```python
# Step 1: Import libraries
import matplotlib.pyplot as plt

# Step 2: Define the function and its derivative
def f(x):
    return (x + 3)**2

def df(x):
    return 2 * (x + 3)

# Step 3: Initialize parameters
x = 2                # starting point
learning_rate = 0.1  # step size
epochs = 50          # number of iterations

# Step 4: Perform Gradient Descent
x_values = [x]
for i in range(epochs):
    grad = df(x)
    x = x - learning_rate * grad
    x_values.append(x)
    print(f"Iteration {i+1}: x = {x:.5f}, f(x) = {f(x):.5f}")
```

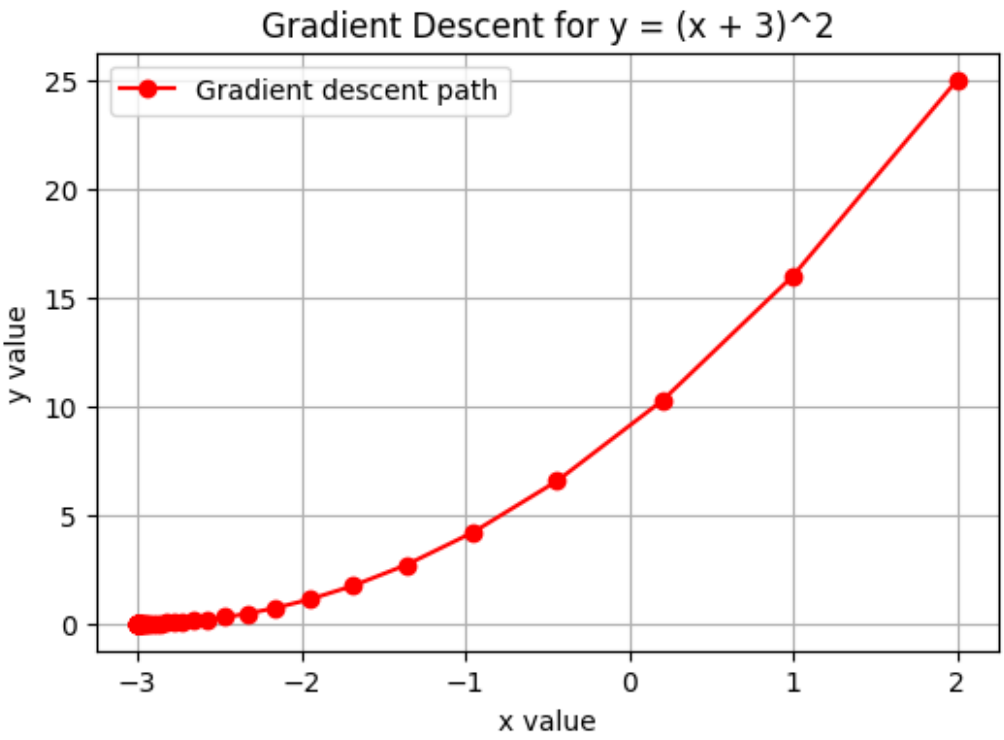
```
Iteration 1: x = 1.00000, f(x) = 16.00000
Iteration 2: x = 0.20000, f(x) = 10.24000
Iteration 3: x = -0.44000, f(x) = 6.55360
Iteration 4: x = -0.95200, f(x) = 4.19430
Iteration 5: x = -1.36160, f(x) = 2.68435
Iteration 6: x = -1.68928, f(x) = 1.71799
Iteration 7: x = -1.95142, f(x) = 1.09951
Iteration 8: x = -2.16114, f(x) = 0.70369
Iteration 9: x = -2.32891, f(x) = 0.45036
Iteration 10: x = -2.46313, f(x) = 0.28823
Iteration 11: x = -2.57050, f(x) = 0.18447
Iteration 12: x = -2.65640, f(x) = 0.11806
Iteration 13: x = -2.72512, f(x) = 0.07556
Iteration 14: x = -2.78010, f(x) = 0.04836
Iteration 15: x = -2.82408, f(x) = 0.03095
Iteration 16: x = -2.85926, f(x) = 0.01981
Iteration 17: x = -2.88741, f(x) = 0.01268
Iteration 18: x = -2.90993, f(x) = 0.00811
Iteration 19: x = -2.92794, f(x) = 0.00519
Iteration 20: x = -2.94235, f(x) = 0.00332
Iteration 21: x = -2.95388, f(x) = 0.00213
Iteration 22: x = -2.96311, f(x) = 0.00136
Iteration 23: x = -2.97049, f(x) = 0.00087
Iteration 24: x = -2.97639, f(x) = 0.00056
Iteration 25: x = -2.98111, f(x) = 0.00036
Iteration 26: x = -2.98489, f(x) = 0.00023
Iteration 27: x = -2.98791, f(x) = 0.00015
```

```
Iteration 28: x = -2.99033, f(x) = 0.00009
Iteration 29: x = -2.99226, f(x) = 0.00006
Iteration 30: x = -2.99381, f(x) = 0.00004
Iteration 31: x = -2.99505, f(x) = 0.00002
Iteration 32: x = -2.99604, f(x) = 0.00002
Iteration 33: x = -2.99683, f(x) = 0.00001
Iteration 34: x = -2.99746, f(x) = 0.00001
Iteration 35: x = -2.99797, f(x) = 0.00000
Iteration 36: x = -2.99838, f(x) = 0.00000
Iteration 37: x = -2.99870, f(x) = 0.00000
Iteration 38: x = -2.99896, f(x) = 0.00000
Iteration 39: x = -2.99917, f(x) = 0.00000
Iteration 40: x = -2.99934, f(x) = 0.00000
Iteration 41: x = -2.99947, f(x) = 0.00000
Iteration 42: x = -2.99957, f(x) = 0.00000
Iteration 43: x = -2.99966, f(x) = 0.00000
Iteration 44: x = -2.99973, f(x) = 0.00000
Iteration 45: x = -2.99978, f(x) = 0.00000
Iteration 46: x = -2.99983, f(x) = 0.00000
Iteration 47: x = -2.99986, f(x) = 0.00000
Iteration 48: x = -2.99989, f(x) = 0.00000
Iteration 49: x = -2.99991, f(x) = 0.00000
Iteration 50: x = -2.99993, f(x) = 0.00000
```

```
# Step 5: Display final result
print("\n Local minimum occurs at x =", round(x, 5))
print("Minimum value of function y =", round(f(x), 5))
```

```
 Local minimum occurs at x = -2.99993
Minimum value of function y = 0.0
```

```
# Step 6: Plot the path of gradient descent
plt.figure(figsize=(6,4))
plt.plot(x_values, [f(i) for i in x_values], 'ro-', label='Gradient
descent path')
plt.title("Gradient Descent for y = (x + 3)^2")
plt.xlabel("x value")
plt.ylabel("y value")
plt.legend()
plt.grid(True)
plt.show()
```


Gradient Descent for y = (x + 3)^2
Gradient descent path
y value
x value
25
20
15
10
5
0
−3
−2
−1
0
1
2