A graph with a blue line

Description automatically generated2.1.2  
def least\_squares(A, x, b):  
 return np.linalg.norm(A.dot(x) - b) \*\* 2 / 2  
  
  
def least\_squares\_grad(A, x, b):  
 return A.T.dot(A.dot(x) - b)  
  
  
*# W is the weight matrix  
# X is the input matrix  
# Y is the output matrix*def sgd(lossFunction, gradFunction, W, X, Y, n\_epochs, learning\_rate, batch\_size):  
 loss = []  
 for epoch in range(n\_epochs):  
 *# Decrease the learning rate every 100 epochs to* if epoch % 100 == 0:  
 learning\_rate /= 10  
 *# Shuffle the data* indices = np.arange(X.shape[0])  
 np.random.shuffle(indices)  
 X\_shuffled = X[indices]  
 Y\_shuffled = Y[indices]  
  
 *# Run the batches* for i in range(0, X.shape[0], batch\_size):  
 X\_batch = X\_shuffled[i:i + batch\_size]  
 Y\_batch = Y\_shuffled[i:i + batch\_size]  
 W -= learning\_rate \* gradFunction(X\_batch, W, Y\_batch)  
  
 *# Record the average loss per epoch* loss\_epoch = lossFunction(X, W, Y)  
 loss.append(loss\_epoch)  
  
 return W, loss  
  
  
def main():  
 *# Generate random data* A = random(1000, 1000, density=0.01, format='csr')  
 x = np.random.randn(1000)  
 b = A.dot(x)  
  
 *# Run SGD* W = np.random.randn(1000)  
 W, loss = sgd(least\_squares, least\_squares\_grad, W, A, b, 100, 0.1, 100)  
  
 *# Plot the loss* plt.plot(loss)  
 plt.xlabel('Epoch')  
 plt.ylabel('Loss')  
 plt.show()