```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Funkcja do minimalizacji
def funkcja(x, y):
    return (x + 3*y)**3 + 2*x

# Pochodne cząstkowe funkcji
def pochodne(x, y):
    df_dx = 3 * (x + 3*y)**2 + 2
    df_dy = 9 * (x + 3*y)**2
    return np.array([df_dx, df_dy])

# Metoda gradientu
def gradient_descent(learning_rate, iterations):
    x = np.random.uniform(1, 100)
    y = np.random.uniform(1, 100)

    history = []

    for _ in range(iterations):
        gradient = pochodne(x, y)
        x = x - learning_rate * gradient[0]
        y = y - learning_rate * gradient[1]
        history.append([x, y, funkcja(x, y)])

    return np.array(history)

# Wizualizacja funkcji
x_vals = np.linspace(1, 100, 100)
y_vals = np.linspace(1, 100, 100)
X, Y = np.meshgrid(x_vals, y_vals)
Z = funkcja(X, Y)

# Wykres 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis', alpha=0.8, edgecolor='k')

# Początkowy punkt
ax.scatter(1, 1, funkcja(1, 1), color='red', marker='o', s=100,
label='Start')

# Znalezienie minimum
learning_rate = 0.01
iterations = 100
history = gradient_descent(learning_rate, iterations)

# Trajektoria minimalizacji
```

```
ax.plot(history[:, 0], history[:, 1], history[:, 2], color='blue',
marker='o', label='Minimization')

# Ostatni punkt - minimum
ax.scatter(history[-1, 0], history[-1, 1], history[-1, 2],
color='green', marker='o', s=100, label='Minimum')

ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('f(X, Y)')
ax.legend()

plt.show()
```