**Programming Assignment #3 Report**

Computer Science and Engineering

2010003970

Namwoo Kim

1. **Summary of algorithm**

This program implements DBSCAN algorithm to classify points into several clusters. The basic structure is based on the pseudo code in the textbook.

First, retrieve one unvisited point(p) and find points which are neighborhood radius(eps) of given point(N). After that, If the number of neighborhood points(N) is more than minPts, creating a new cluster(C)and add current point p.

Second, iterating all points in N called p’. Mark p’ as visited if it is unvisited and then find points which are neighborhood radius of p’ and then add those points to N if the number of points are more than minPts.

Third, add p’ into cluster C if p’ is not yet a member of any cluster.

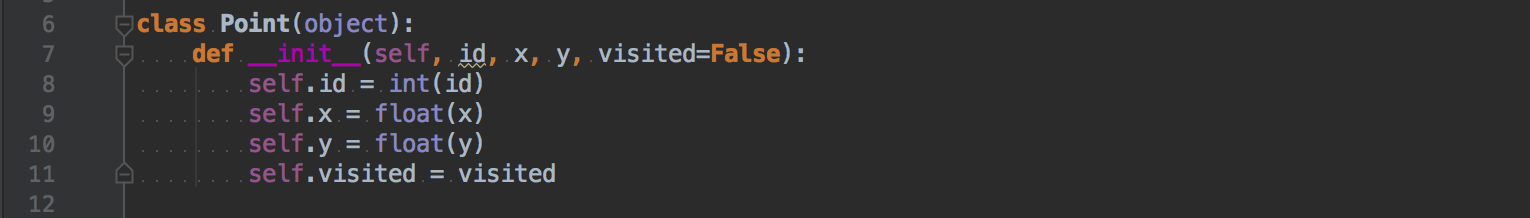
Fourth, if all process ended up, adjust noise points to be included in some cluster.

Finally, sort clusters by number of points and export clusters within the number of clusters.

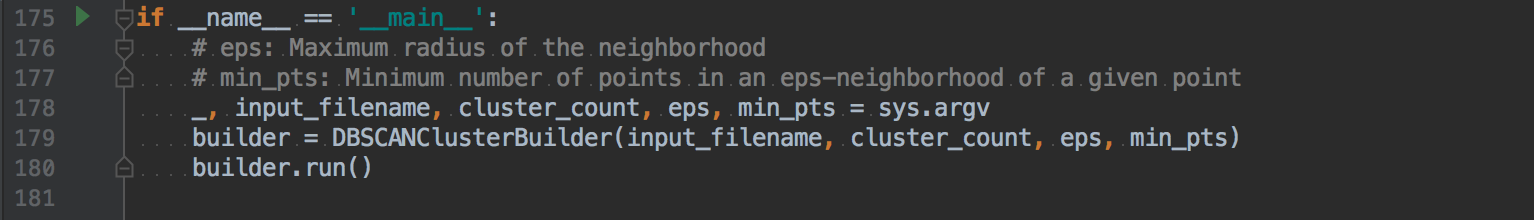
1. **Detailed description of my codes**

**<basic data structure>**

Point class includes id, x-coordinate, y-coordinate and the status whether visited or not.

