**AA\_Lab\_07\_Assignment**

**CE\_054**

Aim :- Implementation of Convex-Hull Algorithm using Graham Scan Algorithm.

Code :

'''

Author : Dhruv B Kakadiya

GScan Algo

'''

import matplotlib.pyplot as plt

import math as m

import random as r

anchor = None

def randome\_point\_creation(no\_points, min\_ = 0, max\_ = 70 ):

    return [[r.randint(min\_, max\_), r.randint(min\_, max\_)] for i in range(no\_points)]

def view\_graph(points, convex = None):

    x, y = zip(\*points)

    plt.scatter(x, y, edgecolors="green")

    if (convex != None):

        for i in range(1, len(convex) + 1):

            if (i == len(convex)):

                i = 0

            p0, p1 = convex[i-1], convex[i]

            plt.plot((p0[0], p1[0]), (p0[1], p1[1]), 'g')

    plt.show()

def angle(p0, p1 = None):

    if (p1 == None):

        p1 = anchor

    y, x = p0[1] - p1[1], p0[0] - p1[0]

    return (m.atan2(y, x))

def distance(p0, p1 = None):

    if (p1 == None):

        p1 = anchor

    y, x = p0[1] - p1[1], p0[0] - p1[0]

    return (pow(y, 2) + pow(x, 2))

def find\_distance(p1, p2, p3):

    return ((p2[0] - p1[0]) \* (p3[1] - p1[1]) - (p2[1] - p1[1]) \* (p3[0] - p1[0]))

def sorting\_points(P):

    if (len(P) <= 1):

        return P

    small, eql, large =[], [], []

    pivot = angle(P[r.randint(0, len(P) - 1)])

    for p in P:

        ang = angle(p)

        if (ang < pivot):

            small.append(p)

        elif (ang > pivot):

            large.append(p)

        else:

            small.append(p)

    return (sorting\_points(small) + sorted(eql, key=distance) + sorting\_points(large))

def GscanAlgo(points, process = False):

    global anchor

    min\_index = 0

    for i, (x, y) in enumerate(points):

        if (min\_index == None or y < points[min\_index][1]):

            min\_index = i

        if (x < points[min\_index][0] < y == points[min\_index][1]):

            min\_index = i

    anchor = points[min\_index]

    sorted\_points = sorting\_points(points)

    del sorted\_points[sorted\_points.index(anchor)]

    hull = [anchor, sorted\_points[0]]

    for p in sorted\_points[1:]:

        while (find\_distance(hull[-2], hull[-1], p) <= 0):

            del hull[-1]

        hull.append(p)

        if (process):

            view\_graph(points, hull)

