Point-in-Polygon Test

Report for the CEGE0096: 1st Assignment

Pratibha Patel

21152395

pratibha.patel.21@ucl.ac.uk

1. Introduction

1.1 About the Software

The software developed is used for the Point-in-Polygon(PiP) test i.e. to check whether a point is inside, outside, or on the boundary of a polygon. The algorithm is run on the polygon(G-shaped) which is created by reading the Comma-Separated-Values(CSV) file having coordinates. The test points can be inputted in two ways. By a CSV file or by user-input. To begin with, firstly the test points are run through Minimum Bounding Rectangle(MBR) which gives surety about the points that are definitely outside the MBR and hence outside the polygon. MBR is defined as the smallest rectangle with sides parallel to axes of the cartesian frame. For MBR, X and Y coordinates of each point defining polygon is listed, and then a maximum of X and Y coordinates (making the top-right corner) and Minimum of X and Y coordinates (making the bottom-left corner) is taken. The MBR class gives two lists as output, one which determines the category with respect to MBR and another returns outside category and positions of other categories.

After the MBR, a more sophisticated algorithm is needed to determine the category of points inside and on the boundary of MBR. For this RCA, an example of the PiP algorithm is used. RCA begins with testing whether the test point is on the boundary, and if it is not on the boundary, it runs a counting number method to check whether the point is inside or outside the polygon. It provides results after categorizing all the test points and hence solving the PiP problem and the results are stored in a CSV file containing the test point's id and category. In the end, the outcome can be visualized by plotting the test points with their respective category and polygon.

1.2 Task Completion

S.no.	Task	Status
1.	Successful implementation of MBR	✅✅✅✅✅
2.	Successful implementation of RCA	✅✅✅✅✅
3.	Successful categorization of special cases.	✅✅✅✅✅
4.	Report writing	✅✅✅✅✅
5.	Object-Oriented Code	✅✅✅✅✅

6.	Commits on the GitHub repository	✅✅✅✅✅
7.	Add Comment to code	✅✅✅✅✅
8.	Apply PEP8 style	✅✅✅✅✅
9.	Plotting of points, polygons, and rays with ppropriate axis labels and annotations.	✅✅✅✅❌**(Ray)**
10.	Error handling functionality	✅✅✅✅✅
11.	Creativity	✅✅✅❌❌

1.3 Limitations of software

The points to form the polygon should be entered in a clockwise-manner and the first and last coordinate of the polygon should be the same(as closed figure). Additionally, the test points and polygon points should be inputted in a particular format(i.e. 3 columns- id, x, y). This results in lesser feasibility. And if a user wants to enter a point through the terminal, only 1 point can be checked at a time. On top of that, different classes and functions are created in different files, this results in importing things again and again and increasing the code, and making it bulky. Graphical User Interface(GUI) is used just to display the choices available to users to input but not further developed to provide more GUI. The classes MBR and RCA are quite heavy and repeated throughout the program due to limited use of OOP(inheritance).

1.4 Development of Software

It took some time to figure out how to read the CSV file without using any module. I was able to implement the MBR part quite easily in comparison to the RCA part. I began approaching the problem, making functions but then thought to reorganize them into classes provided in the lecture. However, that was not enough to get marks for the OOP part, so later converted MBR and RCA functions into classes. And I also used Point and Polygon classes to create point and polygon objects. I researched extensively, watched videos to understand the math involved to solve the problem. After this, I started with the pseudo-code and decided which operators, loops, functions to use to solve the problem. I tested the code and used solutions available online to debug it. I could not understand how to use the debugger provided in PyCharm.

1.5 Time

It took around 2 weeks, working 7-8 hours daily to develop this software.

2. Project Description

2.1 Library requirements: The software requires matplotlib library and Tkinter library(for the creativity part) to run the program.

2.2 File locations: All the files should be held in the same directory because main_from_file and main_from_user imports classes, functions from all the other files to execute the software and solve the problem.

