```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# main arrays:
x = np.array([1, 2, 3, 4, 5])
y = np.array([50, 55, 60, 65, 70])
# learn rate:
learn_rate = 0.1
```

build a simple Perceptron algorithm 'sample (1) with epoch = 1000':

```
# values:
w_1 = 0
b_1 = 0
losses_1 = []
n_1 = 1000


for epoch in range(n_1):
     # calculatethe output as array:
    output = (w_1*x + b_1)
    # calculate the error:
    error = output - y
    # calculate MSE:
    loss = np.pow(error, 2)
    losses_1.append(np.mean(loss))

    # calculate the gradient:
    dw = (1/x.shape[0]) * np.dot(x, error)
    db = (1/x.shape[0]) * np.sum(error)

    # update the weight and bias:
    w_1 -= learn_rate * dw
    b_1 -= learn_rate * db



print('real values: ',np.round(y, 2))
print('predicates: ',np.round((w_1*x + b_1), 2))
print('loss: ',np.round(losses_1[-1], 2))
print('whight: ',np.round(w_1, 2))
print('intercept: ',np.round(b_1, 2))
```

```
real values:  [50 55 60 65 70]
predicates:  [50. 55. 60. 65. 70.]
loss:  0.0
```

```
whight:  5.0
intercept:  45.0
```

build simple Perceptron algorithm 'sample (2) with epoch = 500':

```
# values:
w_2 = 0
b_2 = 0
losses_2 = []
n_2 = 500



for epoch in range(n_2):
    # calculate the output as array:
    output = (w_2*x + b_2)
    # calculate the error:
    error = output - y
    # calculate MSE:
    loss = np.pow(error, 2)
    losses_2.append(np.mean(loss))

    # calculate the gradient:
    dw = (1/x.shape[0]) * np.dot(x, error)
    db = (1/x.shape[0]) * np.sum(error)

    #update the weight and bias:
    w_2 -= learn_rate * dw
    b_2 -= learn_rate * db



print('real values: ',np.round(y, 2))
print('predicates: ',np.round((w_2*x + b_2), 2))
print('loss: ',np.round(losses_2[-1], 2))
print('whight: ',np.round(w_2, 2))
print('intercept: ',np.round(b_2, 2))
```

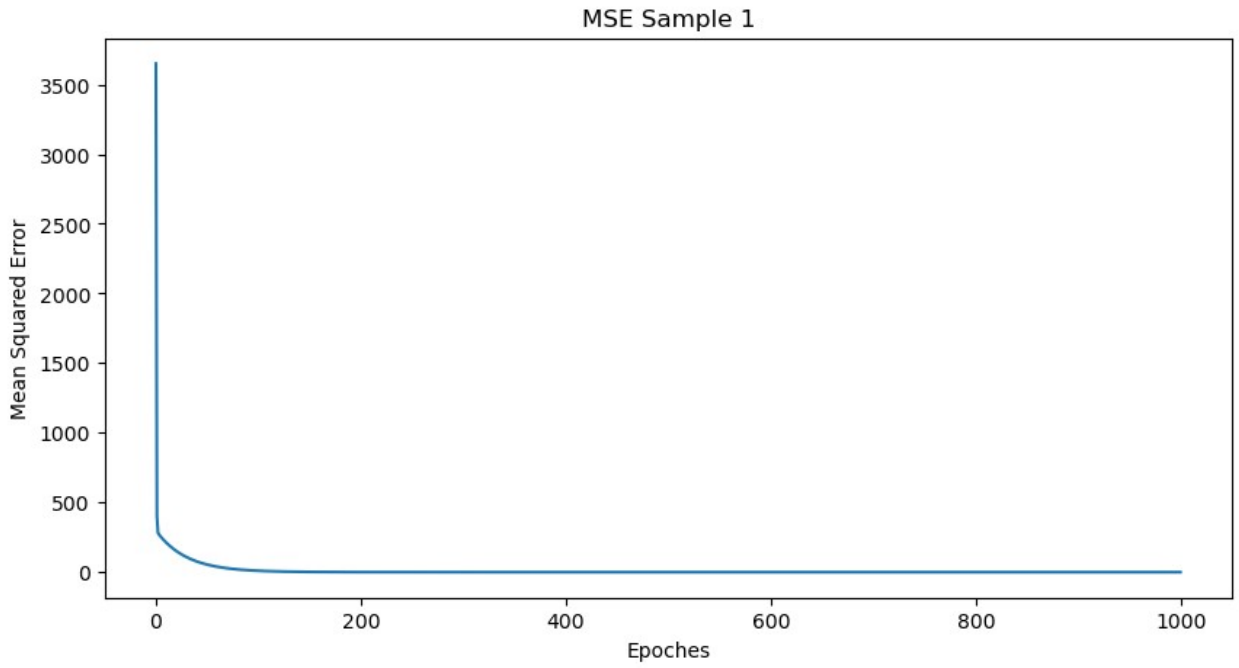
```
real values:  [50 55 60 65 70]
predicates:  [49.99 55.   60.   65.   70.  ]
loss:  0.0
whight:  5.0
intercept:  44.99
```

