w0*tf.ones([1]), name='weight')
    b = tf.Variable(b0*tf.ones([1]), name='bias')


hypothesis = x_train * W + b


loss = tf.reduce_mean(tf.square(hypothesis - y_train))


optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(loss)


sess = tf.Session()
```

```
sess.run(tf.global_variables_initializer())


nb_epoch = 1001
vloss = [] #empty list
vb = [] #empty list
vw = [] #empty list
for step in range(nb_epoch):
    sess.run(train)
    loss1 = sess.run(loss)
    vloss.append(loss1)
    vb.append(w1)

    if step % 50 == 0: # 5번마다
        w1 = sess.run(W)[0] # 기울기
        vw.append(w1)
        b1 = sess.run(b)[0] # bias
        vb.append(b1)

        print(step,'Wt', loss1, 'Wt', w1, 'Wt',b1)
```

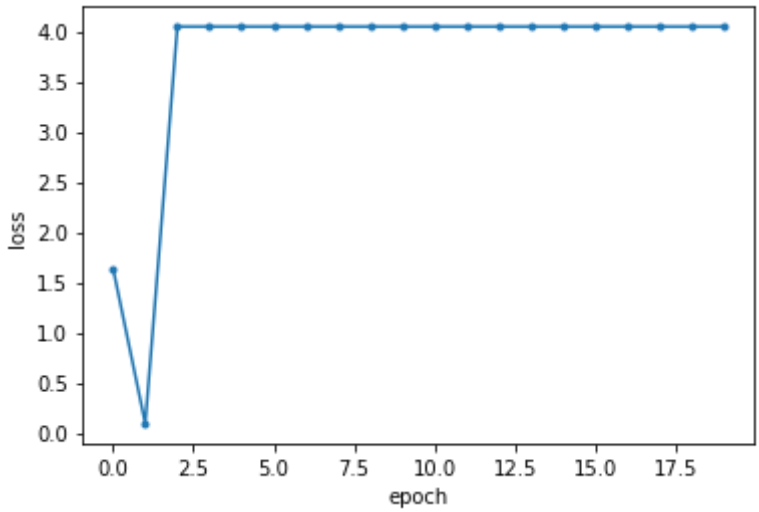
```
0       283.8951        4.0452332       0.093658544
50      0.5752155       1.1656777       -0.23830345
100     0.5611395       1.153036        -0.15546744
150     0.55238634      1.1430669       -0.0901438
200     0.54694307      1.1352054       -0.0386302
250     0.5435576       1.1290058       0.0019928673
300     0.5414526       1.1241169       0.03402782
350     0.54014355      1.1202617       0.059290294
400     0.5393294       1.1172214       0.079212055
450     0.53882307      1.1148238       0.09492209
500     0.5385083       1.1129332       0.107310906
550     0.53831255      1.1114422       0.11708063
600     0.53819054      1.1102664       0.124784924
650     0.5381149       1.1093392       0.1308605
700     0.5380677       1.108608        0.13565154
750     0.53803873      1.1080315       0.13942976
800     0.5380202       1.1075767       0.14240927
850     0.53800887      1.1072181       0.14475876
900     0.5380019       1.1069355       0.14661163
950     0.5379977       1.1067125       0.14807273
1000    0.53799474      1.1065366       0.14922497
```

```
plt.plot(vloss[:20],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```

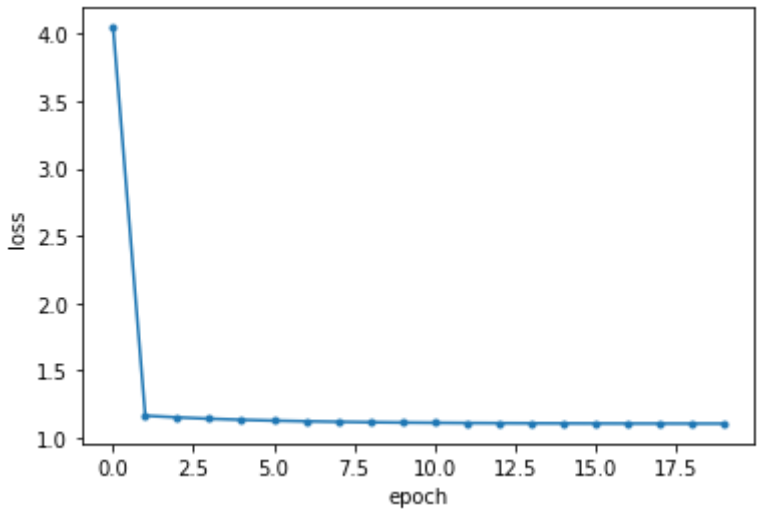

```
plt.plot(vb[:20],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```




```
plt.plot(vw[:20],'.-')
plt.xlabel('epoch')
plt.ylabel('loss')
```

```
Text(0, 0.5, 'loss')
```




```
w1 = sess.run(W)[0] # 기울기
b1 = sess.run(b)[0] # bias


print(w1, b1)
```

```
1.1065366 0.14922497
```

```
str1 = 'y = ' + str(w1) +'x + ' + str(b1)
print(str1)
```

```
y = 1.1065366x + 0.14922497
```

```
plt.figure(figsize=(10,8)) # figsize를 바꾸어보세요
plt.plot(x_train, y_train,'o') #train data 그리기

# 직선 그래프를 그리기 위한 코드
# 그래프의 x좌표를 일정 간격으로 설정함
x1 = np.linspace(np.min(x_train)-1, np.max(x_train)+1)
y1 = w1*x1 + b1
plt.plot(x1, y1)

plt.grid() # 격자
#plt.axis((np.min(x_train) - 1, np.max(x_train) + 1, np.min(y_train) - 1, np.max(y_train) + 1))
plt.title(str1)
```

```
Text(0.5, 1.0, 'y = 1.1065366x + 0.14922497')
```