**Data Science (ITE4005)**

**Programming Assignment #4**

**Recommender System**



컴퓨터공학부 컴퓨터전공

2009004065 유건열

[1. Environment 2](#_Toc484560509)

[2. Summary of algorithm 2](#_Toc484560510)

[3. Detailed Description of codes 4](#_Toc484560511)

[(1) Main flow 4](#_Toc484560512)

[(2) Detail description 4](#_Toc484560513)

[ Imported Libraries 4](#_Toc484560514)

[ Class for rating tuple 4](#_Toc484560515)

[ Class for rating matrix cell 5](#_Toc484560516)

[ Starting the timer, and handling user’s argument 5](#_Toc484560517)

[ Opening files 5](#_Toc484560518)

[ Getting data tuples from training file 6](#_Toc484560519)

[ Making and assigning rating matrix 6](#_Toc484560520)

[ Calculating Pearson correlation coefficient 7](#_Toc484560521)

[ Calculating rating average of two users 8](#_Toc484560522)

[ Calculating P.C.C. value 8](#_Toc484560523)

[ Sorting neighbor groups by P.C.C. 8](#_Toc484560524)

[ Rating prediction 8](#_Toc484560525)

[ Writing ratings 9](#_Toc484560526)

[ Checking running time 9](#_Toc484560527)

[4. Instructions for executing the source code 10](#_Toc484560528)

[5. Result of test 11](#_Toc484560529)

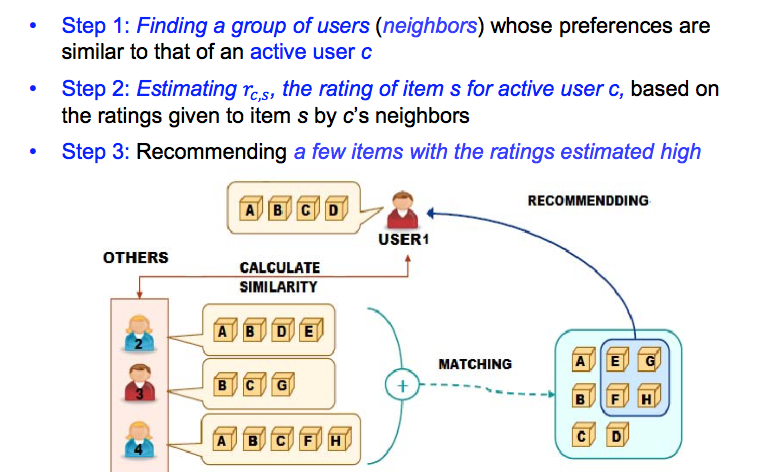
# 1. Environment

- OS : OS X

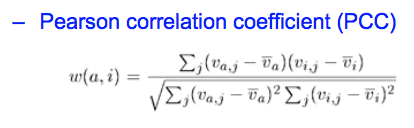
- Language : Python 2.7.12

# 2. Summary of algorithm

The objective of this assignment is to predict users’ movie ratings. It’s about recommender system. In this assignment, I used ‘Collaborative Filtering’.



This picture is saying the main idea of collaborative filtering. First, by rating history, the system finds the neighbors of the target user. In this situation, neighbor means the people who have high similarity measure with the target user. For example, comparing me, if some people rated similar rating to same movies, they can be my neighbors. After organizing neighbor groups, the system will predict the rating. If the target user didn’t watch movie A, the system can predict the rating of the target user for movie A. For prediction, the system scans the ratings of target user’s neighbors for movie A. By these ratings, the system calculates the expected value of rating.



When the system finds neighbor, the similarity measure will be needed. I used Pearson correlation coefficient.

After making neighbor groups by similarity measure, we have to aggregate the ratings on a target item given by the neighbors. There are three methods for aggregation in our class. I used weighted average. The formula is here.



I thought normal average doesn’t consider the difference of the similarity measure. So, to differentiate ‘close’ neighbor and ‘far’ neighbor, I used weighted average.