2.3 File required to execute PiP: Two files main_from_file and main_from_user are required to be executed.

3. Software

Task 1. The MBR Algorithm

To run the MBR algorithm, class MinBoundReact was created. This class takes 4 arguments as input and contains 3 methods: mbr_func(), points_in_mbr() and points_outside_mbr().

```
class MinBoundRect:
      def __init__(self, pt_x_list, pt_y_list, poly_x_list, poly_y_list):
      self.__pt_x_list = pt_x_list
      self.__pt_y_list = pt_y_list
      self.__poly_x_list = poly_x_list
      self.__poly_y_list = poly_y_list
      self.__mbr_category = []
      self.__outside_mbr = []
```

Here, the 4 arguments are the list of X and Y coordinates of the test points and polygon. The two lists mbr_category and outside_mbr can be retrieved in the end.

Main method:

```
def mbr_func(self):
for i in range(len(self.__pt_x_list)):
      if (self.__pt_x_list[i] > max(self.__poly_x_list) or
self.__pt_y_list[i]
           > max(self.__poly_y_list) or       self.__pt_x_list[i] <
           min(self.__poly_x_list) or            self.__pt_y_list[i] <
```

```
            min(self.__poly_y_list)):
        self.__mbr_category.append('outside')
        self.__outside_mbr.append('outside')
```

Here, FOR loop is used to iterate through every test point. This iteration selects X and Y coordinates which are equated against the bounding rectangle defined by the maximum of X and Y coordinate(Top-right corner) and a minimum of X and Y coordinate(Left-bottom) of the polygon. If X and Y of the test-point are less than minimum X and Y of a polygon or greater than the maximum of X and Y of the polygon, then the test point is outside.

```
                elif   min(self.__poly_x_list)   <   self.__pt_x_list[i]   <
max(self.__poly_x_list)
         and                                  min(self.__poly_y_list) <
         self.__pt_y_list[i] < max(self.__poly_y_list):
         self.__mbr_category.append('inside')
         self.__outside_mbr.append(([i + 1]))
```

The second condition is exactly opposite to the first one, i.e. if the X and Y of the test-point are greater than minimum X and Y of a polygon or less than the maximum of X and Y of the polygon, then the test point is inside.

```
    else:
        self.__mbr_category.append('boundary')
        self.__outside_mbr.append(i + 1)
```

The last condition is applied when the above conditions hold False and then the test-point is considered to be on the boundary of the MBR.

The method points_in_mbr() returns the mbr_category which stores the category of test points w.r.t MBR. Similarly, the method points_outside_mbr () returns outside_mbr which stores the outside category and positions of other categories.

```
    def points_in_mbr(self):
        return self.__mbr_category

    def points_outside_mbr(self):
        return self.__outside_mbr
```

Task 2. The RCA Algorithm

To run the RCA Algorithm, class Rca was created. It contains 3 methods. This class takes 6 arguments as input.

```
from mbr import MinBoundRect
```

```
instance = MinBoundRect(pt_x_list, pt_y_list, poly_x_list, poly_y_list)
instance.mbr_func()
outside_list = instance.points_in_mbr()
mbr_out = instance.points_outside_mbr()
```

Class MinBoundReact is imported as the class Rca takes input from MinBoundReact class to further continue the process to categorize points inside the MBR and on the boundary of MBR. Here, the object is created and the method of class MinBoundReact is accessed.

```
class Rca:
    def __init__(self, pt_x_list, pt_y_list, poly_x_list, poly_y_list,
mbr_out,                              outside_list):
       self.__pt_x_list = pt_x_list
       self.__pt_y_list = pt_y_list
       self.__poly_x_list = poly_x_list
       self.__poly_y_list = poly_y_list
       self.__mbr_out = mbr_out
       self.__outside_list = outside_list
```

Here, the 4 arguments are the list of X and Y coordinates of the test points and polygon and 2 arguments mbr_out and outside_list are imported from MinBoundReact class.

