EX - 2 POINT CLASS

T. SADAKOPA RAMAKRISHNAN | IT - B | 3122225002109

Q1. 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.

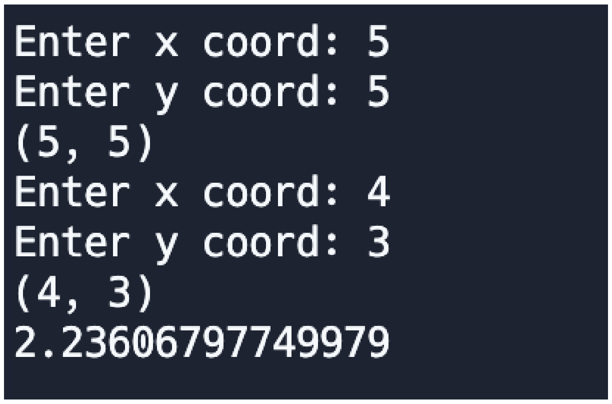
Aim:

To create a Point class in python that has two objects, _x and _y and returns Euclidean distance between ‘this’ object and another object.

Coding:

```
# -*- coding: utf-8 -*-
'''This module returns the distance between
two points given by user.This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 12 2023

Revised on Wed Apr 14 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
'''


class Point:
    '''
    '''
    _x = 0
    _y = 0
    def __init__(self, a, b):
        '''Constructor to initialize the datamembers'''
        self._x = a
        self._y = b
        return

    def distance(self, other):
        '''User defined method that returns the distance
between two points'''
        x_diff = (self._x - other._x) ** 2
        y_diff = (self._y - other._y) ** 2
        dist = (x_diff + y_diff) ** 0.5
        return dist
#End of class Point

def user_input():
    '''
    This function takes in x coordinate and y coordinate
    as input from user and creates two point objects of
    class point and returns the distance between two points
    using a user defined distance() method.
    '''
    n = 2
    points_lst = []
    for i in range(n):
        x_coord = int(input("Enter x coord: "))
        y_coord = int(input("Enter y coord: "))
        p = Point(x_coord, y_coord)
        print((p._x, p._y))
        points_lst.append(p)

    p1 = Point(points_lst[0]._x, points_lst[0]._y)
    p2 = Point(points_lst[1]._x, points_lst[1]._y)
    d = p1.distance(p2)
    return d
#End of function user_input()


#Test cases for this source code:

if __name__ == "__main__":
    #This part of the program will not be executed when the
file is imported.

    #Calculating distance between two points
    print(user_input())
    print()
```

Output of the following code:



```
Enter x coord: 5
Enter y coord: 5
(5, 5)
Enter x coord: 4
Enter y coord: 3
(4, 3)
2.23606797749979
```

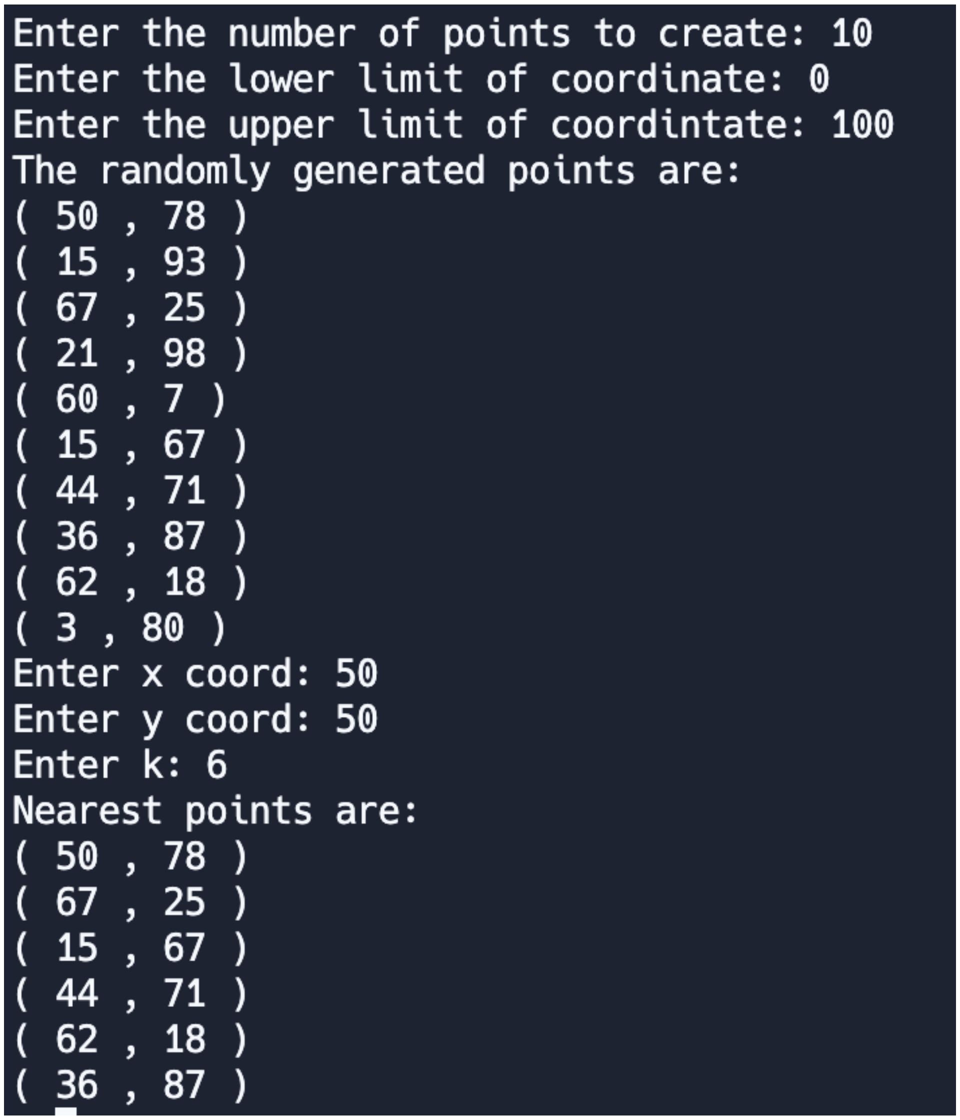
Q2. . Write a Python code to generate a random sequence of n Points. Define a function that, given an integer k and a new Point Pnew, returns k-nearest neighbors of Pnew in the given sequence of n Points.