# 3. Detailed Description of codes

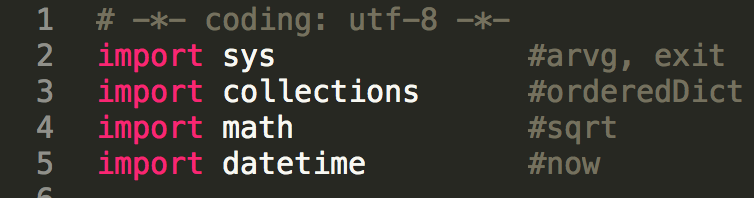
## (1) Main flow

First, the program reads data from base file. And it makes the list of rating tuples. With this, it makes rating matrix. But in this case, the matrix will be a sparse matrix. So, I used dictionary for the rating matrix, instead of 2-dimensional array. And with this rating matrix, it calculates the Pearson correlation coefficient among users. If some pair of users have higher P.C.C value, it means they are neighbor. So, to make finding neighbor easy, the program sorts the P.C.C dictionary.

After making P.C.C dictionary, it reads data from test file. Finally, the program predicts the ratings in the test file, and writes it into output file.

## (2) Detail description

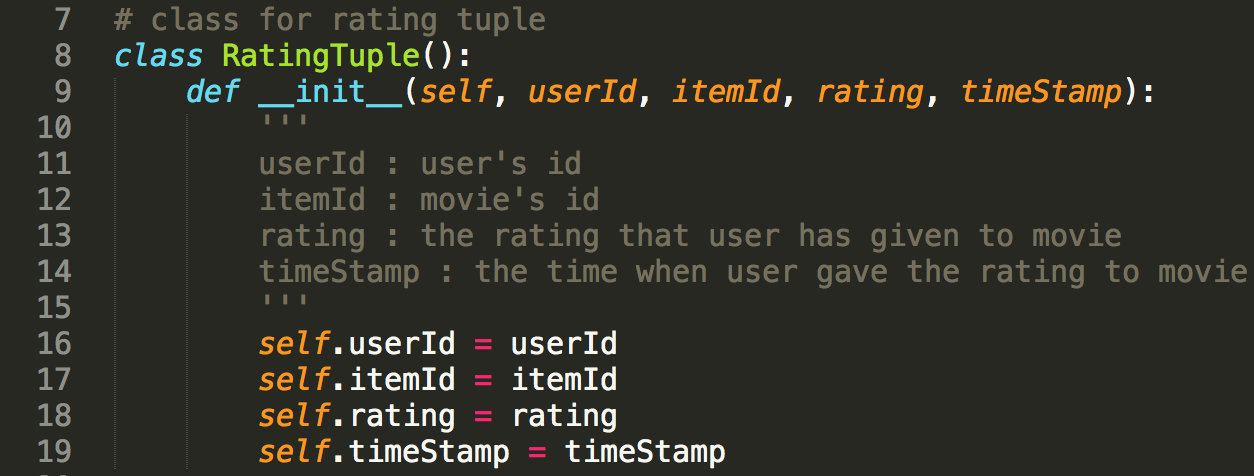
### Imported Libraries



I imported sys for ‘argv’ that is for user’s argument, and for exit, collections for orderedDict, math for sqrt.

Datetime is not related with the main flow. But I imported it to show the running time.

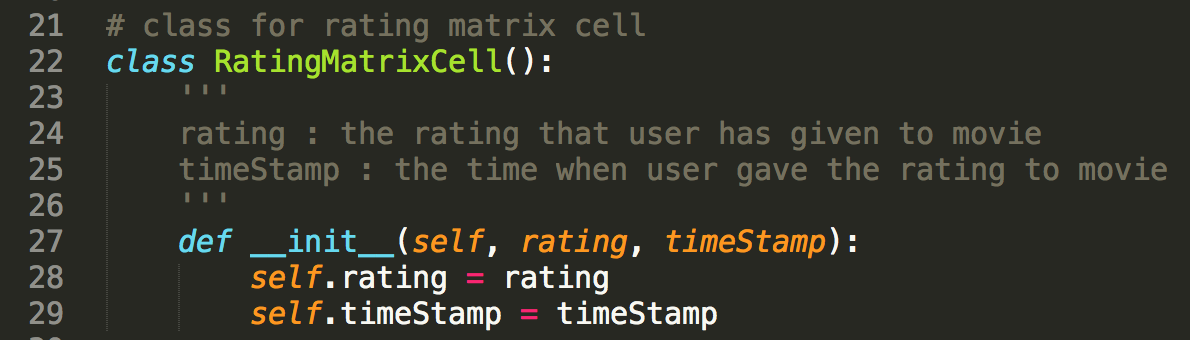
### Class for rating tuple



This is ‘RatingTuple’ class in my program. Every line in base file will be an instance of this ‘RatingTuple’. Lines of test file as well.