Main method:

```
def rca(self):
   append = True
   cycle = False
```

Here, append and cycle are Boolean variables. It is used to control the end of cycle counting number method. Here, again the FOR loop is used to iterate through a list of the test points. The first condition is passed, because the point is outside MBR and therefore, not of our interest.

```
   for items in range(len(self.__pt_x_list)):
         xs = self.__pt_x_list[items]
         ys = self.__pt_y_list[items]
         n = len(self.__poly_x_list) - 1
         if self.__mbr_out[items] == 'outside':
           pass
      else:
      crosses = 0
```

Crosses is an integer type and is used to increment results of each crossing from the counting method. Here, FOR loop iterates over polygon coordinates.

```
for vertices in range(len(self.__poly_x_list)):
    ver_x1 = self.__poly_x_list[vertices]
    ver_y1 = self.__poly_y_list[vertices]
    if vertices == n:
       cycle = True

    if vertices < len(self.__poly_x_list) - 1:
        ver_x2 = self.__poly_x_list[vertices + 1]
        ver_y2 = self.__poly_y_list[vertices + 1]
```

Iterating through all the boundary points, one boundary at a time, creates a line segment for each vertex 1 with an X1 and Y1 and vertex 2 with X2 and Y2.

Cross Product method

To solve the boundary problem, the cross product was used.

```
on_boundary = False
```

on boundary is Boolean type and is used to store boundary points

```
if ((xs - ver_x1) * (ver_y2 - ver_y1) == (ver_x2 - ver_x1) * (ys - ver_y1)
and
      min(ver_x1, ver_x2) <= xs <= max(ver_x1, ver_x2) and
      min(ver_y1, ver_y2) <= ys <= max(ver_y1, ver_y2)):
    on_boundary = True
if on_boundary:
   category = 'boundary'
```

category is a string type and stores the category of point i.e. whether it is inside, outside, or on the boundary of the polygon.

```
   append = True
   break
```

Cross product method is used to find whether a point is on one of the lines or on the other side.

Let's consider Polygon Vertex 1 (X1,Y1), polygon vertex 2 (X2,Y2) and test point (X,Y).

D = (X-X1)(Y2-Y1) - (Y-Y1)(X2-X1)

where,

D>0, on one side

D<0, on the opposite side.

D=0, on line as points are collinear.

Source:https://blog.csdn.net/zhouzi2018/article/details/81735132?utm_medium=distribute.pc_relevant.none-task-blog-title-2&spm=1001.2101.3001.4242

To solve the boundary problem, the above equation of cross product was rearranged and an equality operation was used instead. If the outcome is true, then the point is on line. Along with this, an MBR is carried out to check if the point is within the boundary line. If it is true, it will append the list and break the loop and the variables will be updated.

Counting Number Method:

Now, if the point is not on the boundary, then it is either inside or outside the polygon. To check this, a counting number method is used. This method counts the number of times the ray from the test point crosses the polygon boundary.

Even number of crosses: Outside Polygon

Odd number of crosses: Inside Polygon

```
if (ver_y1 <= ys < ver_y2) or (ver_y1 > ys >= ver_y2):
   x_intersect = (ys - ver_y1) / (ver_y2 - ver_y1)
  if xs < (ver_x1 + x_intersect * (ver_x2 - ver_x1)):
  crosses += 1
```

Source: https://github.com/sasamil/PointInPolygon_Py/blob/master/pointInside.py#L50

Here, it first calculates if the ray is within the Y-coordinate of both the vertex points. If this condition holds true, then it further calculates the X-intersect between the ray of the test point and the edge. If the test point X is less than the intersection, then it has crossed the boundary, and the cross variable will be increased. The opposite of this means, it doesn't cross the boundary.

```
      if cycle:
         cycle = False
         append = True
      if crosses % 2 != 0:
          category = 'inside'
      else:
          category = 'outside'
if append:
   self.__outside_list[items] = category
   append = False
   cycle = False
```

Now, after the ray of one of the test points is checked against all the boundaries, if the cycle is not broken by the boundary condition mentioned above, the variable cycle will hold true and it will further dependent on whether it is odd or even to allocate its the position of inside or outside to the category variable. And finally outside_list will have the final outcome.

```
def get_outside_list(self):
     return self.__outside_list
```

Method get_outside_list() is defined to get the category of all the test points which is stored in the variable outside_list after running the rca() method.

```
def output_id_category(self, filepath):
  test_keys = get_id(filepath)
  test_values = self.get_outside_list()
  res = {}
  for key in test_keys:
       for value in test_values:
       res[key] = value
       test_values.remove(value)
       break
  return res
```

In the end, method output_id_category() is defined to store the id of test points and their category together to provide the result.

