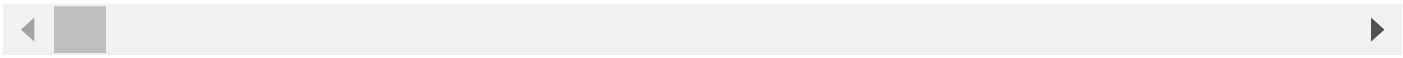
```
# produced by Jerry Wang

import numpy as np
from sympy import symbols, diff


# write the loss function (MSE) for linear regression on the three points (0,1) (2,0) (1,3)
# MSE = 1/n * sum (y[i]-y[i]_pred)^2 for i in range (0, n, ++1), notice that i starts from 0 due to python indexing nature.
def mean_squared_error(m, b):
    points = [(0, 1), (2, 0), (1, 3)]
    n = len(points)

    # initialize the loss to 0
    loss = 0

    # loop through each point to calculate loss
    for x, y in points:
        y_pred = m * x + b
        loss += (y - y_pred)**2

    # calculate mean loss (MSE)
    loss = loss / n

    return loss


# for cost function f(x) = 9*x1^2 + x2^2
# a) compute gradient
x1, x2 = symbols('x1 x2')
f = 9*x1**2 + x2**2

df_dx1 = diff(f, x1)
df_dx2 = diff(f, x2)

gradient = np.array([df_dx1, df_dx2])
print(gradient)
```

```
     [18*x1 2*x2]
```

```
# b) write the gradient descent iteration x(n + 1) = x(n) – δ∇f(x(n)), where x(n) = (x1(n), x2(n)).
# the concept here is to gradually move each component (coordinate) of x towards 0 (minimize the loss).
# x1(n+1) = x1(n) - δ*(18x1(n)); x2(n+1) = x2(n) - δ*(2x2(n))

def gradient(x1, x2):
    grad_x1 = 18 * x1
    grad_x2 = 2 * x2
    return (grad_x1, grad_x2)

def gradient_descent(initial_point, learning_rate, iterations):
    x1, x2 = initial_point
    history = [(x1, x2)]  # store the history of points

    for i in range(iterations):
        grad_x1, grad_x2 = gradient(x1, x2)

        # update the point towards 0 by the learning rate
        x1 = x1 - learning_rate * grad_x1
        x2 = x2 - learning_rate * grad_x2

        history.append((x1, x2))

    return x1, x2, history

initial_point = (1.0,2.0) # starting point (x1, x2), change as needed
learning_rate = 0.1 # step size, change as needed
iterations = 100 # number of iterations, change as needed

final_x1, final_x2, history = gradient_descent(initial_point, learning_rate, iterations)
print(final_x1, final_x2)
print(history)
# I wrote this function to store the final coordinates of x1 and x2, and the historical coordinates reflects the update from every iteration.
```

```
     2.037035976334516e-10 4.0740719526689754e-10
     [(1.0, 2.0), (-0.8, 1.6), (0.6400000000000001, 1.28), (-0.5120000000000002, 1.024), (0.4096000000000003, 0.8192), (-0.3276800000000003,
```



✓ 0s completed at 1:07 PM