import random

# Example function

def f(x):

    return -(x - 3) \*\* 2 + 9   # Maximum at x = 3, f(x) = 9

# Hill climbing for maxima

def hill\_climb\_max(start\_x, step\_size=0.1, max\_iterations=1000):

    current\_x = start\_x

    current\_value = f(current\_x)

    for i in range(max\_iterations):

        neighbors = [current\_x + step\_size, current\_x - step\_size]

        neighbor\_values = [f(x) for x in neighbors]

        # Choose the larger value (maximization)

        best\_neighbor\_value = max(neighbor\_values)

        best\_neighbor = neighbors[neighbor\_values.index(best\_neighbor\_value)]

        if best\_neighbor\_value > current\_value:

            current\_x, current\_value = best\_neighbor, best\_neighbor\_value

        else:

            break

    return current\_x, current\_value

# Hill climbing for minima

def hill\_climb\_min(start\_x, step\_size=0.1, max\_iterations=1000):

    current\_x = start\_x

    current\_value = f(current\_x)

    for i in range(max\_iterations):

        neighbors = [current\_x + step\_size, current\_x - step\_size]

        neighbor\_values = [f(x) for x in neighbors]

        # Choose the smaller value (minimization)

        best\_neighbor\_value = min(neighbor\_values)

        best\_neighbor = neighbors[neighbor\_values.index(best\_neighbor\_value)]

        if best\_neighbor\_value < current\_value:

            current\_x, current\_value = best\_neighbor, best\_neighbor\_value

        else:

            break

    return current\_x, current\_value

# Example runs

start = random.uniform(-10, 10)

x\_max, y\_max = hill\_climb\_max(start)

print("Starting point:", start)

print("Local maximum at x =", x\_max, "with value f(x) =", y\_max)

x\_min, y\_min = hill\_climb\_min(start)

print("Local minimum at x =", x\_min, "with value f(x) =", y\_min)