Experiment 4:

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.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
```

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

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

In [ ]:
```
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
```

In [ ]:
```
num_iterations = 50

# Pass num_iterations to the gradient_descent() function
x, x_history = gradient_descent(initial_x=2, learning_rate=0.1, num_iterati

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

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

In [ ]:
```
# 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')
```

In [6]:
```
# Label the axes and add a title
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Gradient Descent')

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

```
Local minimum: -3.00
```