from sympy import symbols, diff, solve, Matrix

import numpy as np

from scipy.optimize import minimize

# Define symbols

x, y, l = symbols('x y lambda')

# Define the objective function and constraint

f = x\*\*2 + y\*\*2

g = x + y - 1

# Define the Lagrangian

L = f - l \* g

# Compute partial derivatives

partials = [diff(L, var) for var in (x, y, l)]

# Solve the system of equations

solution = solve(partials, (x, y, l), dict=True)[0]

# Extract the optimal values

optimal\_x = solution[x]

optimal\_y = solution[y]

# Compute the Hessian matrix

hessian\_list = [[diff(L.diff(var1), var2) for var1 in (x, y, l)] for var2 in (x, y, l)]

hessian\_matrix = Matrix(hessian\_list)

# Display the Hessian matrix

print("Hessian Matrix:")

print(hessian\_matrix)

# Determine the nature of the stationary point

hessian\_determinant = hessian\_matrix.det()

if hessian\_determinant > 0:

    print("Stationary point is a local minimum.")

elif hessian\_determinant < 0:

    print("Stationary point is a local maximum.")

else:

    print("Second-order test inconclusive (saddle point or test fails)")

# Display the optimal solution

print("\nOptimal solution:")

print(f"x: {optimal\_x}")

print(f"y: {optimal\_y}")

# Finding optimal solution using scipy.optimize.minimize

# Define the objective function

def objective\_function(x):

    return x[0]\*\*2 + x[1]\*\*2

# Initial guess

initial\_guess = [0.0, 0.0]

# Minimize the objective function

result = minimize(objective\_function, initial\_guess, method='BFGS')

# Extract the optimal solution

optimal\_solution = result.x

minimum\_value = result.fun

# Print the results

print("\nOptimal Solution (using scipy.optimize.minimize):", optimal\_solution)

print("Minimum Value:", minimum\_value)