Task 3. The Categorization of Special Cases

The special cases of RCA are boundary points and when the ray crosses the vertex and is counted more than actually needed. So, first the boundary problem is solved using the cross product method. And problems with the counting number method have been solved by using the X-intersect between the ray of the test point and the edge. If the test point X is less than the intersection, then it has crossed the boundary, and the cross variable will be increased. In this way, the problem of counting vertices wrong is solved.

Solving Boundary case:

```
on_boundary = False
if ((xs - ver_x1) * (ver_y2 - ver_y1) == (ver_x2 - ver_x1) * (ys - ver_y1)
and
      min(ver_x1, ver_x2) <= xs <= max(ver_x1, ver_x2) and
      min(ver_y1, ver_y2) <= ys <= max(ver_y1, ver_y2)):
      on_boundary = True

if on_boundary:
   category = 'boundary'
   append = True
   break
```

Solving vertex case:

```
if (ver_y1 <= ys < ver_y2) or (ver_y1 > ys >= ver_y2):
   x_intersect = (ys - ver_y1) / (ver_y2 - ver_y1)
```

```
  if xs < (ver_x1 + x_intersect * (ver_x2 - ver_x1)):
  crosses += 1
```

Task 5. Object-Oriented Programming

5 classes are used in this software, out of which 2 are created-MinBoundReact, Rca and 3 are taken from lecture-Point, Polygon, Plotter. The MinBoundReact and Rca classes have been discussed above. In classes, private attributes have been used throughout to make it easier to debug. It is also used to control the way in which users interact with methods of the class and prevent them from altering the code.

Changes made to Point and Polygon classes

```
class Point:
      def __init__(self, name, x, y):
        self.__name = name
        self.__x = x
        self.__y = y

      def get_x(self):
      return self.__x

    def get_y(self):
        return self.__y

    def __repr__(self):
         return f'Point({self.__x},{self.__y})'
```

Here, __repr__() is added to see what is being inputted and also helps in debugging the code.

```
class Polygon:
   def __init__(self, name, points):
         self.__name = name
         self.__points = points

   def get_points(self):
       return self.__points

    def get_xs(self):
       return [point.get_x() for point in self.__points]
    def get_ys(self):
     return [point.get_y() for point in self.__points]
```

Here, two methods are added-get_xs() and get_ys() to get all the X coordinates and Y coordinates respectively in the polygon object.

Functions

A.Create Objects

```
def points_object(filepath):

   if isinstance(filepath, str) and filepath.endswith('.csv'):
     point_obj_list = []
      for i in get_point_cor(filepath):
     point_obj_list.append(Point('Id', i[0], i[1]))

   elif isinstance(filepath, list):
        for i in filepath:
            point_obj_list = [Point(i[0], i[1], i[2]) for i in filepath]
   return point_obj_list

def polygon_object(filepath):
      return Polygon('polygon', points_object(filepath))
```

Here, in the function points_object(), the first condition takes CSV files and creates points objects from them and in the elif statement, it creates point objects from the list because 2 options are provided to the user to input coordinates.

B.Read CSV File:

```
def read_from_csv(filepath):
      data = []
      with open(filepath, 'r') as f:
        for values in f.readlines():
            values = values.replace('\n', '')
            values = values.split(',')
            data.append(values)

        data_new = data[1:]
        xy_floats = [[float(j) for j in i] for i in data_new]
        xy = []
        for i in xy_floats:
            xy.append([i[1], i[2]])  # creating list of x,y values
        return xy
```

This function reads the CSV file and takes filepath as input and returns the list of X and Y coordinates together.

C.Write in CSV file

```
def output_csv(filepath, dictionary):
   with open(filepath, 'w') as f:
         f.write(str('Id, Category\n'))  # Adding headers to csv file
      for key, value in dictionary.items():
            f.write(f'{key},{value}\n')  # To write the output in csv file
   return 'Output can be found in out.csv file'
```

This function writes in a CSV file and gives output after categorization of test points has been completed. It takes filepath and dictionary(id, category) as input.