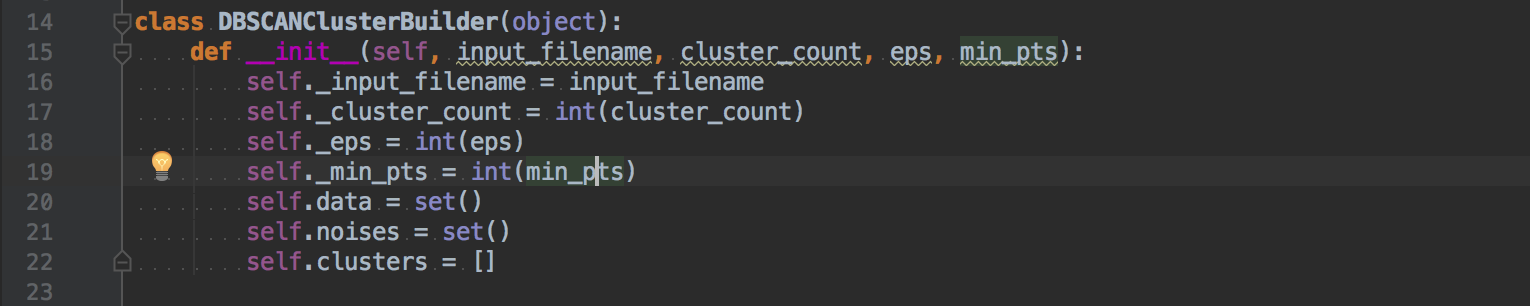


**<Initializer part>**



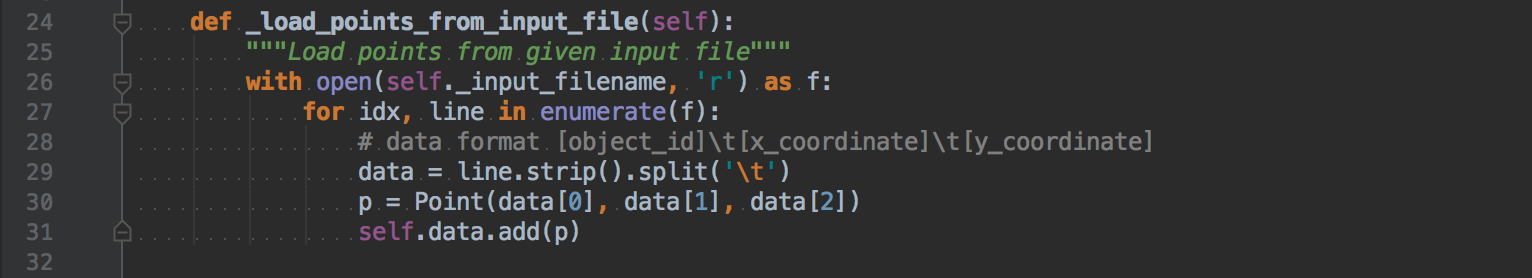
This is an initial function for DBSCANClusterBuilder class. It takes input filename, maximum cluster count, eps(radius), minimum number of points of radius neighborhood.

**<class DBSCANClusterBuilder>**

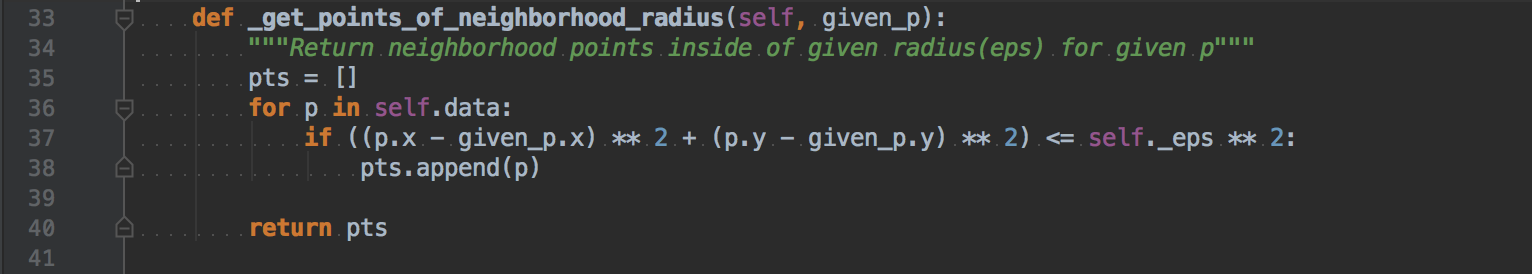


This builder class contains variables as above. The variable self.data contains all points from input file, self.noises contains all noise points from all clusters and self.clusters contains all newly created cluster informations.