* UserId : user’s id
* ItemId : movie’s id
* Rating : the rating that user has given to movie
* Timestamp : the time when user gave the rating to movie

### Class for rating matrix cell



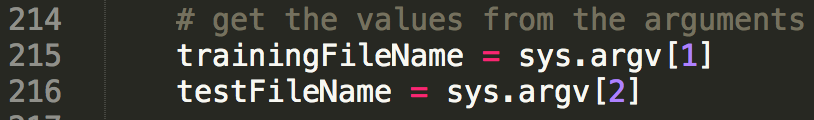
This is ‘RatingMatrixCell’ class in my program. Rating Matrix is the set of instances of this class.

### Starting the timer, and handling user’s argument



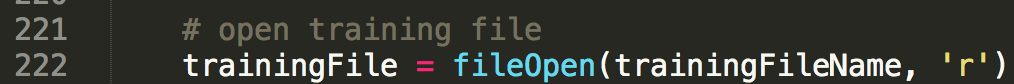
This program takes some time for one example like 5~6 minutes. so I wanted check the running time. Before starting, check the start time. And when predicting the rating by neighbors’ rating, I wanted to limit the number of neighbors. Because, if I don’t do this, the other users that have low similarity measure can affect to the prediction.

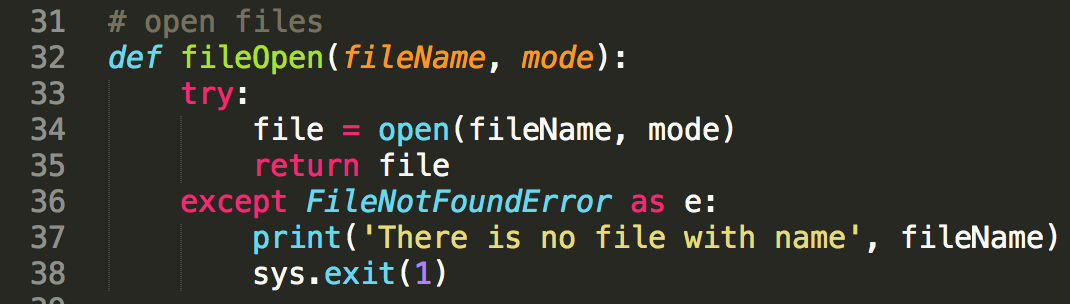
After these things, it checks the arguments’ number first. We need 3 argument including file name. (training file name, test file name) If the number of arguments is not 3, it shows ‘how to execute’ and the program will be exit.



If user inserts 3 arguments exactly, we can get training file name, and test file name.