Task 6. On the Use of Git and GitHub

To begin with, I joined the git repository provided in the assignment. From there, I downloaded the files provided. Then added the link to the GitHub repository as my remote repository in PyCharm. Throughout the software development, I made commits and pushed those commits from local to remote repository. The entire development process enabled me to appreciate its usefulness as I could retrieve my old code if I messed-up the code on a local repository.

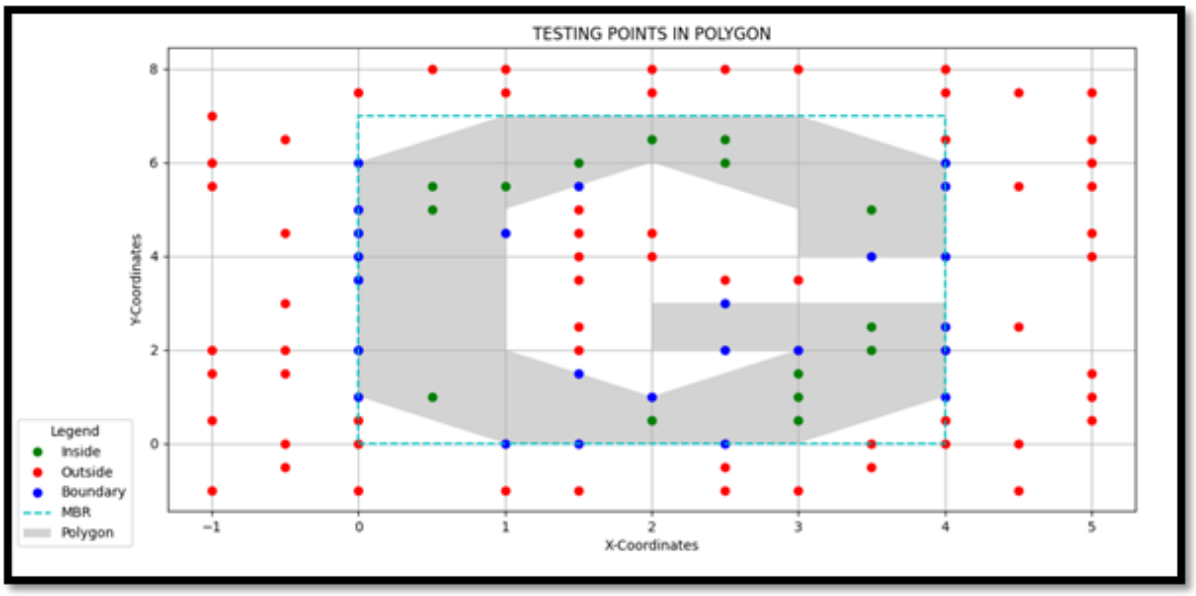
Task 9. Plotting

```
class Plotter:
      def __init__(self):
      plt.figure()

      def add_polygon(self, xs, ys):
      plt.fill(xs, ys, 'lightgray', label='Polygon')

     def add_point(self, x, y, kind=None):
         if kind == 'outside':
            plt.plot(x, y, 'ro', label='Outside')
         elif kind == 'boundary':
            plt.plot(x, y, 'bo', label='Boundary')
         elif kind == 'inside':
            plt.plot(x, y, 'go', label='Inside')
         else:
            plt.plot(x, y, 'ko', label='Unclassified')

     def add_mbr(self, x, y):
         plt.plot(x, y, 'c--',label='MBR')
```

```
def show(self):
    handles, labels = plt.gca().get_legend_handles_labels()
    by_label = OrderedDict(zip(labels, handles))
    plt.legend(by_label.values(), by_label.keys(), loc='lower right',
    bbox_to_anchor=(-.03, -.09), ncol=1, title='Legend')
    plt.title('TESTING POINTS IN POLYGON')
    plt.xlabel('X-Coordinates')
    plt.ylabel('Y-Coordinates')
    plt.grid(True)
    plt.show()
```

Changes made to Plotter:

a. Added title to the plot
b. Label the X and Y axes.
c. Moved legend out of the plot. Moved it in the left-bottom corner. Gave the legend box a title.
d. Added plot to visualize the MBR, gave it dotted lines to not overburden the plot.
e. Added grid to make visualization easier and can check that the points are categorized correctly.