**<function \_load\_points\_from\_input\_file>**

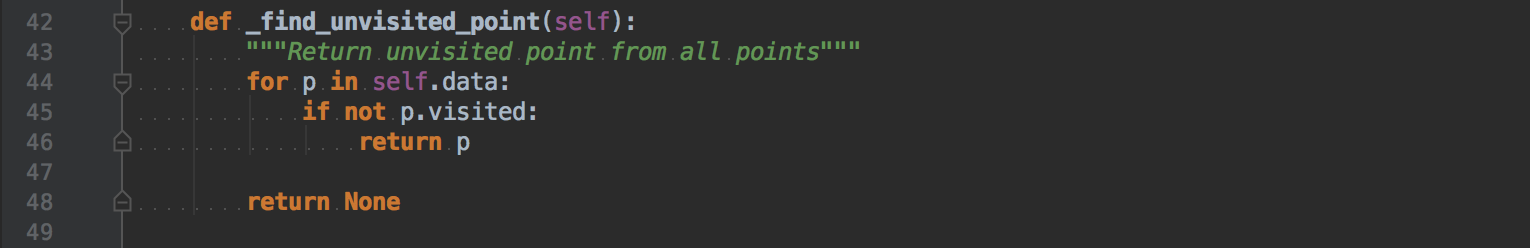
  
This function load data from input file and transform into Point class which can include id, x-coordinate and y-coordinate. Then, add point into variable self.data(which is a list that contains all points)

**<function \_get\_points\_of\_neighborhood\_radius>**



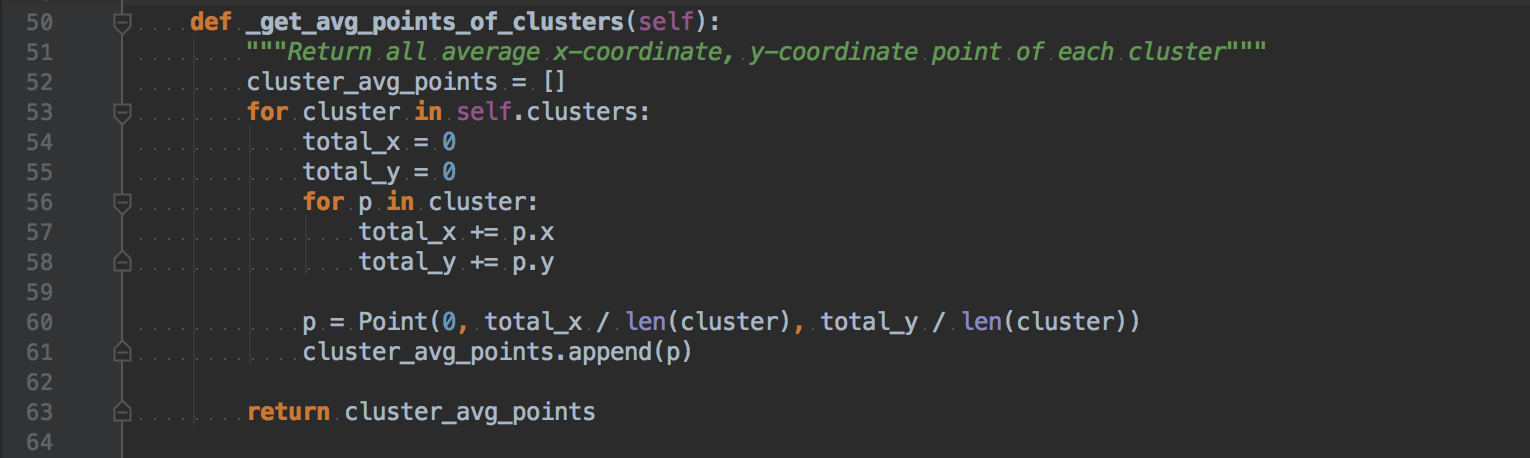
This function returns all points which is inside of radius(self.eps) for given point P.

**<function \_find\_unvisited\_point>**



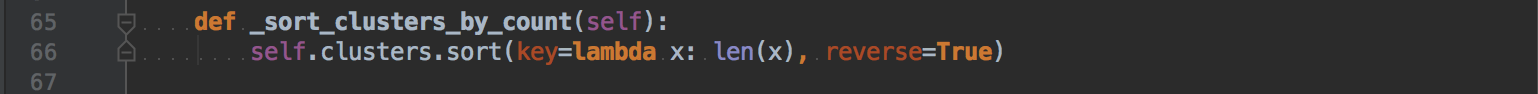
This function simply retrieve all points in self.data and then return one point which is unvisited.