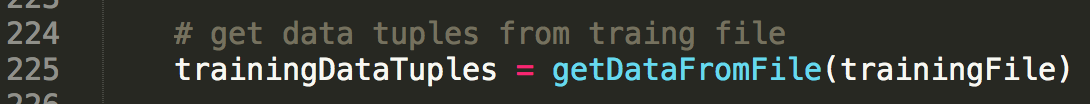
### Opening files

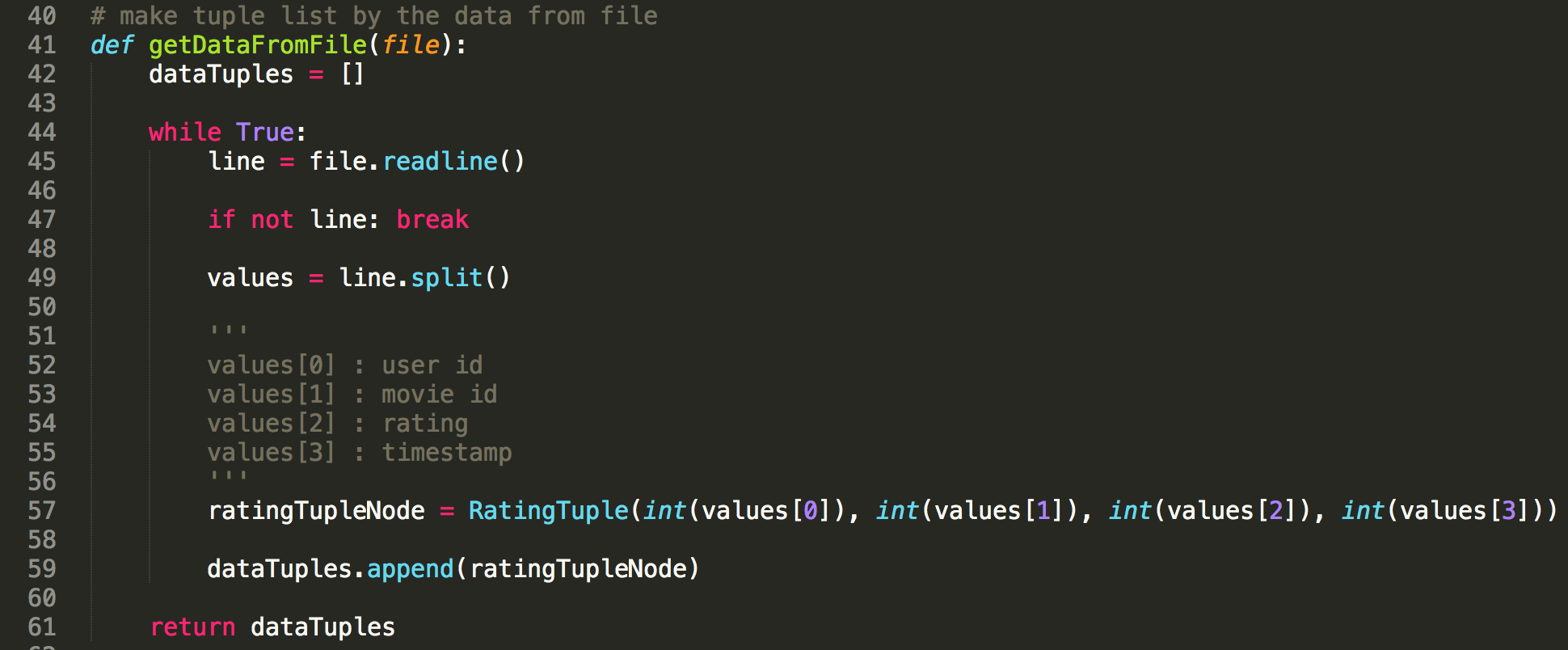




It opens file for training data. ‘fileOpen’ function will do that. If ‘FileNotFoundError’ occurs, program will be exit. (If user insert file name is not in directory.)

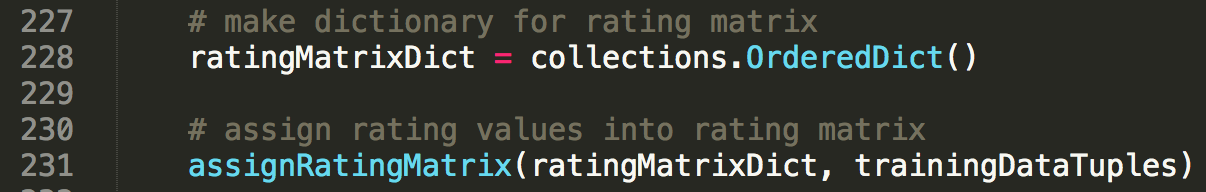
### Getting data tuples from training file

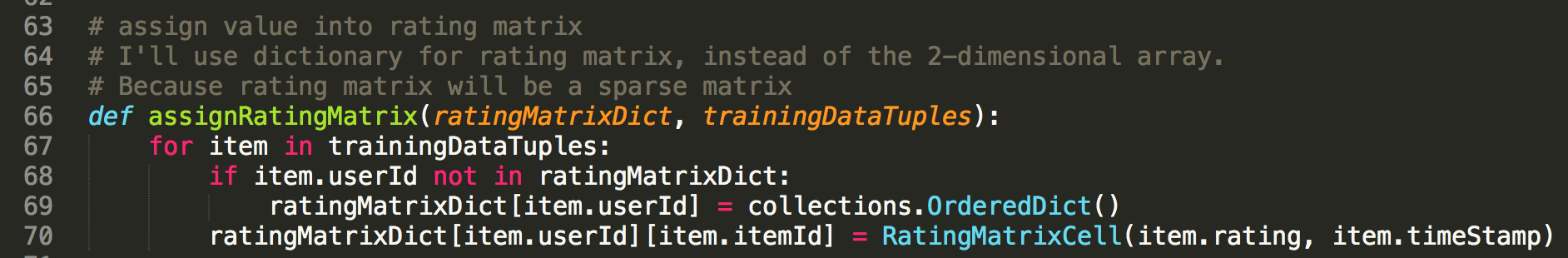




‘getDataFromFile’ method reads every line from training file. Each line has user’s id, movie’s id, rating, and time stamp. ‘dataTuples’ list saves every ‘RatingTuple’ instance. These instances generate with above separated values.