    return hull

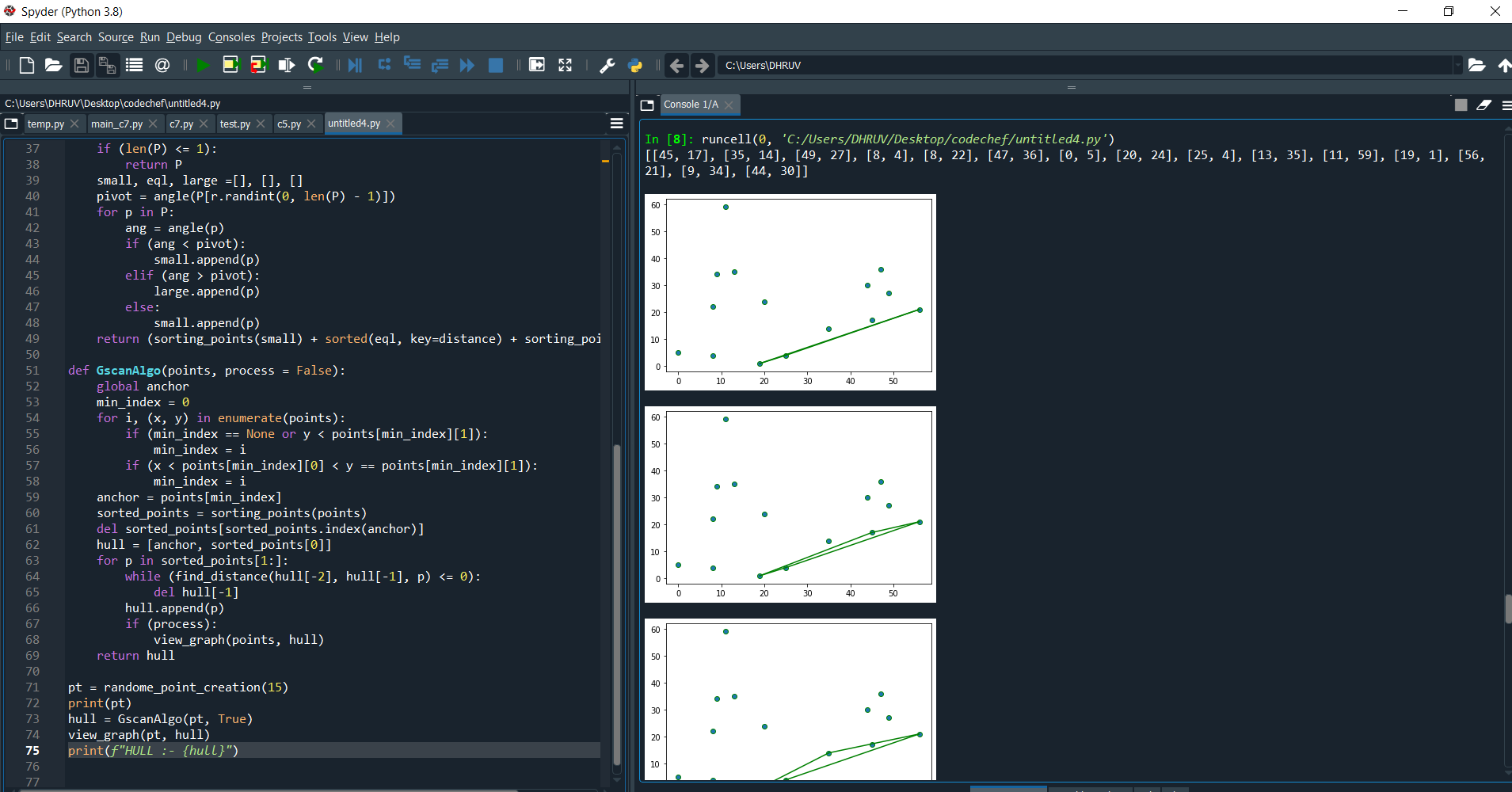
pt = randome\_point\_creation(15)

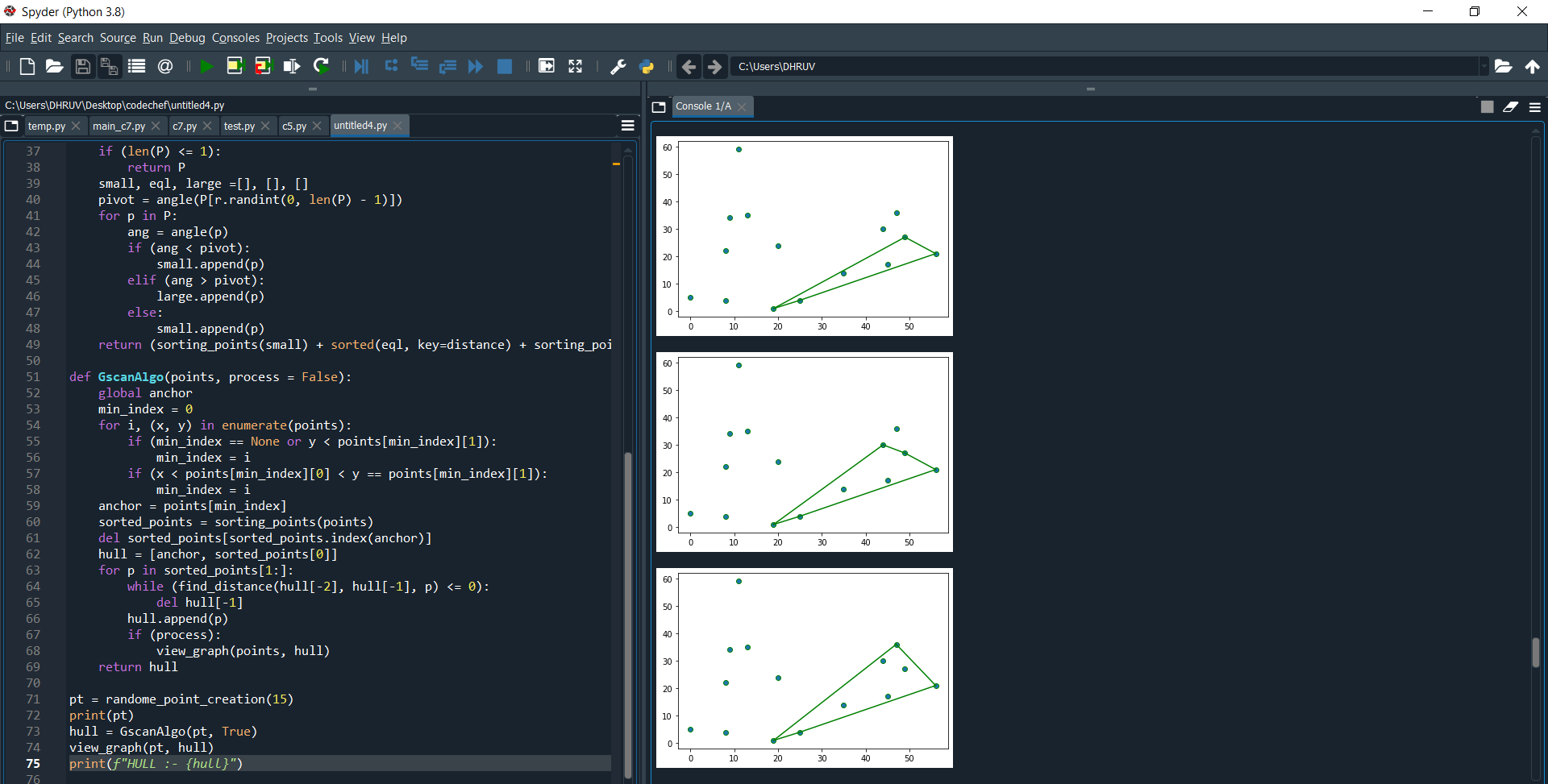
print(pt)

