SSN College of Engineering

Department of Information Technology

UIT2201 — Programming and Data Structures

2022 – 2023

Exercise — 02

U. Pranaav | IT-B | 3122225002093

1.

I. AIM:

To define a class Point, a simple class to represent 2-dimensional points (Non-mutable). Each object has two fields: '_x' and '_y'. Methods include 'distance' that returns Euclidean distance between 'this' object and another object.

II. CODE:

```
# -*- coding: utf-8 -*-


'''
This module provides a class used for storing the coordinates of
points and the stored data can be used to find the distance
between two points. This is a part of the excercises given under
the course UIT2201 (Programming and Data Structures).

In this source code I have executed my own logic. The code
follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 12 2023

Revised on Wed Apr 14 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

'''


import random


class point:

    '''
    The given class stores the coordinates of a point and
    performs functions such as finding distance between
```

```
    two points adding two points as well as subtracting
    two points.

    The input data is not modified in any way and there are
    no side effects.

    methods:
        __init__: the constructor

        __add__: for using the '+' operation on class objects

        __sub__: for using the '-' operation on class objects

        __str__: for displaying class objects in human readable
        form.

        distance: for calculating distance between two point
        objects

    '''

    __slots__ = ('_x', '_y')  #setting the variables so that we can optimize memory usage


    def __init__(self,x_coord,y_coord):

        '''
        The constructor here takes in 3 arguments, and assigns
        the values of x coordinates and y coordinates to special
        variables _x and _y which are used for performing other
        functions.

        The input is not modified and there are no side effects.

        args:
            self: the object
            x_coord: value of x coordinate
            y_coord: value of y coordinate

        returns:
            variables _x and _y used in other functions in class.

        '''

        self._x = x_coord
        self._y = y_coord


    def __add__(self,other):

        '''
        This function takes in two points as input and calculates
        the sum of each point's x coordinates and each point's y
        coordinates and returns a point with the calculated x and
        y coordinates.
```

```
        The input is not modified and there are no side effects.

        args:
            self: first object
            other: second object

        Returns:
            The sum of coordinates of two points.

        '''

        return point(self._x+other._x,self._y+other._y)


    def __sub__(self,other):

        '''
        This function takes in two points as input and calculates
        the difference of each point's x coordinates and each point's y
        coordinates and returns a point with the calculated x and
        y coordinates.

        The input is not modified and there are no side effects.

        args:
            self: first object
            other: second object

        Returns:
            The difference of coordinates of two points.

        '''

        return point(self._x-other._x,self._y-other._y)


    def __str__(self):

        '''
        This function takes in only the object and returns the
        given object as a str data type.

        The input is not modified in any way and there are no side
        effects

        args:
            self: the object to be displayed

        Returns:
            An object of the str data type.

        '''

        return f'({self._x},{self._y})'
```

```
    def distance(self,other):

        '''
        This function takes in two points as input and calculates
        the distance between them.

        The input is not modified and there are no side effects.

        args:
            self: first object
            other: second object

        Returns:
            The distance between the two points.

        '''

        x_diff = (self._x-other._x)**2
        y_diff = (self._y-other._y)**2
        return (x_diff + y_diff)**0.5
#end of class point


#defining a function for generating a random number of point objects
def point_gen(point_count):

    '''
    The given function generates a random number of point
    objects and returns them as a list of point objects.

    args:
        point_count: the number of point objects to be
        generated

    Returns:
        A list of point objects.

    '''

    point_list = []
    for case in range(point_count):
        x_val = random.randint(-1000,1000)
        y_val = random.randint(-1000,1000)
        point_1 = point(x_val,y_val)  #creating a new object each iteration
        point_list.append(point_1)

    return point_list
#end of function point_gen


#driver code
if __name__ == '__main__':
    #this part of the code will only be run when the function is called directly
    #it will not be executed when it is imported as a module

    #initializing variable for storing distances
```

```
distances = []

#generating n number of objects based on user input and storing in a list
number_of_objects = int(input("Enter number of objects:"))
point_list = point_gen(number_of_objects)

input_point = tuple(eval(input("Enter coordinates of point in a tuple (x, y):")))
print()  #for spacing between lines

#setting the user input point as an object of class point
x_coord,y_coord = input_point
user_point = point(x_coord,y_coord)

for points in point_list:
    distances.append(user_point.distance(points))

print("Points and the distances of all the points generated and input point are:")
for index in range(len(point_list)):
    print("Point: ",(point_list[index]._x,point_list[index]._y)," Distance: ",distances[index])
print()

#testing the addition, subtraction operations on points
point_1 = point(5,10)
point_2 = point(6,12)

print(f"Point 1 is: {point_1} and Point 2 is: {point_2}")
print("Sum of both points is:",point_1 + point_2)
print("Difference between both points is:",point_1 - point_2)
print()
#end of code
```