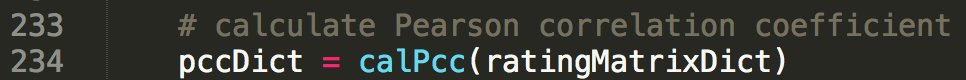
### Making and assigning rating matrix

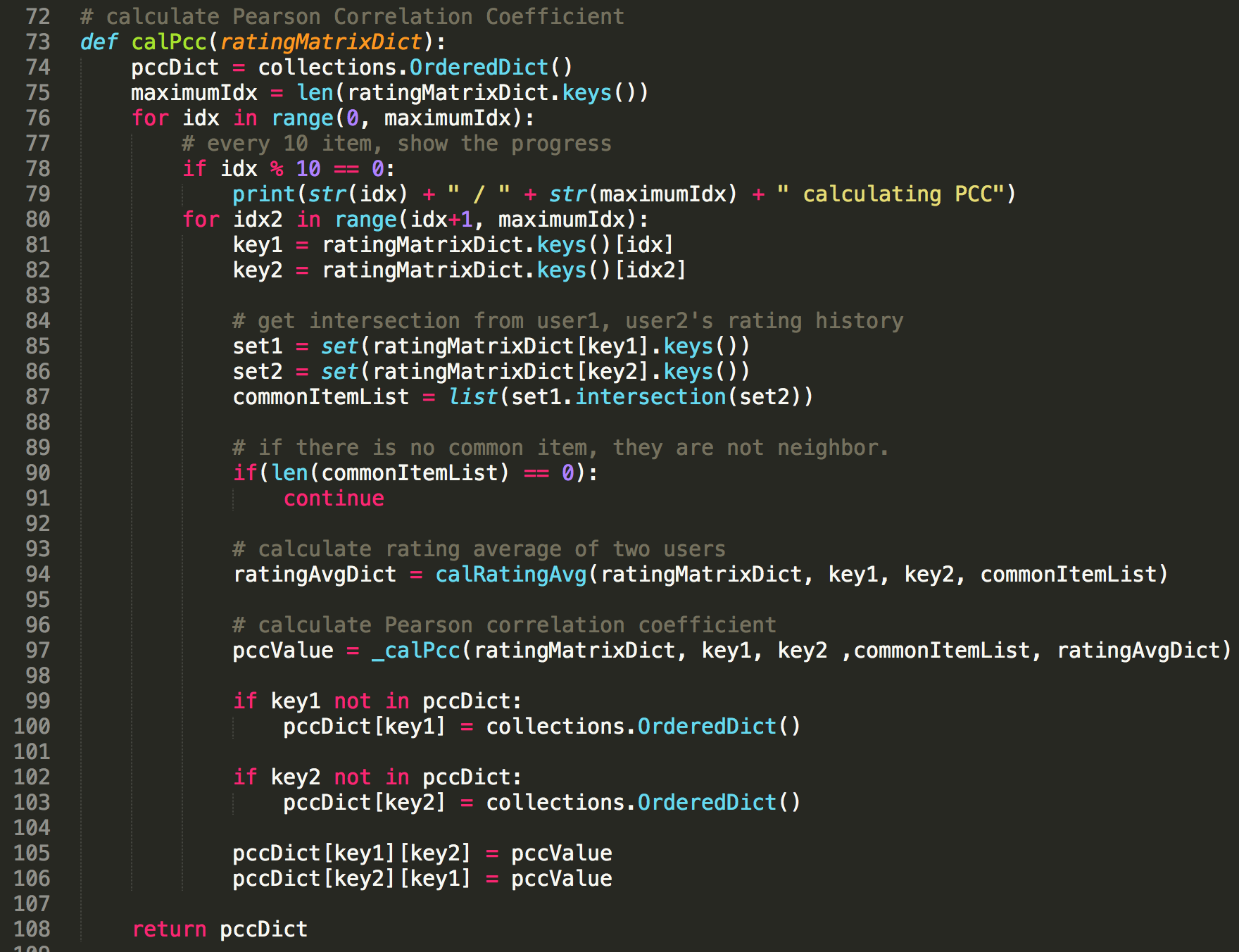




It’s time to make rating matrix. In normal case of movie rating, the rating matrix will be a sparse matrix. Because there