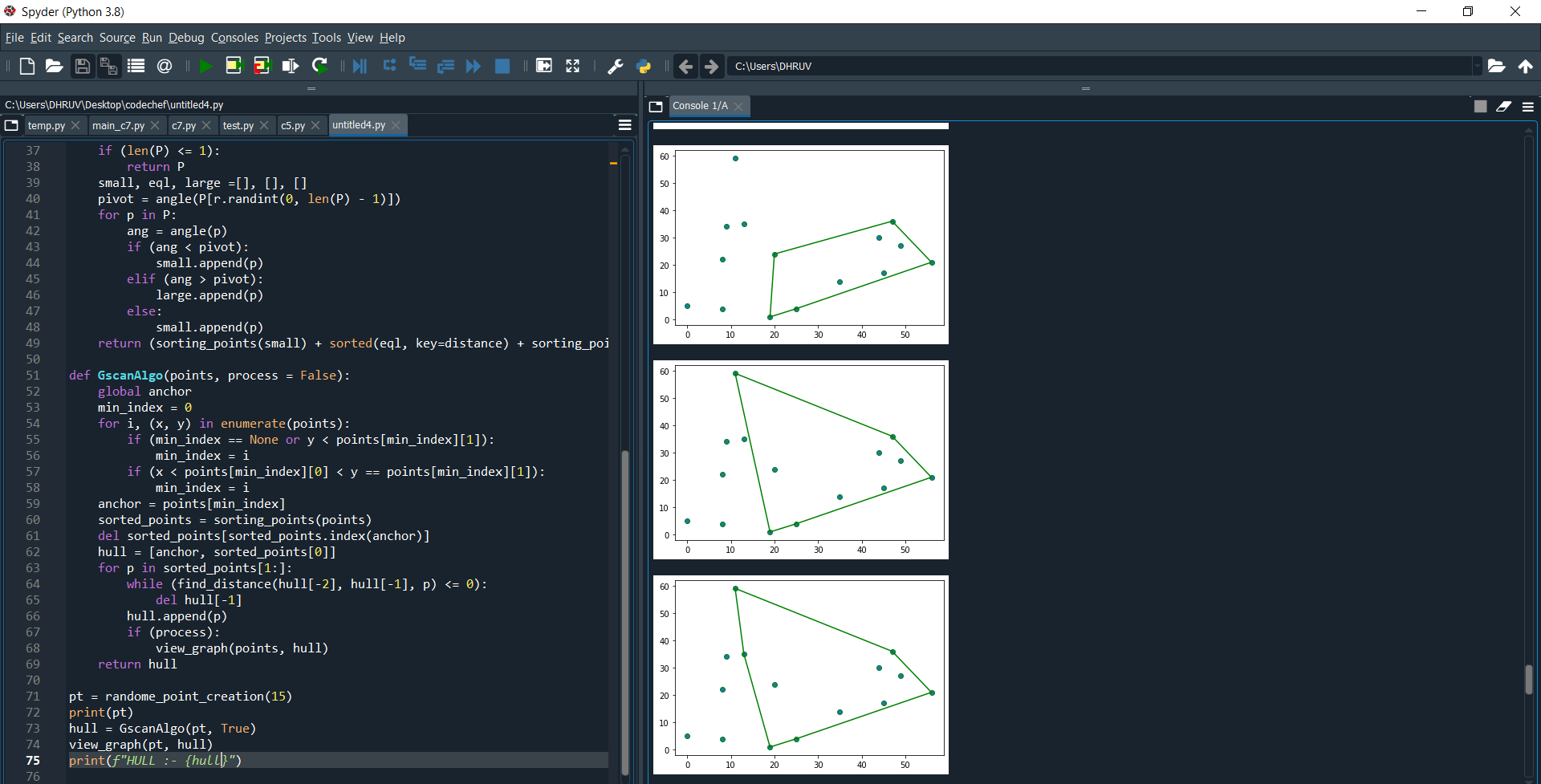
hull = GscanAlgo(pt, True)