III. OUTPUT:

Enter number of objects:10

Enter coordinates of point in a tuple (x, y):(5,6)

Points and the distances of all the points generated and input point are:

Point: (-465, 962) Distance: 1065.2868158388144

Point: (-459, -616) Distance: 776.0025773153076

Point: (-266, 702) Distance: 746.898252776106

Point: (-478, 239) Distance: 536.2629951805364

Point: (-982, 892) Distance: 1326.335176341184

Point: (382, 182) Distance: 416.0588900624526

Point: (-570, -433) Distance: 723.4265684919237

Point: (-432, 392) Distance: 583.0651764597162

Point: (-709, -330) Distance: 789.1083575783493

Point: (-254, -801) Distance: 847.5435092076394

Point 1 is: (5,10) and Point 2 is: (6,12)

Sum of both points is: (11,22)

Difference between both points is: (-1,-2)

2.

I. AIM:

To write a Python code to generate a random sequence of n Points and define a function that, given an integer k and a new Point Pnew, returns k-nearest neighbors of Pnew in the given sequence of n Points.

II. CODE:

```
# -*- coding: utf-8 -*-


'''
This module imports a class called point class used for
storing the coordinates of points and finding the distance
between two points. The module contains a function that can
find a specific number of points nearest to a user input point.
This is a part of the excercises given under the course UIT2201
(Programming and Data Structures).

In this source code I have executed my own logic. The code
follows good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 12 2023

Revised on Wed Apr 14 2023

Original Author: U. Pranaav <pranaav2210205@ssn.edu.in>

'''


from pointclass import point
from pointclass import point_gen
```

```python
from random import randint


#function for finding a set number of nearest points
def nearest(Pnew,k,point_list):

    '''
    This function returns the required number of points
    nearest to the given point by calculating distances
    between input point and all the points in point_list
    and returns the list containing the points.

    The input is not modified and there are no side effects.

    args:
        Pnew: the point object
        k: the number of nearest elements required
        point_list: list of all the points

    Returns:
        A list of points containing k number of nearest points.

    '''

    distance_dict = {}

    for points in point_list:
        distance_dict[(points._x,points._y)] = Pnew.distance(points)

    items =[x for x in distance_dict.items()]  #getting key value pairs of points and distances
    distance_val = [x[1] for x in items]  #getting a list of distances

    distance_val.sort()

    count = 0  #used for getting correct index and k number of iterations
    nearest_points = []

    if len(items) > k:
        while count < k:
            for coords in items:
                if coords[1] == distance_val[count]:  #we will add the coordinates in a ordered form
                    nearest_points.append(coords)
            count+=1

    else:
        while count < len(items):
            for coords in items:
                if coords[1] == distance_val[count]:
                    nearest_points.append(coords)
            count+=1

    return nearest_points
#end of function nearest


#driver code
```

```
if __name__ == '__main__':
    #this part of the code will only be run when the function is called directly
    #it will not be executed when it is imported as a module

    #initializing variable for storing distances
    distances = []

    #generating n number of objects based on user input and storing in a list
    number_of_objects = int(input("Enter number of objects:"))
    point_list = point_gen(number_of_objects)

    Pnew = tuple(eval(input("Enter coordinates of point in a tuple (x,y):")))
    k = int(input("Enter number of nearest points required:"))
    print()  #for spacing

    x_coord,y_coord = Pnew
    user_point = point(x_coord,y_coord)

    for points in point_list:
        distances.append(user_point.distance(points))

    print("Points and the distances of all the points generated and input point are:")
    for index in range(len(point_list)):
        print("Point: ",(point_list[index]._x,point_list[index]._y)," Distance: ",distances[index])
    print()

    print("k number of nearest points and their distances are:")
    for points,distance in nearest(user_point,k,point_list):
        print(f"Point {points}: Distance {distance}")
    print()

    #now we will test a case where the value of required nearest numbers is greater than total
number of objects

    test_k = 4
    test_num_objects = 3
    test_point_list = point_gen(test_num_objects)
    xcoord = randint(-1000,1000)
    ycoord = randint(-1000,1000)
    Pnew = (xcoord,ycoord)
    distances = []

    print(f"Value of k is: {test_k}\nNumber of objects generated is: {test_num_objects}")
    print(f"Randomly generated point is: {Pnew}")
    print()

    x_coord,y_coord = Pnew
    user_point = point(x_coord,y_coord)

    for points in test_point_list:
        distances.append(user_point.distance(points))

    print("Points and the distances of all the points generated and input point are:")
    for index in range(len(test_point_list)):
        print("Point: ",(test_point_list[index]._x,test_point_list[index]._y)," Distance:
",distances[index])
```

```
    print()

    print("k number of nearest points and their distances are:")
    for points,distance in nearest(user_point,test_k,test_point_list):
        print(f"Point {points}: Distance {distance}")
    print()

    #we will now test the code for a random number of points
    print("Now testing for a random number of test cases and random values of k")
    test_cases = randint(1,10)
    print(f"Number of test cases is: {test_cases}")

    for case in range(test_cases):
        distances = []

        #generating n number of objects based on randint and storing in a list
        number_of_objects = randint(4,10)
        point_list = point_gen(number_of_objects)
        Pnew = (randint(-1000,1000),randint(-1000,1000))
        k = randint(1,5)
        print()  #for spacing

        print(f"Value of k is: {k}\nNumber of objects generated is: {number_of_objects}")
        print(f"Randomly generated point is: {Pnew}")
        print()

        x_coord,y_coord = Pnew
        user_point = point(x_coord,y_coord)

        for points in point_list:
            distances.append(user_point.distance(points))

        print("Points and the distances of all the points generated and input point are:")
        for index in range(len(point_list)):
            print("Point: ",(point_list[index]._x,point_list[index]._y)," Distance:
",distances[index])
        print()

        print("k number of nearest points and their distances are:")
        for points,distance in nearest(user_point,k,point_list):
            print(f"Point {points}: Distance {distance}")
        print()
    #end of code
```