are many movies, and many users, but users don’t give rating to most of the movies. So I used dictionary for rating matrix, instead of 2-dimensional array.

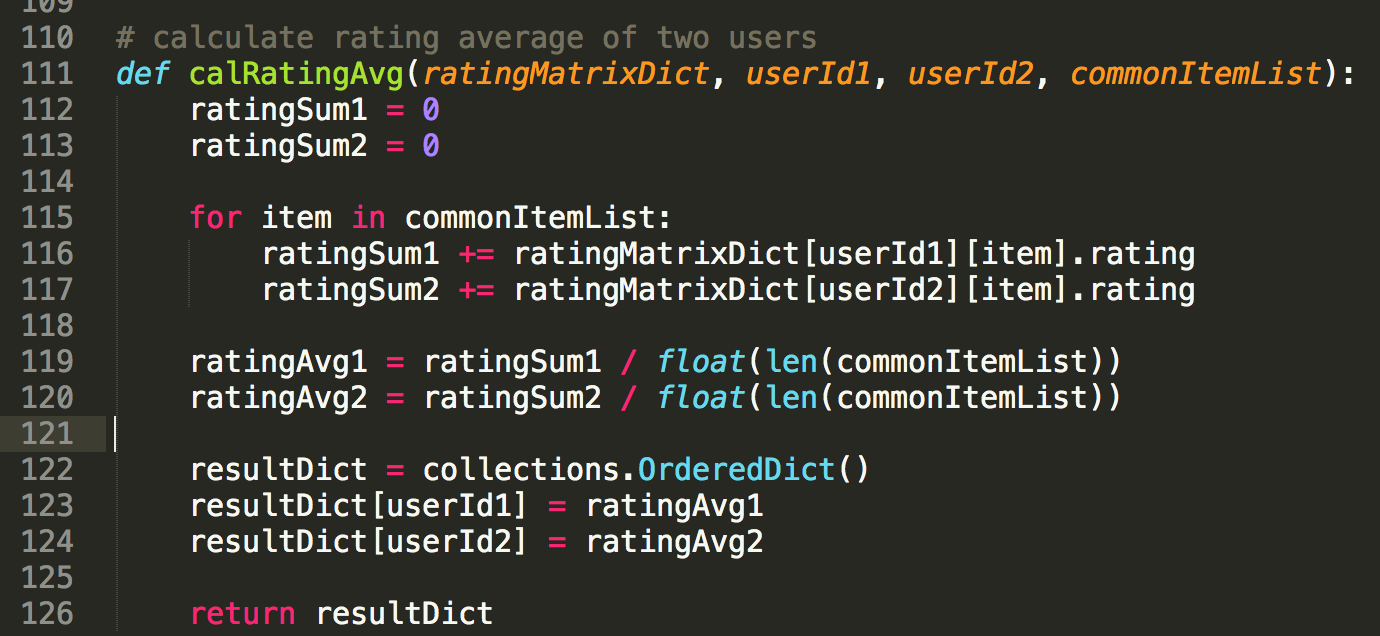
### Calculating Pearson correlation coefficient





