```
import matplotlib.pyplot as plt
```

```
# ✅ Step 2 – Define the function and its derivative

"""
In Gradient Descent, we need:
The function whose minimum we want to find
Its derivative (slope), which tells us which direction the function is increasing or decreasing
"""

def f(x):
  return (x + 3)**2     # y = (x + 3)^2

def df(x):
  return 2 * (x+3)      # derivative = 2*(x + 3)

"""
💡 Simple Explanation:
Code  Meaning
def f(x): We are creating a function named f(x)
return (x+3)**2 This function calculates the value of y = (x + 3)²
def df(x):  This creates another function (df) to calculate derivative (slope)
return 2*(x+3)  Derivative of (x+3)² = 2(x + 3)

✅ Why do we need derivative (df)?

Gradient Descent works by moving opposite to the slope.
Derivative tells us the steepness and direction of slope.
"""
```

```
'\n💡 Simple Explanation:\nCode\tMeaning\ndef f(x):\tWe are creating a function named f(x)\nreturn (x+3)**2\tThis function calculates the v
alue of y = (x + 3)²\ndef df(x):\tThis creates another function (df) to calculate derivative (slope)\nreturn 2*(x+3)\tDerivative of (x+3)²
= 2(x + 3)\n\n✅ Why do we need derivative (df)?\n\nGradient Descent works by moving opposite to the slope.\nDerivative tells us the steep
ness and direction of slope.\n'
```

```
#✅ Step 3 – Initialize values (starting point, learning rate, and number of steps)

x = 2                         # starting point
learning_rate = 0.1
epochs = 25                   # how many times to update

"""
💡 Simple Explanation:
Variable    Meaning (in easy words)
x = 2   This is where we start, like guessing a value near the minimum. Here we start from x = 2.
learning_rate = 0.1 This controls how big each step should be when we move towards the minimum. Small = slow but safe, too big = risky.
epochs = 25 How many times we repeat the process of updating x (i.e., how many steps we take downhill).
"""
```

```
'\n💡 Simple Explanation:\nVariable\tMeaning (in easy words)\nx = 2\tThis is where we start, like guessing a value near the minimum. Here w
e start from x = 2.\nlearning_rate = 0.1\tThis controls how big each step should be when we move towards the minimum. Small = slow but saf
e, too big = risky.\nepochs = 25\tHow many times we repeat the process of updating x (i.e., how many steps we take downhill).\n'
```

```
# ✅ Step 4 – Perform Gradient Descent (update x step-by-step)

for i in range(epochs):
    grad = df(x)                  # calculate slope at current x
    x = x - learning_rate * grad  # move in opposite direction of slope
    print(f"Step {i+1}: x = {x:.4f}, f(x) = {f(x):.4f}")
```

```
Step 1: x = 1.0000, f(x) = 16.0000
Step 2: x = 0.2000, f(x) = 10.2400
Step 3: x = -0.4400, f(x) = 6.5536
Step 4: x = -0.9520, f(x) = 4.1943
Step 5: x = -1.3616, f(x) = 2.6844
Step 6: x = -1.6893, f(x) = 1.7180
Step 7: x = -1.9514, f(x) = 1.0995
Step 8: x = -2.1611, f(x) = 0.7037
Step 9: x = -2.3289, f(x) = 0.4504
Step 10: x = -2.4631, f(x) = 0.2882
Step 11: x = -2.5705, f(x) = 0.1845
Step 12: x = -2.6564, f(x) = 0.1181
Step 13: x = -2.7251, f(x) = 0.0756
Step 14: x = -2.7801, f(x) = 0.0484
Step 15: x = -2.8241, f(x) = 0.0309
Step 16: x = -2.8593, f(x) = 0.0198
Step 17: x = -2.8874, f(x) = 0.0127
Step 18: x = -2.9099, f(x) = 0.0081
Step 19: x = -2.9279, f(x) = 0.0052
Step 20: x = -2.9424, f(x) = 0.0033
Step 21: x = -2.9539, f(x) = 0.0021
Step 22: x = -2.9631, f(x) = 0.0014
Step 23: x = -2.9705, f(x) = 0.0009
Step 24: x = -2.9764, f(x) = 0.0006
Step 25: x = -2.9811, f(x) = 0.0004
```

```
#✅ Step 5 – Show the final result (local minimum point)
```

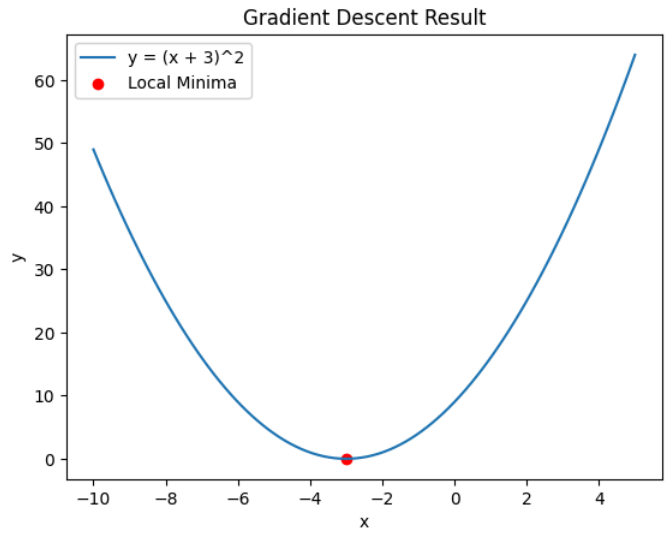
```
print("\nLocal minima occurs at x =", round(x, 2))
```

```
Local minima occurs at x = -2.98
```

```
# OPTIONAL...
# Step 6: (Optional) Visualize the function and minima point
import numpy as np
import matplotlib.pyplot as plt

# Plot the function
X = np.linspace(-10, 5, 100)   # values of x between -10 and 5
Y = f(X)                       # compute f(x)

plt.plot(X, Y, label='y = (x + 3)^2')   # draw the curve
plt.scatter(x, f(x), color='red', label='Local Minima')  # red point at minima
plt.title("Gradient Descent Result")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()
```



Start coding or generate with AI.