**<function \_get\_avg\_points\_of\_clusters>**



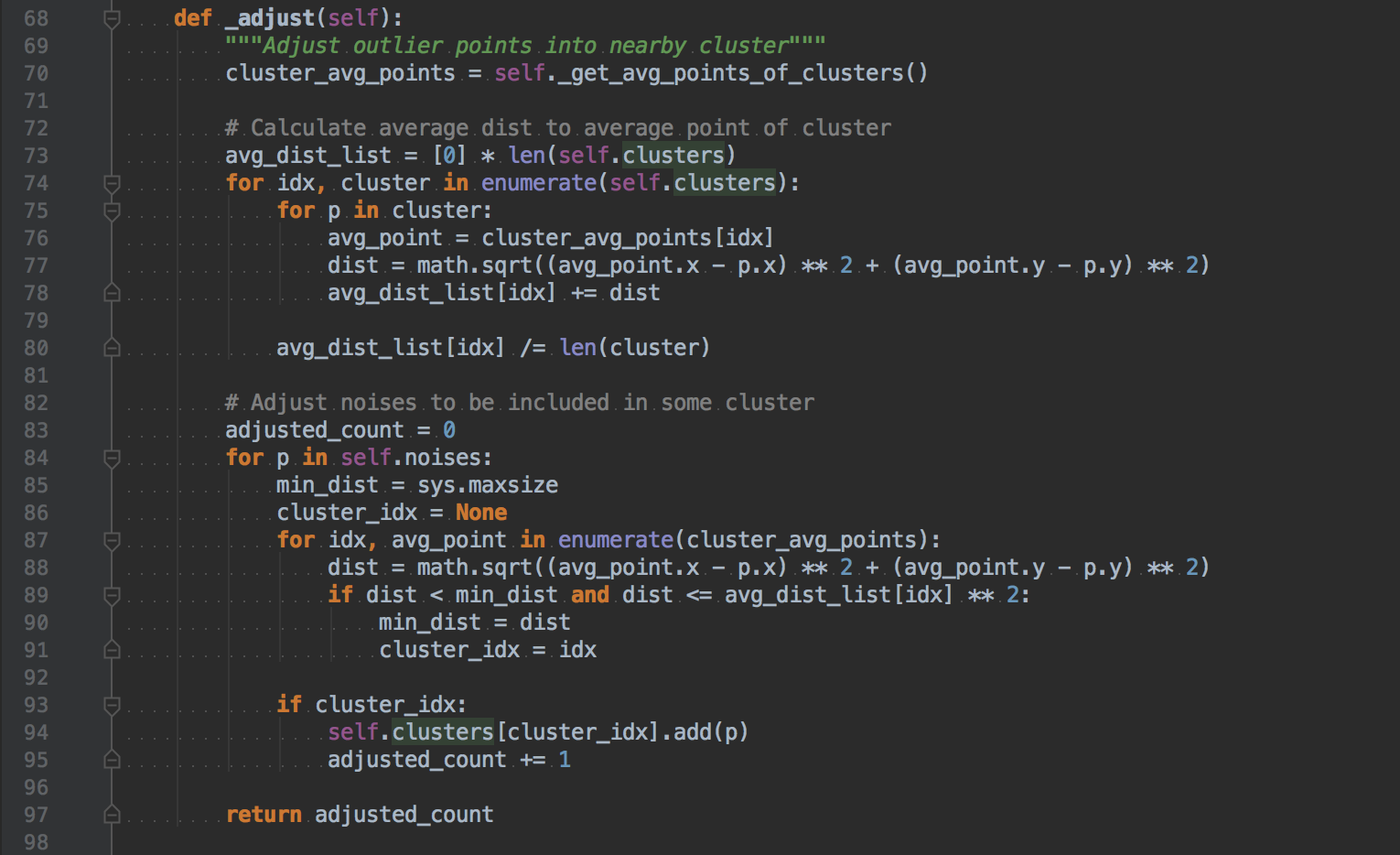
This function calculates average x-coordinate, y-coordinate of each clusters and return as a list.

**<function \_sort\_clusters\_by\_count>**



This function sort clusters by the number of points included in each clusters into descending order.

