AA_Lab_08_Assignment

CE_054

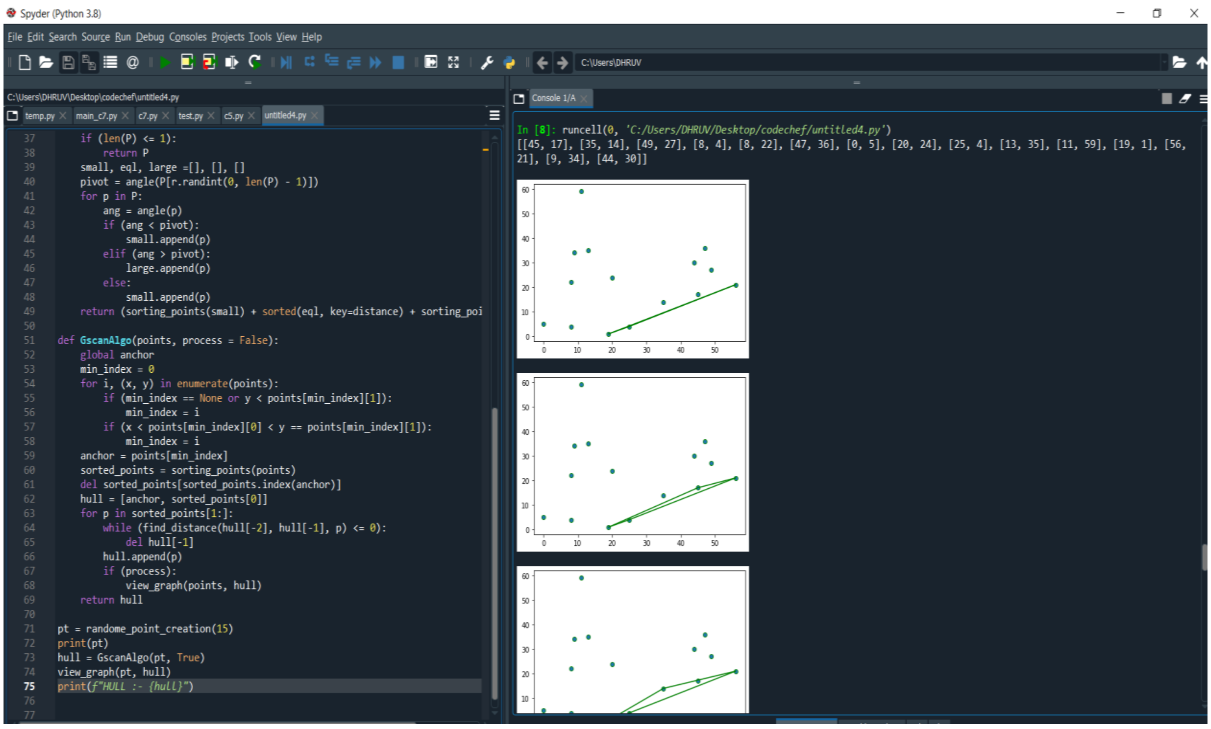
Aim :- Implementation of Convex-Hull Algorithm using Graham Scan Algorithm.

Code :

```python
'''
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_p
oints)]

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))
```