An object of the plotter class is created. To plot the points, a FOR loop is used.

```
plotter = Plotter()
plotter.add_polygon(poly_x_list, poly_y_list)
for i in range(len(pt_x_list)):
    plotter.add_point(pt_x_list[i], pt_y_list[i], list_of_output[i][0])
    plotter.add_mbr([min(poly_x_list), max(poly_x_list), max(poly_x_list),
                     min(poly_x_list), min(poly_x_list)], [min(poly_y_list),
```

```
                    min(poly_y_list), max(poly_y_list), max(poly_y_list),
                    min(poly_y_list)])
plotter.show()
```

Plotting of the ray is not completed, because I ran out of time. Though figured-out how to plot a single-ray using the arrow function.

Task 10. Error Handling

```
1. def user_choice():
       while True:
            choice = int(input('Enter your choice:\n'))
            if choice == 1:
                csv_input()
                break

            elif choice == 2:
                 keyboard_input()
                 break
            else:
                 print('Invalid choice! Choice can be 1 or 2 \n')
              print('Try again!!')

       return choice
```

This function is used to take input from users. If the user enter 1, csv_input() function is invoked, which takes CSV as input and further processes to categorize points and if the user Enter' 2 , the keyboards_input() is invoked, will allow user to enter point from the keyboard and if user types something else. Invalid choice message will be displayed. Here, while and break are used so that if the user enters the wrong choice, it will get the message invalid choice and get a chance to again enter the choice.

```
2. try:
       x = float(input('x coordinate: '))
       y = float(input('y coordinate: '))
       name = input('Enter point name: ')

   except ValueError as v:
        raise Exception('invalid datatype') from None
```

In the function to take input from the user, try and except block is used to check that the user's input is the correct data type, but if it enters incorrect, they will get a message saying 'Invalid datatype'.

```
3. def output_filename():
        filename = '{}.csv'.format(str(input('Enter output filename:
'')))
            return filename
```

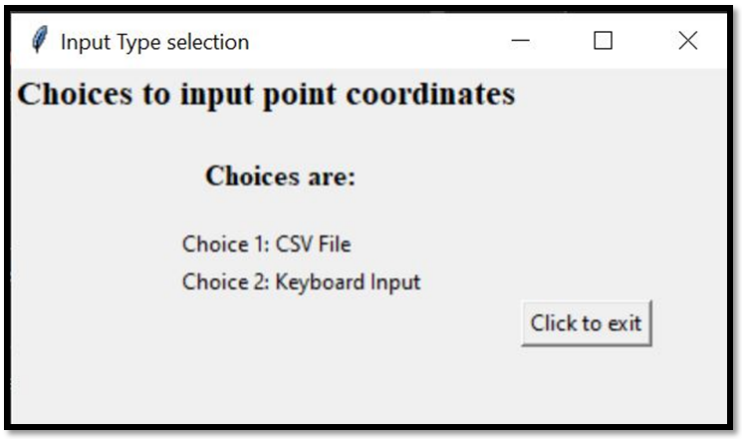
If the user takes option 1 to input coordinates, i.e. using the CSV File, as a preventive measure, the function output_filename() has the .csv extension so that users don't have to type. It formats the output to the same directory correctly without worrying about the user's input.

Task 11. Additional Features

Added GUI:

```
import tkinter as tk
def input_choice_display():
    my_w = tk.Tk()
    my_w.geometry('400x200')  # To specify size
    my_w.title('Input Type selection')
      font1 = ('Times', 14, 'bold')
      font2 = ('Times', 12, 'bold')
    l1 = tk.Label(my_w, text='Choices to input point coordinates',
         font=font1).grid(row=1, column=1)
    l2 = tk.Label(my_w, text='\n     Choices are:', font=font2).grid(row=2,
         column=1)
    l3 = tk.Label(my_w, text='\nChoice 1: CSV File').grid(row=3, column=1)
    l4 = tk.Label(my_w, text='              Choice 2: Keyboard
         Input').grid(row=4, column=1)
    eb = tk.Button(my_w, text='Click to exit',command=my_w.quit).grid(row=6,
         column=2)
    my_w.mainloop()
```

To implement GUI, input_choice_display() function is created. It was done by importing the tkinter module. It displays the choices available to the user to input the test point coordinates.