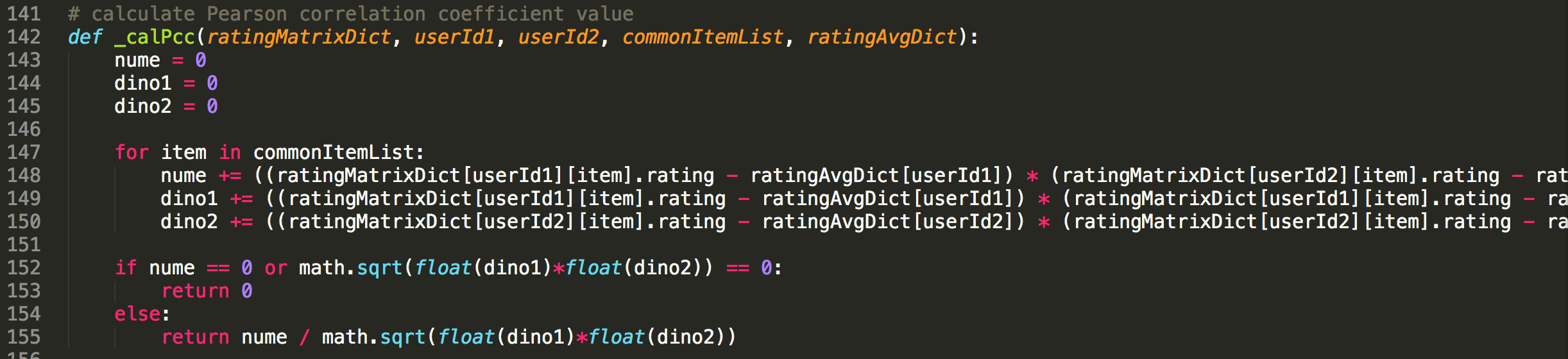
I used Pearson correlation coefficient for similarity measure. This value is for finding neighbor groups. First, it compares every pair of users’ rating history. It finds the common items among two users. If there are no common item, it passes the pair. Otherwise, it calculates the P.C.C value, and saves it in ‘pccDict’.

### Calculating rating average of two users



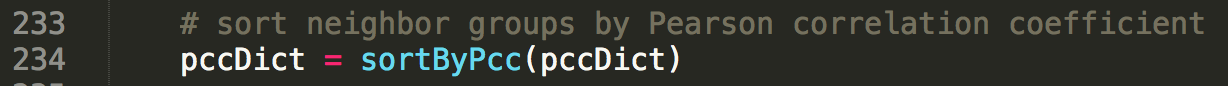
This method is for calculating the rating average of two users. It is needed for calculating P.C.C.

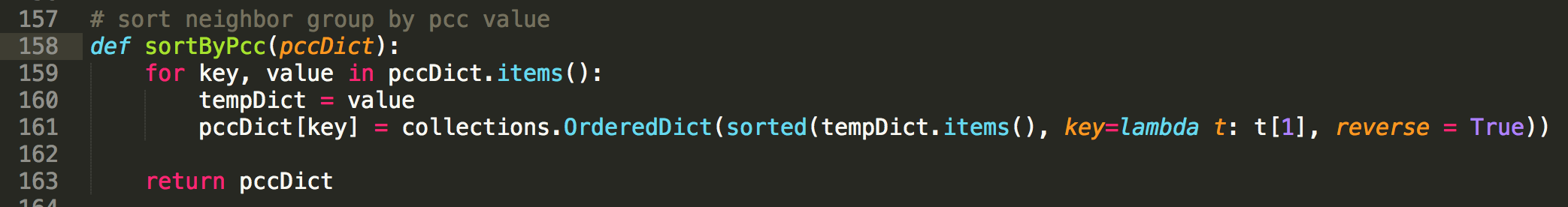
### Calculating P.C.C. value



This method for calculating P.C.C. value. We already have common item list of two users, and rating average dictionary.

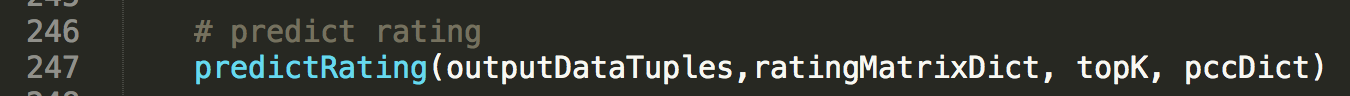
### Sorting neighbor groups by P.C.C.





We have to predict the rating with neighbor group’s rating. If we want to predict exactly, we should use neighbor that has high similarity measure. So, to make prediction easy, it sorts ‘pccDict’ by P.C.C.

