Initialize Parameters :

$$X_0 = 3$$

$$Learning\ rate = 0.01$$

$$\frac{dy}{dx} = \frac{d}{dx}(x+5)^2 = 2 * (x+5)$$

Iteration 1 :

$$X_1 = X_0 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_1 = 3 - (0.01) * \big(2 * (3+5)\big) = 2.84$$

Iteration 2 :

$$X_2 = X_1 - (learning\ rate) * (\frac{dy}{dx})$$

$$X_2 = 2.84 - (0.01) * \big(2 * (2.84+5)\big) = 2.6832$$

Step 4 : We can observe that the X value is slowly decreasing and should converge to -5 (the local minima).

Implementing Gradient Descent in Python

Importing libraries

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

We then define the function f(x) and its derivative f'(x) –

```
def f(x):
  return (x+3)**2

def df(x):
  return 2*x + 6
```

F(x) is the function that has to be decreased, and df is its derivative (x). The gradient descent approach makes use of the derivative to guide itself toward the minimum by revealing the function's slope along the way.

The gradient descent function is then defined.

```
def gradient_descent(initial_x, learning_rate, num_iterations):
  x = initial_x
  x_history = [x]

  for i in range(num_iterations):
     gradient = df(x)
     x = x - learning_rate * gradient
     x_history.append(x)

  return x, x_history
```

The starting value of x, the learning rate, and the desired number of iterations are sent to the gradient descent function. In order to save the values of x after each iteration, it initializes x to its original value and generates an empty list. The method then executes the gradient descent for the provided number of iterations, changing x every iteration in accordance with the equation x = x - learning rate * gradient. The function produces a list of every iteration's x values together with the final value of x.

The gradient descent function may now be used to locate the local minimum of f(x) –

```
initial_x = 2
learning_rate = 0.1
num_iterations = 50

x, x_history = gradient_descent(initial_x, learning_rate, num_iterations)

print("Local minimum: {:.2f}".format(x))
```

Output

```
Local minimum: -3.00
```

In this illustration, x is set to 2 at the beginning, with a learning rate of 0.1, and 50 iterations are run. Finally, we publish the value of x, which ought to be close to the local minimum at x=-3.

Plotting the function f(x) and the values of x for each iteration allows us to see the gradient descent process in action –

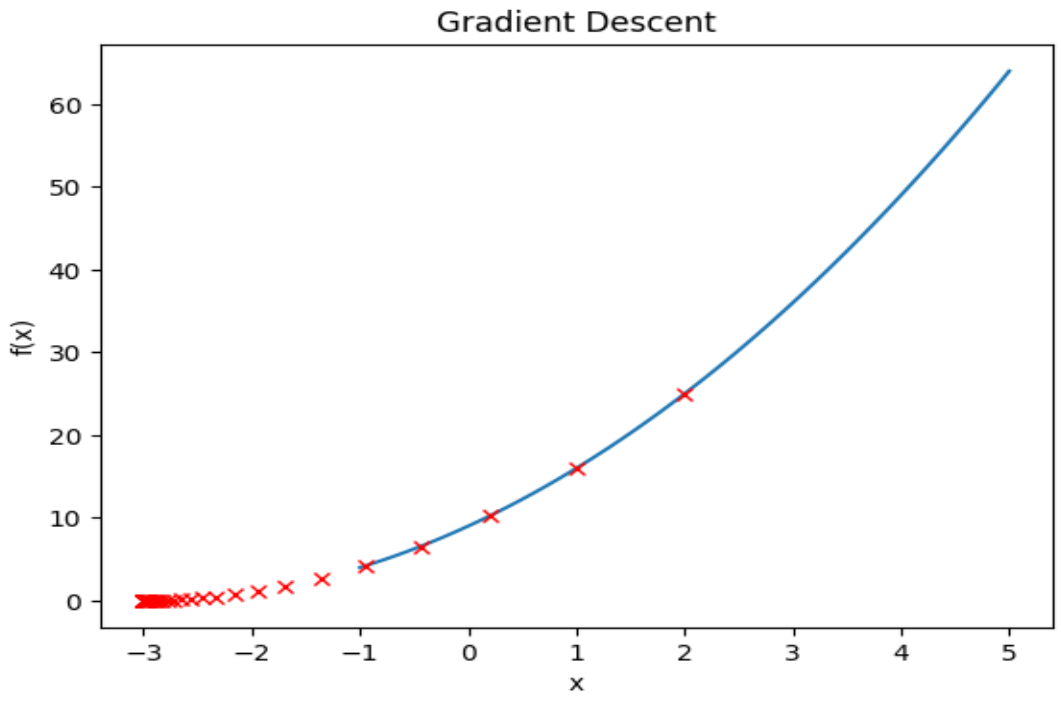
```
#Create a range of x values to plot
x_vals = np.linspace(-1, 5, 100)

#Plot the function f(x)
plt.plot(x_vals, f(x_vals))

# Plot the values of x at each iteration
plt.plot(x_history, f(np.array(x_history)), 'rx')

#Label the axes and add a title
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Gradient Descent')

#Show the plot
plt.show()
```



Conclusion: Thus we have implemented Gradient Descent Algorithm to find the local minima of a function $y=(x+3)^2$ starting from the point x=2 that is -3.