**<function \_adjust>**

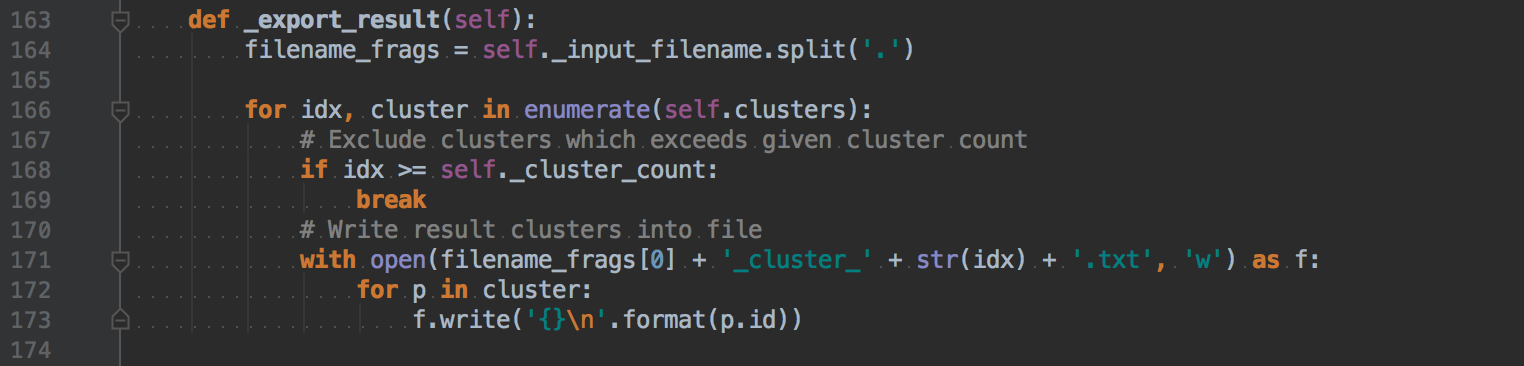


Line 73~80: Calculates average distance from average point of each cluster to every point included in the cluster.

Line 83~91: Retrieve all noises and find the proper cluster index if the distance to average point of each cluster is below than the square of average distance.

Line 93: Add this point to specific cluster if there’s a point that satisfying rules (minimum distance to average point of each cluster and this distance is below than the square of average distance of each cluster)