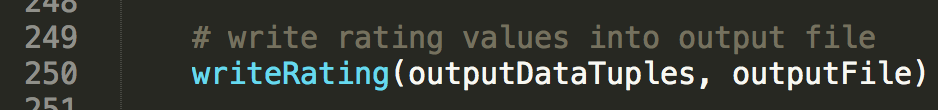
### Rating prediction

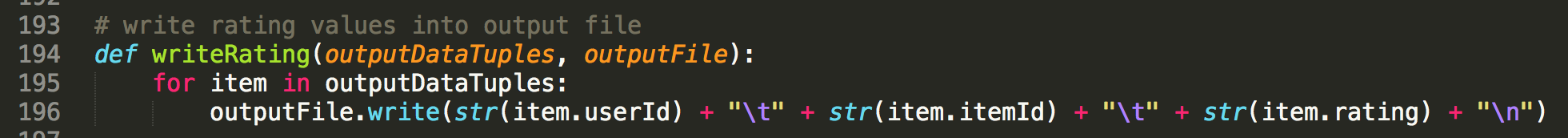




Finally, let’s predict the ratings. I used weighted sum here. Because I wanted close neighbor affects to the rating more. For accuracy, it just scans TOP K neighbors only.

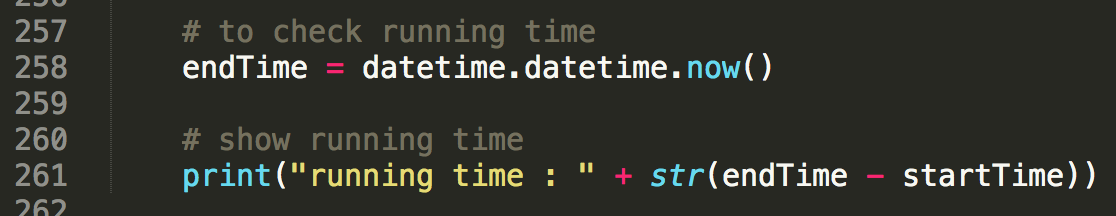
### Writing ratings





After prediction, it writes ratings into output file.

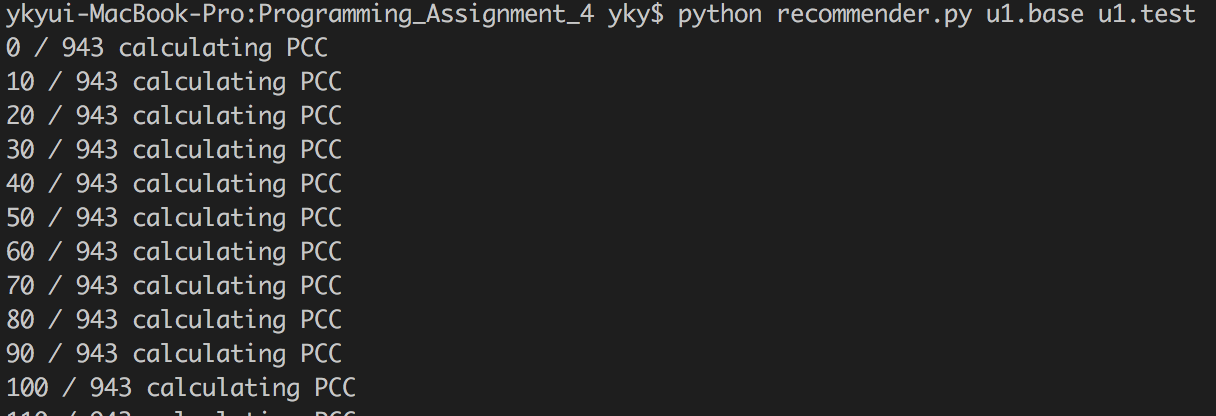
### Checking running time

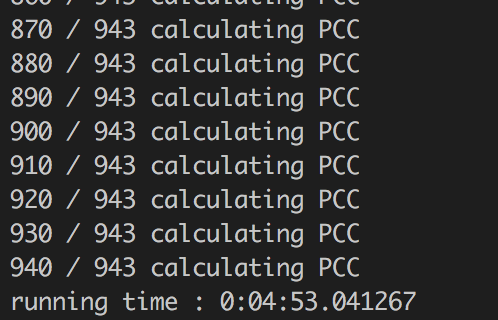


At the end of the process, it shows the running time.

# 4. Instructions for executing the source code

I used OS X, and python. So, you don’t need to compile it into exe file. You can execute it just by python command.





Just input this command in the directory that has recommender.py, base file, and test file.

$ python recommender.py [training\_file\_name] [test\_file\_name]

# 5. Result of test