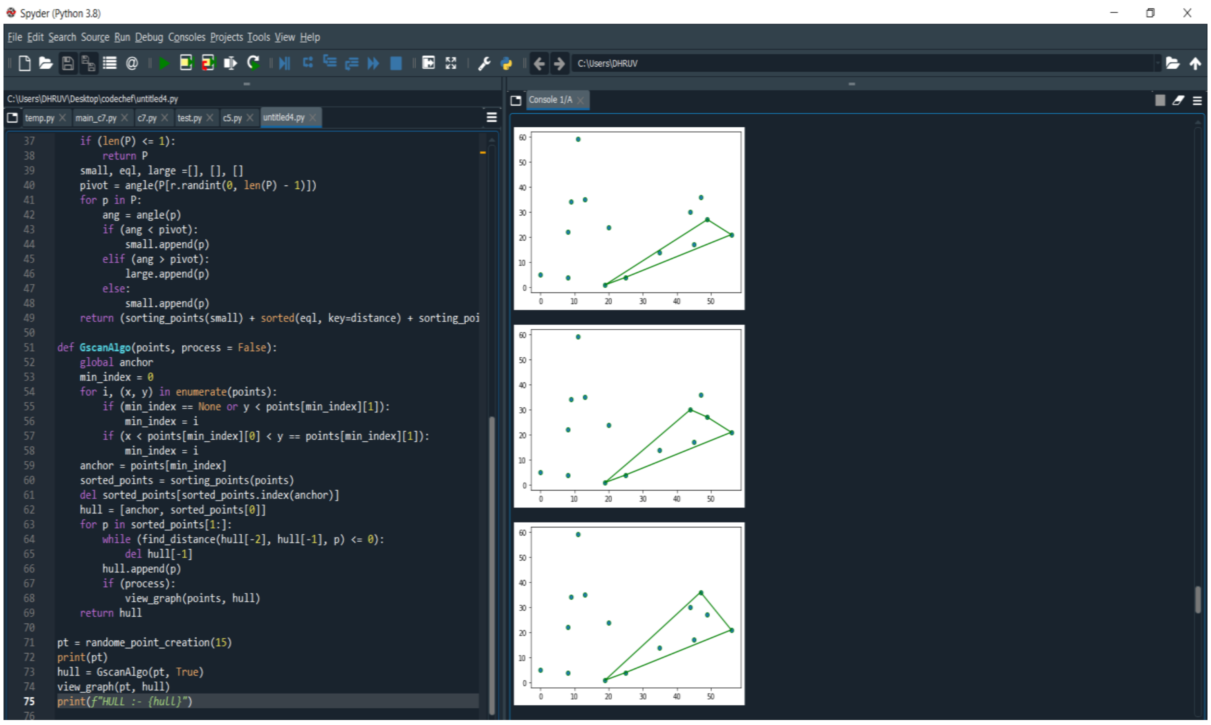
```
def find_distance(p1, p2, p3):
    return ((p2[0] - p1[0]) * (p3[1] - p1[1]) - (p2[1] - p1[1]) * (p3[0] - p1[
0]))

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 :-

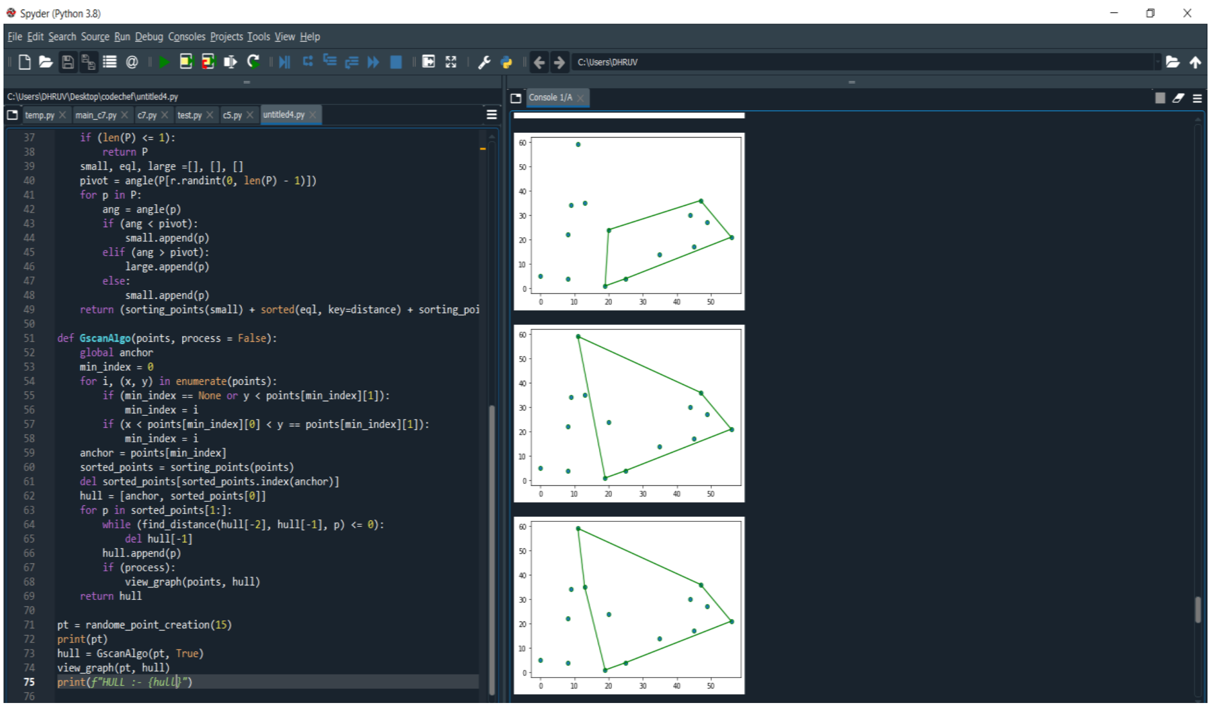

Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DHRUV
C:\Users\DHRUV\Desktop\codechef\untitled4.py
temp.py
main_c7.py
c7.py
test.py
c5.py
untitled4.py
Console 1/A

