The gradient descent algorithm I implemented successfully converged and predicted the correct weight and bias terms as shown by the plot and console log. The logic I used to implement the algorithm came directly from slides 9 and 10 and should be straightforward to understand.

```
# Hint: please check the slides of the gradient descent (1.3, the 9th to 10th page)
#########################################################################
"""
notes:
p=2
y size = n x 1 = [1000 x 1]
X size = n x (p+1) = [1000 x 2] ?
beta size = (p+1) x 1 = [2 x 1]

basically our output hb(x) = beta[0] + xbatch * beta[1]
"""

def gradient_descent(beta, lr, x_batch, y_batch):
    alpha = lr

    slope = beta[0]
    intercept = beta[1]

    hb = slope * x_batch + intercept                              #define output
    cost = (1/2) * np.sum([val**2 for val in (hb - y_batch)])     # define cost function aka J(beta)

    dslope = np.sum((hb - y_batch) * x_batch)                     # differentiate with respect to the weght component beta[0] aka slope
    dintercept = np.sum((hb - y_batch))                           # differentiate with respect to the bias component beta[1] aka intercept

    slope_next = slope - alpha * dslope                           # put all the terms together to weight/slope for the next slope term
    intercept_next = intercept - alpha * dintercept               # put all the terms together to bias/intercept for the next slope term


    weight_next = slope_next
    bias_next =   intercept_next
    beta_next = [weight_next,bias_next]

    return cost, beta_next

#########################################################################
```

```
77  #########################################################################
78  #                        Main Function
79  ##########################################################
80  beta = np.random.randn(2)
81  cost = 0
82
83  for epoch in range(training_epochs):
84      cost, beta_next = gradient_descent(beta,learning_rate,x
85      print('Epoch %3d, cost %.3f, weight %.3f bias is %.3f'
86      beta = beta_next
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```
Epoch 975, cost 504.402, weight 2.020 bias is -3.035
Epoch 976, cost 504.402, weight 2.020 bias is -3.035
Epoch 977, cost 504.402, weight 2.020 bias is -3.035
Epoch 978, cost 504.402, weight 2.020 bias is -3.035
Epoch 979, cost 504.402, weight 2.020 bias is -3.035
Epoch 980, cost 504.402, weight 2.020 bias is -3.035
Epoch 981, cost 504.402, weight 2.020 bias is -3.035
Epoch 982, cost 504.402, weight 2.020 bias is -3.035
Epoch 983, cost 504.402, weight 2.020 bias is -3.035
Epoch 984, cost 504.402, weight 2.020 bias is -3.035
Epoch 985, cost 504.402, weight 2.020 bias is -3.035
Epoch 986, cost 504.402, weight 2.020 bias is -3.035
Epoch 987, cost 504.402, weight 2.020 bias is -3.035
Epoch 988, cost 504.402, weight 2.020 bias is -3.035
Epoch 989, cost 504.402, weight 2.020 bias is -3.035
Epoch 990, cost 504.402, weight 2.020 bias is -3.035
Epoch 991, cost 504.402, weight 2.020 bias is -3.035
Epoch 992, cost 504.402, weight 2.020 bias is -3.035
Epoch 993, cost 504.402, weight 2.020 bias is -3.035
Epoch 994, cost 504.402, weight 2.020 bias is -3.035
Epoch 995, cost 504.402, weight 2.020 bias is -3.035
Epoch 996, cost 504.402, weight 2.020 bias is -3.035
Epoch 997, cost 504.402, weight 2.020 bias is -3.035
Epoch 998, cost 504.402, weight 2.020 bias is -3.035
Epoch 999, cost 504.402, weight 2.020 bias is -3.035
Epoch 1000, cost 504.402, weight 2.020 bias is -3.035
```