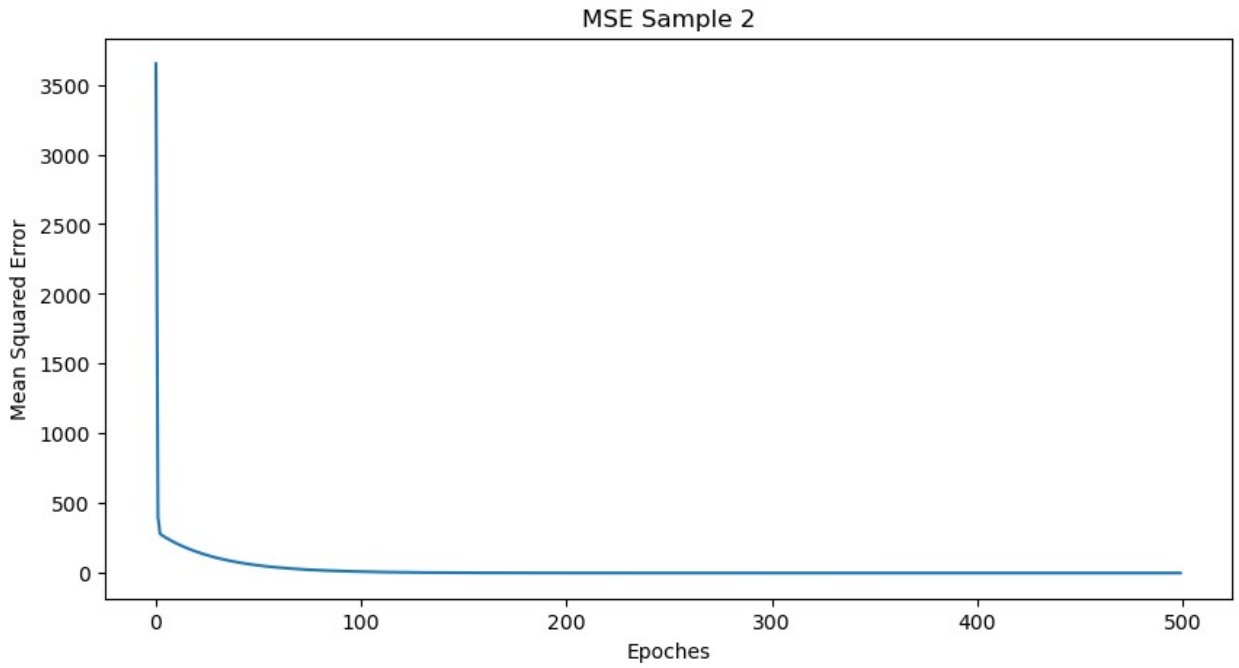
draw the charts by matplotlib and seaborn:

```
# chart 1:
plt.figure(figsize=(10,5))
sns.lineplot(data=losses_1)
plt.xlabel('Epoches')
plt.ylabel('Mean Squared Error')
plt.title('MSE Sample 1')

#chart 2:
plt.figure(figsize=(10,5))
sns.lineplot(data=losses_2)
plt.xlabel('Epoches')
plt.ylabel('Mean Squared Error')
plt.title('MSE Sample 2')

plt.show()
```





insght:

- we notice that the line of the first model which has 1000 epoches is smoother than the second one, which mean it arrived to convengece fast than the second one and more accurate.