Limitations: Could not add more functionality to this because I ran out of time and very little knowledge of the package, made this by watching videos online and referring to a book.

2. Users can pass test points using the CSV File.

Git Log

Total of 62 commits were made throughout the development of this software.

commit 565e879536574b1219d54078a30e269f75420a8f (HEAD -> master)
Author: pratibha1708 <ucesatb@ucl.ac.uk>
Date: Mon Nov 22 07:18:00 2021 +0000

Final changes done

This is how it looks like, it has been rearranged to fit.

commit 8743752847022ab00ac0e6c2aa9d10dead63d355 (HEAD -> master) Author: pratibha1708 <ucesatb@ucl.ac.uk> Date: Mon Nov 22 02:34:40 2021 +0000 Functions for main_from_user added
commit 8f78d5910843f7711dc86765962df3f693fcd86d Author: pratibha1708 <ucesatb@ucl.ac.uk>
Date: Sat Nov 20 20:53:30 2021 +0000 Made final changes
commit 099965abf3740e84131380a9a2c5213e40c25679 Author: pratibha1708 <ucesatb@ucl.ac.uk>
Date: Fri Nov 19 12:35:16 2021 +0000 Added MBR plot
commit e2a3ea3144690be01038cb70c6ad1fe77d348602 Author: pratibha1708 <ucesatb@ucl.ac.uk> Date: Fri Nov 19 11:47:48 2021 +0000 GUI added
commit 434f3fcd81b1280dee5b9e3c265a2deab1bce28d Author: pratibha1708 <ucesatb@ucl.ac.uk>
Date: Thu Nov 18 20:29:06 2021 +0000 Updated rca
commit ce5e1658540fe6978f722948b6e03ebe9dee508c Author: pratibha1708 <ucesatb@ucl.ac.uk>
Date: Thu Nov 18 20:28:05 2021 +0000 Added axis, title
commit eae42ff295c233d34134e0b6ace0e848b626ab8f Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Thu Nov 18 20:26:46 2021 +0000 Updating

commit 236cae96bb0fb1a29d377ed112c6f8b8aa79e250 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Thu Nov 18 20:25:33 2021 +0000 Updated functions

commit b4356a295429a2de6b838494b2baf4073ff9c7be Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Thu Nov 18 20:23:26 2021 +0000 Added repr method in class Point

commit d5f878cde2609ec63d0504c2d0fa8b06d1f627c4 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 19:35:38 2021 +0000 updated rca

commit bc9ea8981491f3409beeb8111c68e9b7ca8048ff Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 19:29:48 2021 +0000 Created file for input from user

commit 79d466283265c8584b1b2b5dba56f17a4fccf2c2 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 19:28:28 2021 +0000 Added function to read the csv and create list of x,y together

commit e5bb02ccd6185cd9ae015666793b5221543921e2 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 19:27:28 2021 +0000 Updating things in format required by assignment

commit 03c12be9c01ef46886a751552f2b4b67e08b01c6 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 10:24:31 2021 +0000 Update read_csv

commit 012ec05a9fae94043be68ec9de1f8c17e6120e32 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 17 10:21:55 2021 +0000 Changed id, from floating numbers to int for displaying result

commit 895933f95b9bf1370b801602ecc3eca861fc6b01 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 20:33:08 2021 +0000 Updating RCA

commit e630a79cebb5d3f73617114fe8ad38786ca1f77b Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 20:29:43 2021 +0000 Creating instances

commit ca9119fffa043582cb678129dca0256bc9fc81e9 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 20:25:51 2021 +0000 To plot mbr to visualise

commit 53c7e6b83adf61708623e20ac627130c786cf700 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 20:23:47 2021 +0000 Added function to write output in csv

commit 21aa914b9410dc3008c08f6c3e68e91ccacf7228 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 20:21:39 2021 +0000 Result of each point in csv file

commit 54cbffb957e30cd44365658d406a9a4c310aa234 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 17:46:58 2021 +0000 Added function to display output as dictionary(id:category)

commit eaec03682a739b23576b21ba48b5757ac12969d3 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 10:08:55 2021 +0000 Creating instances

