**Labsheet 7**

**Graham’s Scan**

from functools import cmp\_to\_key

class Point:

    def \_\_init\_\_(self, x = None, y = None):

        self.x = x

        self.y = y

p0 = Point(0, 0)

def nextToTop(S):

    return S[-2]

def distSq(p1, p2):

    return ((p1.x - p2.x) \* (p1.x - p2.x) + (p1.y - p2.y) \* (p1.y - p2.y))

def orientation(p, q, r):

    val = ((q.y - p.y) \* (r.x - q.x) - (q.x - p.x) \* (r.y - q.y))

    if val == 0:

        return 0

    elif val > 0:

        return 1

    else:

        return 2

def compare(p1, p2):

    o = orientation(p0, p1, p2)

    if o == 0:

        if distSq(p0, p2) >= distSq(p0, p1):

            return -1

        else:

            return 1

    else:

        if o == 2:

            return -1

        else:

            return 1

def convexHull(points, n):

    ymin = points[0].y

    min = 0

    for i in range(1, n):

        y = points[i].y

        # Pick the bottom-most or choose the left most point in case of tie

        if ((y < ymin) or (ymin == y and points[i].x < points[min].x)):

            ymin = points[i].y

            min = i

    points[0], points[min] = points[min], points[0]

    p0 = points[0]

    points = sorted(points, key=cmp\_to\_key(compare))

    for i in range(1, n):

        while ((i < n - 1) and

        (orientation(p0, points[i], points[i + 1]) == 0)):

            i += 1

        points[m] = points[i]

        m += 1

    if m < 3:

        return

    S = []

    S.append(points[0])

    S.append(points[1])

    S.append(points[2])

    for i in range(3, m):

        while ((len(S) > 1) and

        (orientation(nextToTop(S), S[-1], points[i]) != 2)):

            S.pop()

        S.append(points[i])

    while S:

        p = S[-1]

        print("(" + str(p.x) + ", " + str(p.y) + ")")

        S.pop()

input\_points = [(0, 3), (1, 1), (2, 2), (4, 4),

                (0, 0), (1, 2), (3, 1), (3, 3)]

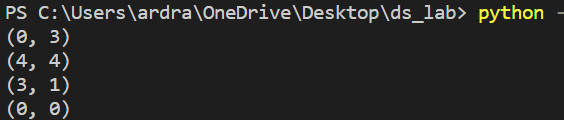
points = []

for point in input\_points:

    points.append(Point(point[0], point[1]))

n = len(points)

convexHull(points, n)

****

**Jarvis Scan**

def jarvis(points):

    n = len(points)

    start = points[0]

    for i in range(1, n):

        if points[i][0] < start[0]:

            start = points[i]

    current = start

    convex\_hull = [start]

    while True:

        next\_point = points[0]

        for i in range(1, n):

            angle = angle\_between(current, next\_point, points[i])

            if angle < 0 or (angle == 0 and distance(current, points[i]) > distance(current, next\_point)):

                next\_point = points[i]

        current = next\_point

        if current == start:

            break

        convex\_hull.append(current)

    return convex\_hull

def angle\_between(p1, p2, p3):

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

    return angle

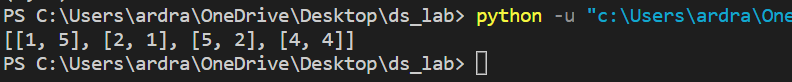
def distance(p1, p2):

    return ((p2[0] - p1[0]) \*\* 2 + (p2[1] - p1[1]) \*\* 2) \*\* 0.5

points = [[2, 3], [4, 4], [5, 2], [2, 1], [1, 5]]

convex\_hull = jarvis(points)

print(convex\_hull)

****