PRACTICAL NO:4

Aim-:Implement Gradient Descent Algorithm to find the local minima of a function.

For example, find the local minima of the function $y=(x+3)^2$ starting from the point x=2.

INPUT

```
import matplotlib.pyplot as plt
def function(x):
    return (x + 3)**2
def gradient(x):
    return 2 * (x + 3)
# Gradient Descent Algorithm
def gradient_descent(starting_x, learning_rate, num_iterations):
    x = starting_x
    x_values = [x]
# Track x values for visualization
for i in range(num_iterations):
grad = gradient(x)
 x = x - learning_rate * grad
x_values.append(x)
if abs(grad) < 1e-6:
return x, x_values


# Parameters for Gradient Descent
starting_x = 2
# Starting point
learning_rate = 0.1
# Learning rate (alpha)
num_iterations = 100
# Maximum number of iterations
# Perform Gradient Descent
```

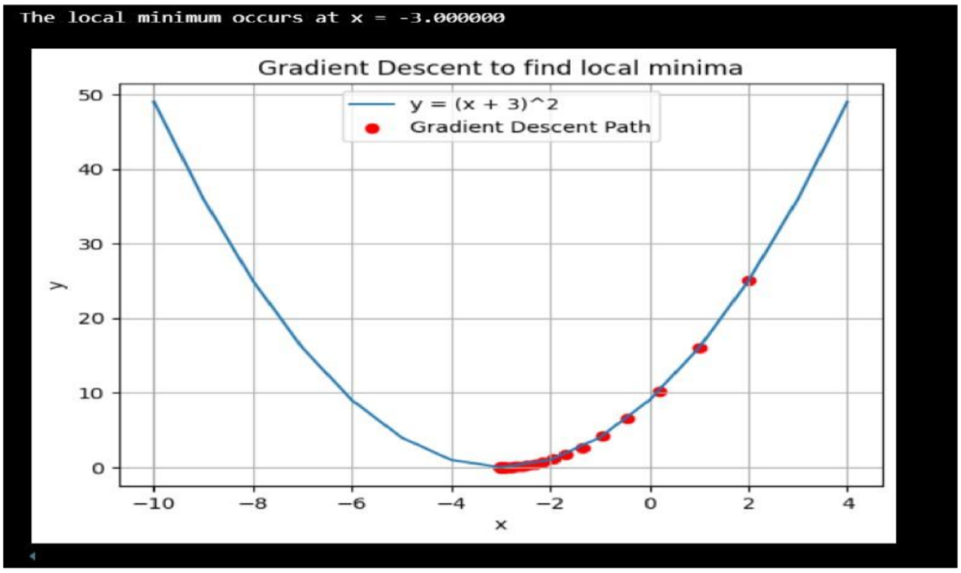
```
final_x, x_values = gradient_descent(starting_x, learning_rate, num_iterations)

# Print the final result
print(f"The local minimum occurs at x = {final_x:.6f}")

# Plotting the function and the gradient descent path
x_range = range(-10, 5)
y_values = [function(x) for x in x_range]

plt.plot(x_range, y_values, label='y = (x + 3)^2')
plt.scatter(x_values, [function(x) for x in x_values], color='red', label='Gradient Descent Path')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Gradient Descent to find local minima')
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUT



```
The local minimum occurs at x = -3.000000
```