commit 48020edd9f867fc0dfa14cb09c0322bfcaf8aafa Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 16 10:06:16 2021 +0000 RCA Special cases

commit 66290b2a3fc87cc88b0af633194e4b4b293cdd05 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Sun Nov 14 22:40:07 2021 +0000 Running RCA: counting no. method

commit e3630815b21cae9b5a610bdc6b581cf566e1ef91 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Sun Nov 14 20:47:30 2021 +0000 Added code under categorize point that implements the MBR part of problem

commit b4e4a0fe2de89c47b3a300da2c36f7c439395f7b Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Sun Nov 14 20:44:18 2021 +0000 MBR running successfully and returns position of all points w.r.t to Minimum Boundary Rectangle and points that are definitely outside MBR(i.e. definitely
outside polygon

commit d996055d52b3092d2634a337cbbda72d211a1cd5 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Fri Nov 12 21:35:07 2021 +0000 Created class MinBoundRect()to draw minimum boundary rectangle and test the input points in relation to that boundary.

commit 7a3347518e39e2717ab31dd7f1e726752cf2ac74 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Fri Nov 12 15:05:57 2021 +0000 Adding the function to return the id's of point

commit aeea731a0d551b24f2e6c2c2d6515f4c6aacde8d Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Thu Nov 11 15:14:04 2021 +0000 Checks point outside and on boundary

commit 8f280c411ba42bdd8d1c35f498b840eb11a0d5e5 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Thu Nov 11 11:49:22 2021 +0000 Successful plotting of MBR and it fetches min,max from input csv fille

commit ff3fa6c99b99784637fb11e2f2f225d6813ce503 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 10 18:03:32 2021 +0000 MBR Function

commit 0446e0f2c4845973a32799c919be4665d4b5d332 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 10 16:34:42 2021 +0000 plotting minimum boundaryrectangle

commit a7e98c7fc168def75807a63eace0c5ad17b1c378 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 10 11:57:13 2021 +0000 Function for finding boundaries of MBR

commit a2dc918b0eab1749f87348bf08096f2dd91bd103 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 10 10:18:29 2021 +0000 Read list of x,y coordinates from file

commit 896adeaecb138e0eba5e855cc70377baa0505797 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Wed Nov 10 10:15:49 2021 +0000 Create list of points objects for testing

commit 8025b504c173f07a1b89c6fdd991235f1026660b Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 9 20:05:50 2021 +0000 creating point objects

commit 6b5bed6f497d9d8defd51f3e0431976b0c0d18a2 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 9 18:04:26 2021 +0000 plotting polygon

commit 93002c87bd5f79c137dee92ae81b4e2616c971f7 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 9 17:20:30 2021 +0000 commiting functions read_csv, get_xs and get_ys

commit ea68303659aa137659fe7f31b83b260af0fe0c48 Merge: 7507af9 768140a

Author: pratibha1708 <ucesatb@ucl.ac.uk> Date: Tue Nov 9 15:38:40 2021 +0000 Merge branch 'master' of https://github.com/CEGE0096/point-in-polygon-test-pratibha1708

commit 7507af99a6e7db47aad602d919e2932b86592359 Author: pratibha1708 <ucesatb@ucl.ac.uk>

Date: Tue Nov 9 15:04:33 2021 +0000 Committing file (read_csv) containing function to load csv.

commit 5d04401caa9290f10889d19a6eef40b7267c38be Author: pratibha1708 <ucesatb@ucl.ac.uk>

commit cf4c790cecf875d0303593ceabb04775885499cf Author: github-classroom[bot] <66690702+github-classroom[bot]@users.noreply.github.com> Date: Wed Nov 3 11:33:36 2021 +0000 Initial commit

4. REFERENCES

- Python Programming for the Absolute Beginner, Third Edition, Michael Dawson.
- https://blog.csdn.net/zhouzi2018/article/details/81735132?utm_medium=distribute.pc_relevant.none-task-blog-title-2&spm=1001.2101.3001.4242
- https://github.com/sasamil/PointInPolygon_Py/blob/master/pointInside.py#L50
- https://www.plus2net.com/python
- https://stackoverflow.com/questions/52725278/during-handling-of-the-above-exception-another-exception-occurred