III. OUTPUT:

Enter number of objects:10

Enter coordinates of point in a tuple (x,y):(5,6)

Enter number of nearest points required:3

Points and the distances of all the points generated and input point are:

Point: (429, -258) Distance: 499.4717209212149

Point: (877, 568) Distance: 1037.4140928288955

Point: (-101, 680) Distance: 682.2843981801137

Point: (-678, 782) Distance: 1033.7625452684963

Point: (-494, 878) Distance: 1004.6815415841977

Point: (-517, -35) Distance: 523.607677560213

Point: (-767, -822) Distance: 1132.0636024535017

Point: (983, 981) Distance: 1380.9811729346638

Point: (489, -682) Distance: 841.1896337925236

Point: (594, -777) Distance: 979.8010002036128

k number of nearest points and their distances are:

Point (429, -258): Distance 499.4717209212149

Point (-517, -35): Distance 523.607677560213

Point (-101, 680): Distance 682.2843981801137

Value of k is: 4

Number of objects generated is: 3

Randomly generated point is: (-134, 100)

Points and the distances of all the points generated and input point are:

Point: (-850, -169) Distance: 764.8640402058395

Point: (-924, -484) Distance: 982.4235339200706

Point: (-624, -337) Distance: 656.5584513202157

k number of nearest points and their distances are:

Point (-624, -337): Distance 656.5584513202157

Point (-850, -169): Distance 764.8640402058395

Point (-924, -484): Distance 982.4235339200706

Now testing for a random number of test cases and random values of k

Number of test cases is: 7

Value of k is: 5

Number of objects generated is: 6

Randomly generated point is: (146, -484)

Points and the distances of all the points generated and input point are:

Point: (-883, -279) Distance: 1049.2216162470158

Point: (976, -107) Distance: 911.607920106007

Point: (-879, -806) Distance: 1074.3877326179781

Point: (-159, -274) Distance: 370.3039292257105

Point: (-588, 999) Distance: 1654.703901004648

Point: (204, -154) Distance: 335.0582038989644

k number of nearest points and their distances are:

Point (204, -154): Distance 335.0582038989644

Point (-159, -274): Distance 370.3039292257105

Point (976, -107): Distance 911.607920106007

Point (-883, -279): Distance 1049.2216162470158

Point (-879, -806): Distance 1074.3877326179781

Value of k is: 4

Number of objects generated is: 8

Randomly generated point is: (652, 549)

Points and the distances of all the points generated and input point are:

Point: (382, -384) Distance: 971.2821423252875

Point: (602, -418) Distance: 968.2917948635112

Point: (409, 270) Distance: 369.98648623970035

Point: (-570, 877) Distance: 1265.2541246721942

Point: (-700, 906) Distance: 1398.3393722555336

Point: (-426, -525) Distance: 1521.696421760924

Point: (458, 929) Distance: 426.65677071857186

Point: (-265, -222) Distance: 1198.0525864919287

k number of nearest points and their distances are:

Point (409, 270): Distance 369.98648623970035

Point (458, 929): Distance 426.65677071857186

Point (602, -418): Distance 968.2917948635112

Point (382, -384): Distance 971.2821423252875

Value of k is: 4

Number of objects generated is: 7

Randomly generated point is: (481, -283)

Points and the distances of all the points generated and input point are:

Point: (-716, 345) Distance: 1351.737030638726

Point: (-341, -950) Distance: 1058.5712068632888

Point: (346, -383) Distance: 168.00297616411441

Point: (-768, -338) Distance: 1250.2103822957158

Point: (-361, 657) Distance: 1261.968303880886

Point: (388, -732) Distance: 458.53026072441503

Point: (352, 294) Distance: 591.2444502910788

k number of nearest points and their distances are:

Point (346, -383): Distance 168.00297616411441

Point (388, -732): Distance 458.53026072441503

Point (352, 294): Distance 591.2444502910788

Point (-341, -950): Distance 1058.5712068632888

Value of k is: 5

Number of objects generated is: 6

Randomly generated point is: (959, -160)

Points and the distances of all the points generated and input point are:

Point: (951, -482) Distance: 322.0993635510632

Point: (-718, 633) Distance: 1855.0412394337761

Point: (999, 928) Distance: 1088.7350458215258

Point: (-929, -136) Distance: 1888.1525362109917

Point: (120, 711) Distance: 1209.3642958182618

Point: (-202, -730) Distance: 1293.3758154534976

k number of nearest points and their distances are:

Point (951, -482): Distance 322.0993635510632

Point (999, 928): Distance 1088.7350458215258

Point (120, 711): Distance 1209.3642958182618

Point (-202, -730): Distance 1293.3758154534976

Point (-718, 633): Distance 1855.0412394337761

Value of k is: 5

Number of objects generated is: 10

Randomly generated point is: (564, -719)

Points and the distances of all the points generated and input point are:

Point: (-545, -335) Distance: 1173.6000170415814

Point: (-365, 128) Distance: 1257.1594966431269

Point: (780, -888) Distance: 274.2571785751469

Point: (-897, -259) Distance: 1531.7052588536737

Point: (634, 173) Distance: 894.7424210352385

Point: (-758, -62) Distance: 1476.2564140419508

Point: (-362, 710) Distance: 1702.7968170043073

Point: (12, -419) Distance: 628.2547254100044

Point: (891, 808) Distance: 1561.6203123678945

Point: (889, 630) Distance: 1387.5972038023137

k number of nearest points and their distances are:

Point (780, -888): Distance 274.2571785751469

Point (12, -419): Distance 628.2547254100044

Point (634, 173): Distance 894.7424210352385

Point (-545, -335): Distance 1173.6000170415814

Point (-365, 128): Distance 1257.1594966431269

Value of k is: 4

Number of objects generated is: 6

Randomly generated point is: (-444, -174)

Points and the distances of all the points generated and input point are:

Point: (360, 238) Distance: 903.4157403986273

Point: (807, -593) Distance: 1319.3036041791138

Point: (973, 182) Distance: 1461.0355916267065

Point: (-867, -977) Distance: 907.600132216826

Point: (146, -101) Distance: 594.4989486954539

Point: (-151, 210) Distance: 483.01656286301403

k number of nearest points and their distances are:

Point (-151, 210): Distance 483.01656286301403

Point (146, -101): Distance 594.4989486954539

Point (360, 238): Distance 903.4157403986273

Point (-867, -977): Distance 907.600132216826

Value of k is: 5

Number of objects generated is: 9

Randomly generated point is: (-811, 285)

Points and the distances of all the points generated and input point are:

Point: (384, -372) Distance: 1363.6986470624659

Point: (-748, -899) Distance: 1185.6749132877865

Point: (-450, 855) Distance: 674.7006743734587

Point: (-393, 492) Distance: 466.44721030358835

Point: (165, 267) Distance: 976.165969494942

Point: (467, -539) Distance: 1520.6117190131083

Point: (-497, -52) Distance: 460.61372102880307

Point: (893, -838) Distance: 2040.770687754996

Point: (-271, 745) Distance: 709.365914038728

k number of nearest points and their distances are:

Point (-497, -52): Distance 460.61372102880307

Point (-393, 492): Distance 466.44721030358835

Point (-450, 855): Distance 674.7006743734587

Point (-271, 745): Distance 709.365914038728

Point (165, 267): Distance 976.165969494942