<function \_export\_result>

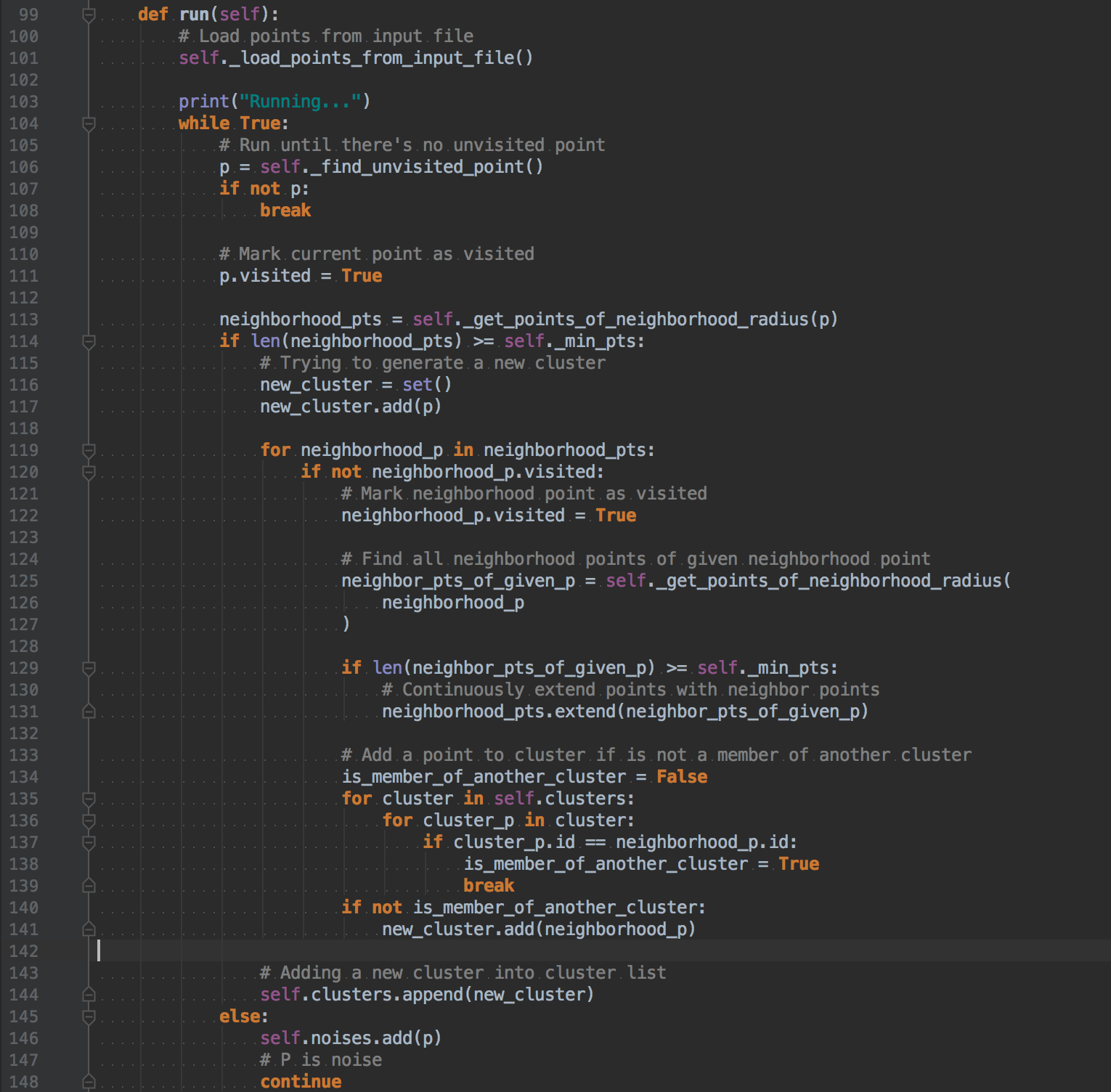


This function simply export data into given format.

Line 168: Check the number of clusters exported and stops when the number of clusters exceed given cluster count.

Line 173: Export id of Point

**<function run>**



This function’s logic is also same as the pseudo code from textbook.

Line 101: Load all points from input file

Line 106: Retrieve one unvisited point from data

Line 113: Retrieve all radius neighborhood points of given point

Line 114~117: Check if the number of radius neighborhood points is more than minimum counts. Then, make a new cluster and add current point into the cluster.

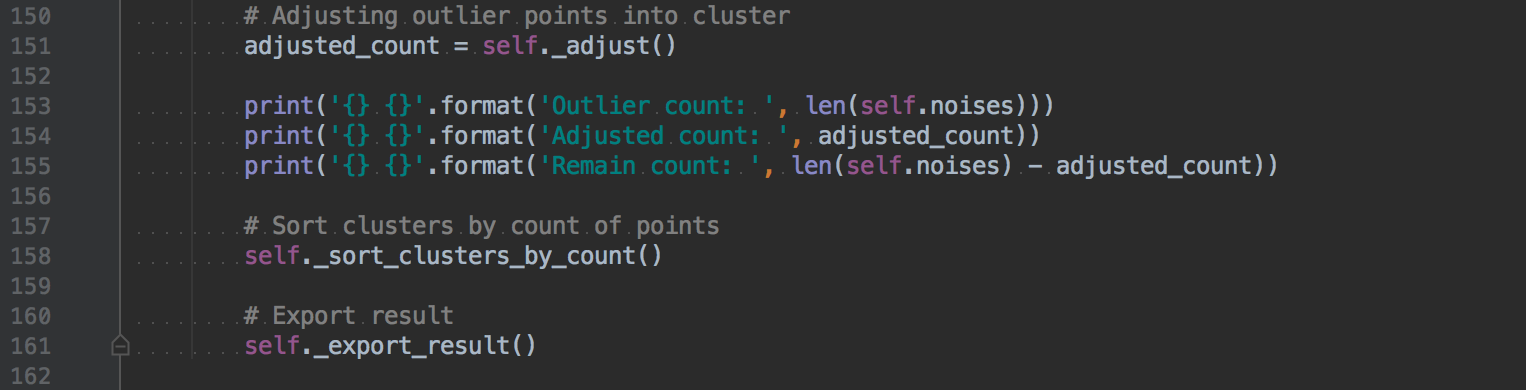
Line 119~120: Retrieve all points of radius neighborhood points and find unvisited one

Line 125~131: Retrieve all radius neighborhood points of given neighborhood point and merge points into the list which generated in Line 113 If the number of points is more than minimum count.

Line 134~141: If current point is not a member of another cluster, add the point into newly created cluster which generated in Line 116~7.