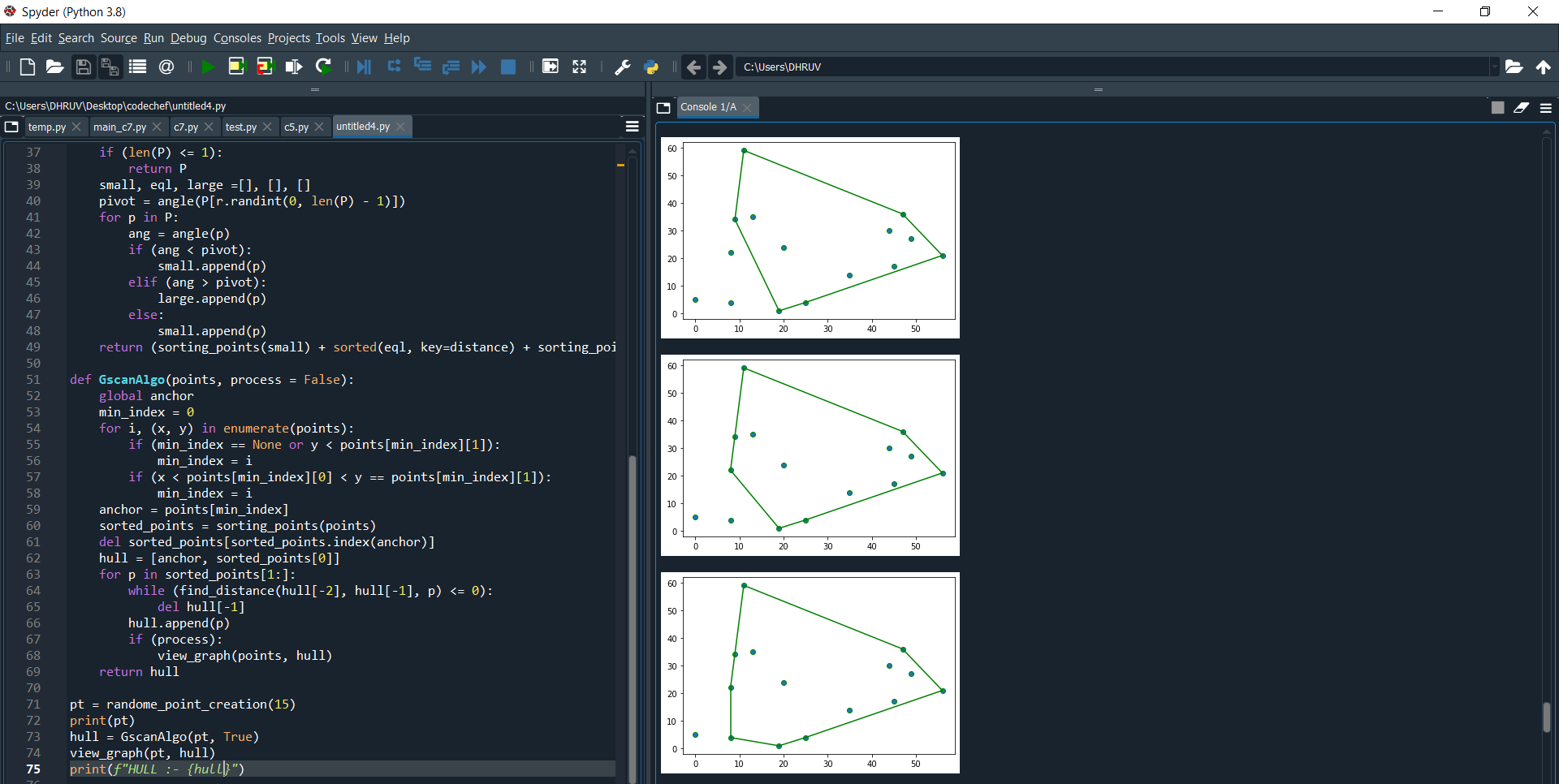
view\_graph(pt, hull)

