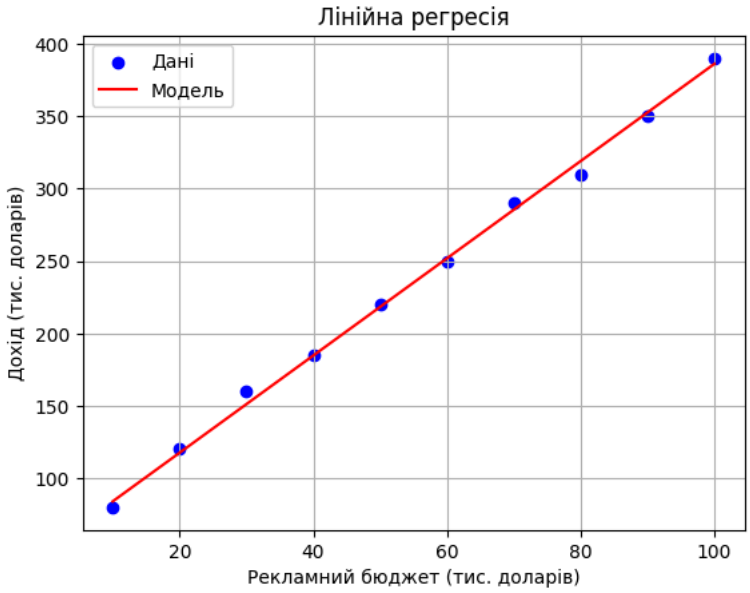
```
import numpy as np
import matplotlib.pyplot as plt


x_train = np.array([10, 20, 30, 40, 50, 60, 70, 80, 90, 100])
y_train = np.array([80, 120, 160, 185, 220, 250, 290, 310, 350, 390])


def compute_model_output(x, w, b):
    return w * x + b



def compute_cost(x, y, w, b):
    m = len(x)
    cost = 0
    for i in range(m):
        f_wb = compute_model_output(x[i], w, b)
        cost += (f_wb - y[i]) ** 2
    return cost / (2 * m)


def compute_gradient(x, y, w, b):
    m = len(x)
    dj_dw = 0
    dj_db = 0
    for i in range(m):
        f_wb = compute_model_output(x[i], w, b)
        dj_dw += (f_wb - y[i]) * x[i]
        dj_db += (f_wb - y[i])
    dj_dw /= m
    dj_db /= m
    return dj_dw, dj_db



def gradient_descent(x, y, w_init, b_init, alpha, iterations):
    w = w_init
    b = b_init
    J_history = []

    for i in range(iterations):
        dj_dw, dj_db = compute_gradient(x, y, w, b)
        w -= alpha * dj_dw
        b -= alpha * dj_db
        J_history.append(compute_cost(x, y, w, b))

    return w, b, J_history



w_init = 3
b_init = 50
alpha = 0.0001
iterations = 10000

w_final, b_final, J_hist = gradient_descent(x_train, y_train, w_init, b_init, alpha, iterations)
print(f"Оптимальні параметри: w = {w_final:.2f}, b = {b_final:.2f}")
```

```
Оптимальні параметри: w = 3.36, b = 50.45
```

```
plt.scatter(x_train, y_train, color='blue', label='Дані')
plt.plot(x_train, compute_model_output(x_train, w_final, b_final), color='red', label='Модель')
plt.xlabel('Рекламний бюджет (тис. доларів)')
plt.ylabel('Дохід (тис. доларів)')
plt.legend()
plt.grid(True)
plt.title('Лінійна регресія')
plt.show()
```




```
plt.plot(J_hist)
plt.xlabel('Ітерації')
plt.ylabel('Функція вартості J(w, b)')
plt.title('Збіжність градієнтного спуску')
plt.grid(True)
plt.show()
```