Line 144: Add a newly generated cluster into cluster list.

Line 146: Mark as a noise If the given point has no radius neighborhoods which has the number of points more than minimum points.



Line 151: Execute adjust function to adjust noise points which can be included in some cluster.

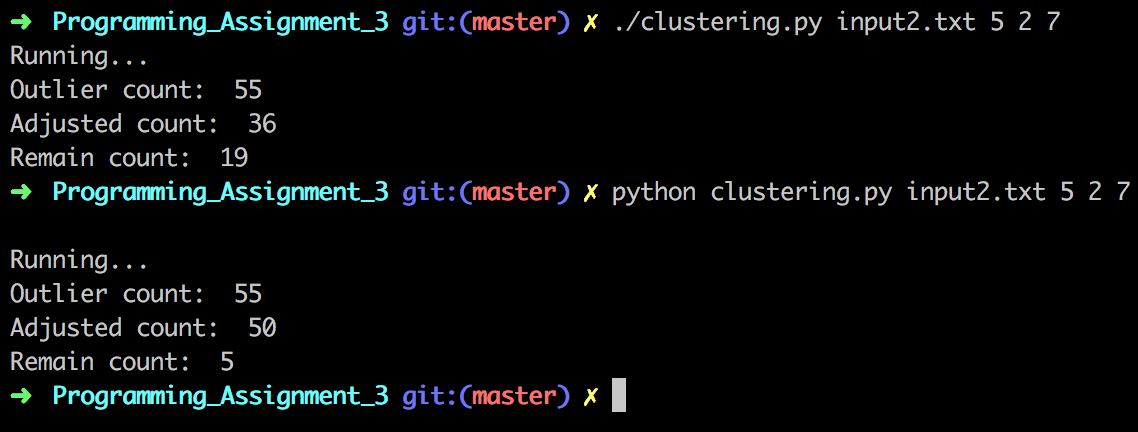
Line 158: Sort clusters by the number of included points

Line 161: Export result

1. **Instructions for compiling my source code at TA’s computer**

This program tested with Python 2.7 and 3.6. You can easily run this program in any other environments which has python because python program doesn’t need to compile for run.

In mac OS, there is a preinstalled Python.

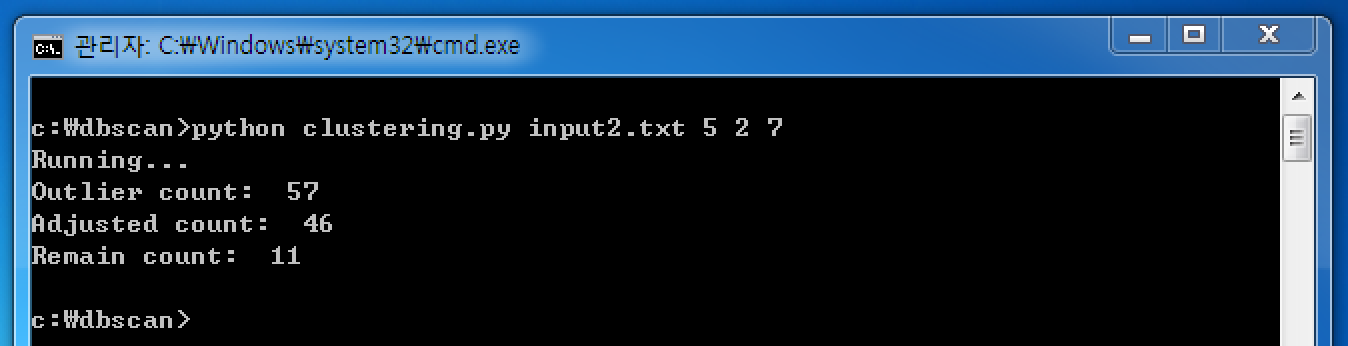


OK for both ways.

In windows, Install python 2.7 first and then execute this in your shell.

setx PATH "C:\Python27;C:\Python27\Scripts;C:\Python27\Lib\site-packages"

This is the most important part. Adding a PATH variable in the shell and quit and re-open terminal. If you enter python in terminal and It works, you can run my program.



1. **Any other specification of my implementation and testing**

I’ve tested both mac OS and Windows environments and successfully tested with the program that TA given to me. I think there’s nothing to mention for any specifications.