```python
    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_poi

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)
print(f"HULL :- {hull}")
```

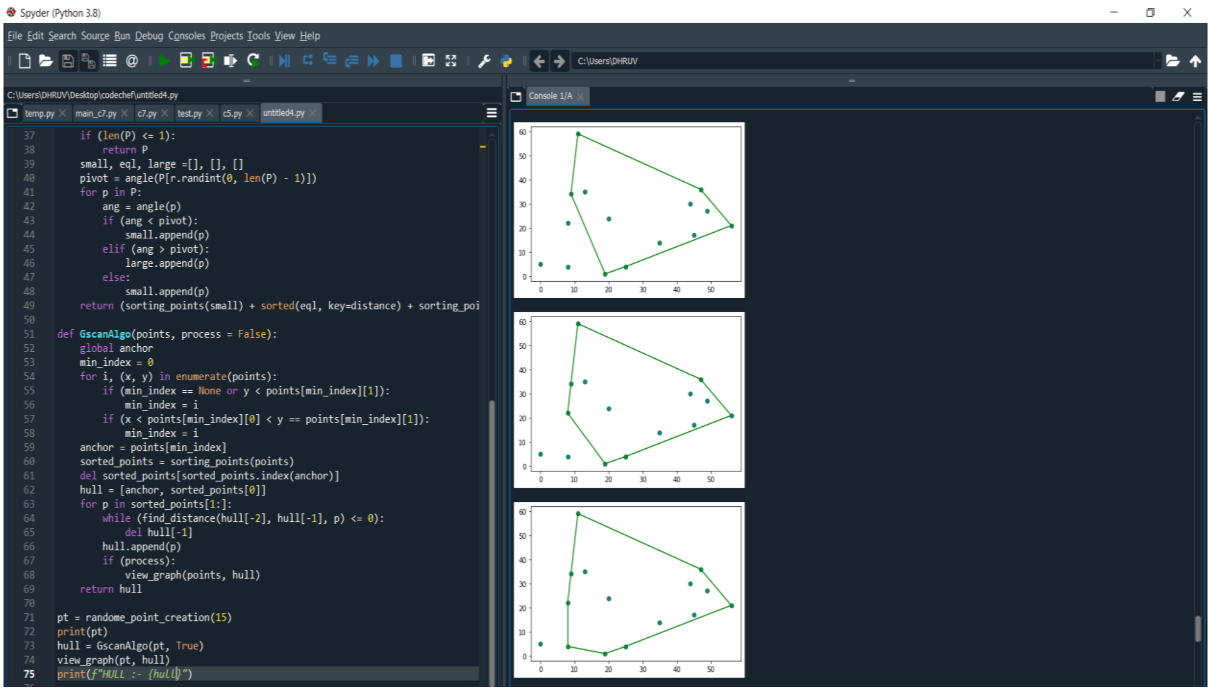
```
In [8]: runcell(0, 'C:/Users/DHRUV/Desktop/codechef/untitled4.py')
[[45, 17], [35, 14], [49, 27], [8, 4], [8, 22], [47, 36], [0, 5], [20, 24], [25, 4], [13, 35], [11, 59], [19, 1], [56,
21], [9, 34], [44, 30]]
```


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DHRUV
C:\Users\DHRUV\Desktop\codechef\untitled4.py
temp.py
main_c7.py
c7.py
test.py
c5.py
untitled4.py
Console 1/A

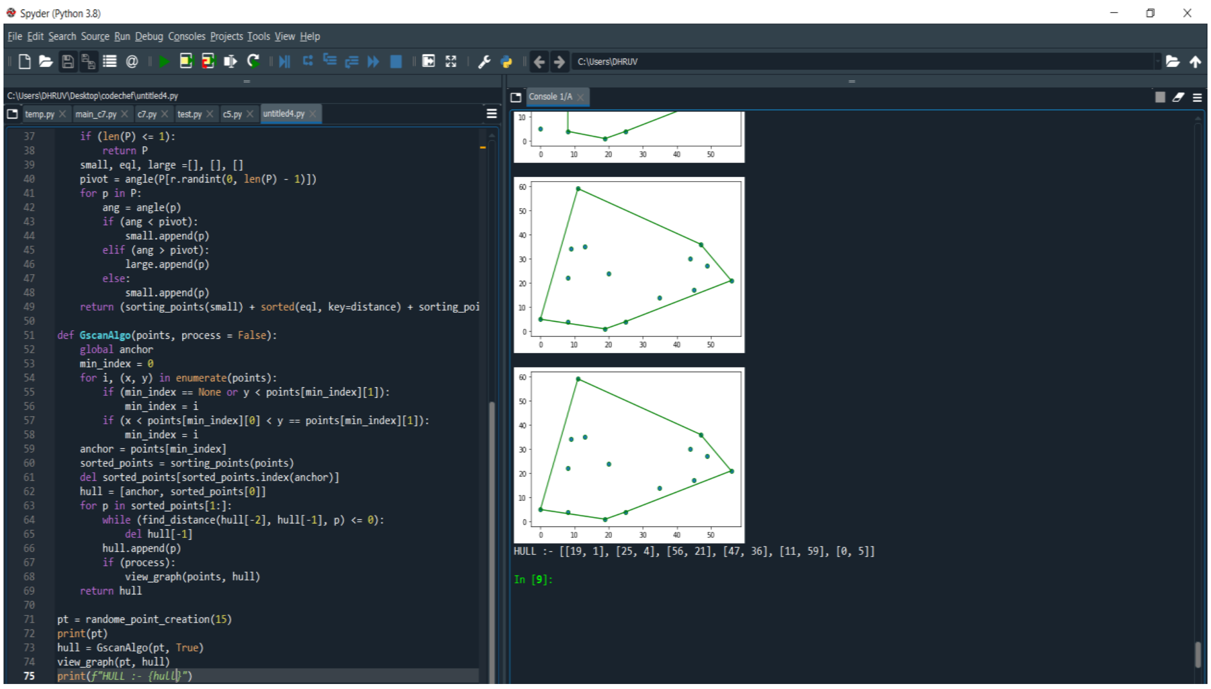
```python
    if (len(P) <= 1):
        return P
    small, eql, large =[], [], []
    pivot = angle(P[r.randint(0, len(P) - 1)])
    for p in P:
        ang = angle(p)
        if (ang < pivot):
            small.append(p)
        elif (ang > pivot):
            large.append(p)
        else:
            small.append(p)
    return (sorting_points(small) + sorted(eql, key=distance) + sorting_poi

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)
print(f"HULL :- {hull}")
```



```
    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_poi

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)
print(f"HULL :- {hull}")
```



```
    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_poi

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)
print(f"HULL :- {hull}")
```



Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\DHRUV
C:\Users\DHRUV\Desktop\codechef\untitled4.py
temp.py
main_c7.py
c7.py
test.py
c5.py
untitled4.py
Console 1/A

```python
    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_poi

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)
print(f"HULL :- {hull}")
```

```
HULL :- [[19, 1], [25, 4], [56, 21], [47, 36], [11, 59], [0, 5]]

In [9]:
```

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).