Aim:

To create a Point class and generate a random sequence of n points and get integer k and a new point Pnew and return k - nearest neighbors of Pnew in the given sequence of n Points.

Coding:
```
# -*- coding: utf-8 -*-
'''This module generates a random sequence of n Points.
Using a
function that when given an integer k and a new Point PNew,
k nearest
neighbors of PNew is found.This is a part
of the exercises given under the course UIT2201 (Programming
and Data Structures).

In this source code I've executed my own logic and may
contain bugs.
The source code has followed good coding practices.

Your comments and suggestions are welcome.

Created on Wed Apr 12 2023

Revised on Wed Apr 14 2023

Original Author: T. Sadakopa Ramakrishnan
<sadakopa2210221@ssn.edu.in>
'''


class Point:
    _x = 0
    _y = 0
    def __init__(self, a, b):
        '''Constructor to initialize the datamembers'''
        self._x = a
        self._y = b
        return

    def distance(self, other):
        '''User defined method that returns the distance
between two points'''
        x_diff = (self._x - other._x) ** 2
        y_diff = (self._y - other._y) ** 2
        dist = (x_diff + y_diff) ** 0.5
        return dist

    def pNew_distance(self,points):
        '''
        User defined method that returns distance between
the
        pnew point and randomly generated n - number of
points.
        '''
        distances = []
        for i in range(len(points)):
            distances.append(self.distance(points[i]))
        return distances
#End of class

#Importing random module to create random x and y
coordinates.
import random


def nPoints():
    '''
    This function takes in number of points to generate and
    lower limit and upper limit of each points as an input
    and returns a list of all the point object created
    using class Point.

    n - number of points to create
    a - lower limit of coordinate
    b - upper limit of coordinate

    Returns: List of point objects
    '''
    n = int(input("Enter the number of points to create: "))
    a = int(input("Enter the lower limit of coordinate: "))
    b = int(input("Enter the upper limit of coordinate: "))
    return [Point((random.randint(a,b)),
(random.randint(a,b))) for i in range(n)]
#End of function nPoints

def knn(k, distances):
    '''
    This function takes in k value as an input and the
distances
    list as input and sorts the distances list in ascending
order
    and returns k number of points closest to Pnew point.
    '''
    for i in range(len(distances)-1):
        for j in range(i, len(distances)-1):
            if distances[j] > distances[j+1]:
                distances[j], distances[j+1] =
distances[j+1], distances[j]
                points[j], points[j+1] = points[j+1],
points[j]

    print("Nearest points are: ")
    for i in range(k):
        print('(', points[i]._x, ',', points[i]._y, ')' )
#End of function knn

#Test cases for this source code:

if __name__ == "__main__":
    #This part of the program will not be executed when the
file is imported.

    #Generating random n - number of points
    points = nPoints()
    print("The randomly generated points are: ")

    #Printing the points
    for point in points:
        print("(", point._x, ',', point._y, ")")

    #Finding the k - nearest neighbors to the Pnew point
from
    #the randomly generated set of points.
    pNew = Point((int(input("Enter x coord: "))),
(int(input("Enter y coord: "))))
    knn(int(input("Enter k: ")), pNew.pNew_distance(points))
    print
```

Output for the following code:



```
Enter the number of points to create: 10
Enter the lower limit of coordinate: 0
Enter the upper limit of coordintate: 100
The randomly generated points are:
( 50 , 78 )
( 15 , 93 )
( 67 , 25 )
( 21 , 98 )
( 60 , 7 )
( 15 , 67 )
( 44 , 71 )
( 36 , 87 )
( 62 , 18 )
( 3 , 80 )
Enter x coord: 50
Enter y coord: 50
Enter k: 6
Nearest points are:
( 50 , 78 )
( 67 , 25 )
( 15 , 67 )
( 44 , 71 )
( 62 , 18 )
( 36 , 87 )
```