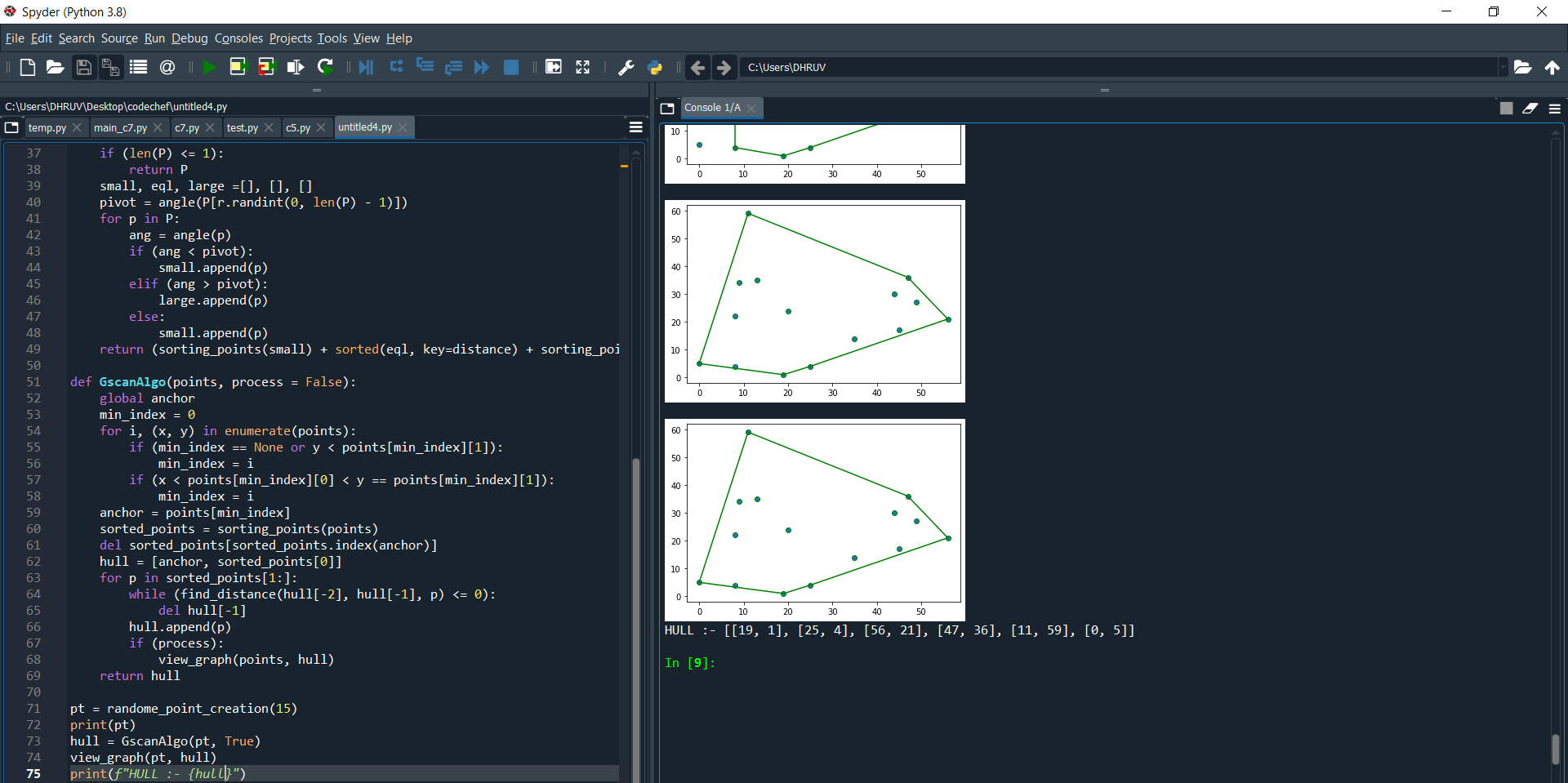
Outputs :-











Theory :-

The convex hull is the minimum closed area which can cover all given data points.

Graham’s Scan algorithm will find the corner points of the convex hull. In this algorithm, at first, the lowest point is chosen. That point is the starting point of the convex hull. Remaining n-1 vertices are sorted based on the anti-clockwise direction from the start point. If two or more points are forming the same angle, then remove all points of the same angle except the farthest point from start.

Using Graham’s scan algorithm, we can find Convex Hull in O(nLogn) time.

Jarvis’s Algorithm  for Convex Hull. The worst case time complexity of Jarvis’s Algorithm is O(n^2).