**Data Science**

**Assignment4 : Recommender**

2016025532 컴퓨터전공 심수정

**1. Summary of Algorithm**

이 과제는 \*.base에 정의된 user, item, rank, time stamp 등을 활용하여 \*.test file에서 기존의 정의되지 않은 user-item pair에 대해 rank를 예측하는 것을 목표로 합니다.

그것을 위해 저의 code는 CF algorithm을 사용합니다. CF algorithm은 나와 가장 유사한 특성을 가지는 다른 user(neighbor user)들을 찾아, 이들이 평가한 내용을 바탕으로 평가하는 algorithm입니다. 이 경우에는 ‘같은 item에 대해 얼마나 비슷한 평가를 매겼는가’가 기준이 되며, 이를 위해 similarity function을 사용하게 됩니다. 저의 algorithm에서 사용한 similarity function은 아래와 같습니다.

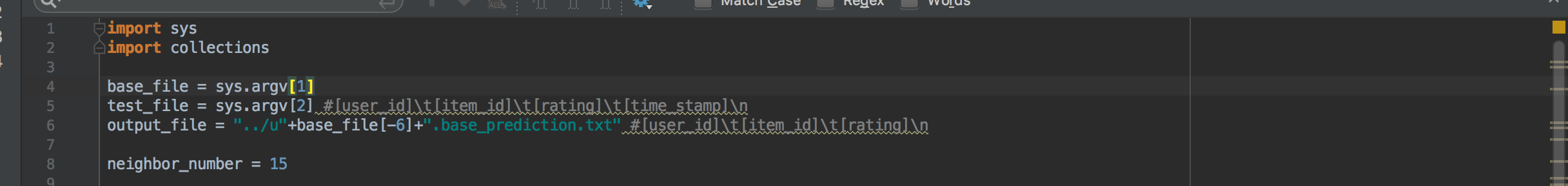
이 함수를 통해 similarity가 0 이상인 user 중 가장 similarity가 높은 user 대략 15명 정도를 neighbor로 가정합니다.

또한 “neighbor user들을 통해 어떻게 평가할까”를 예측하기 위해 아래와 같은 함수를 사용합니다.

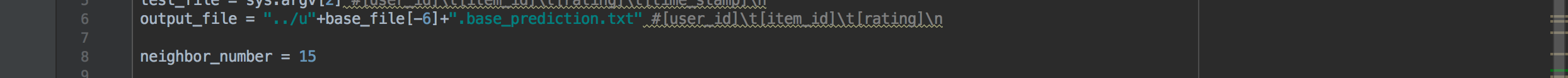
이 때 제가 이 project에서 사용한 weight값은 아래와 같습니다.

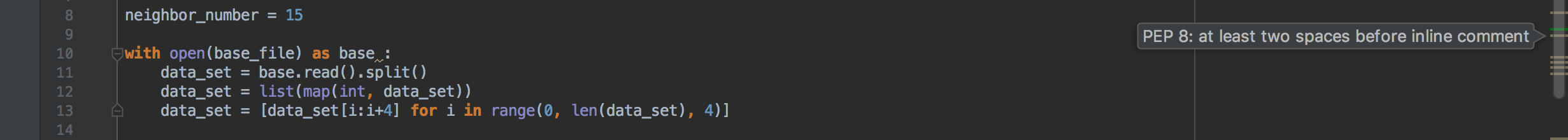
이를 통해서 예측을 하고, 평가를 하여 \*.base\_prediction.txt 파일에 예측된 값과 user, item을 함께 출력하게 됩니다.

**2. Detailed Description of Code**

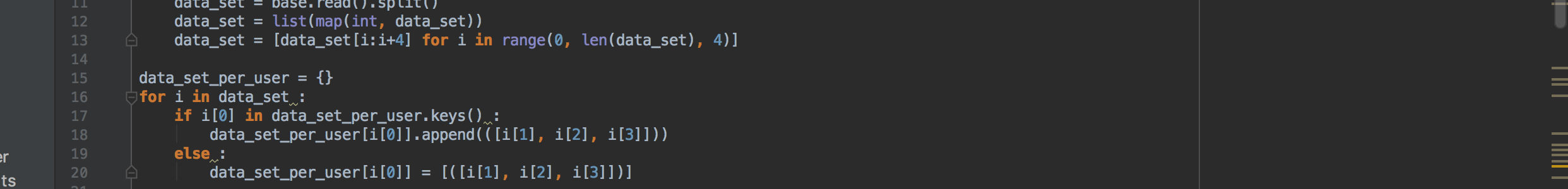


code를 실행할 때 base file, test file을 받아옵니다. 그리고 이를 통하여 결과를 쓸 output file의 이름을 정합니다.

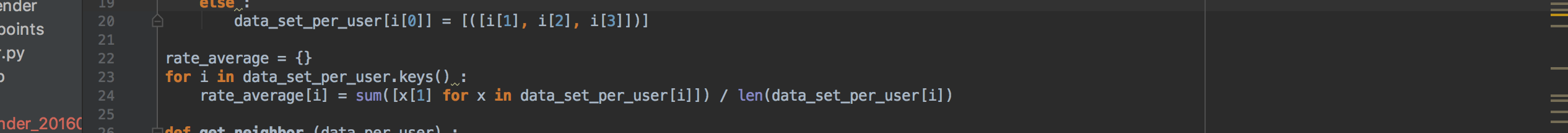


neighbor의 수를 15로 지정합니다.

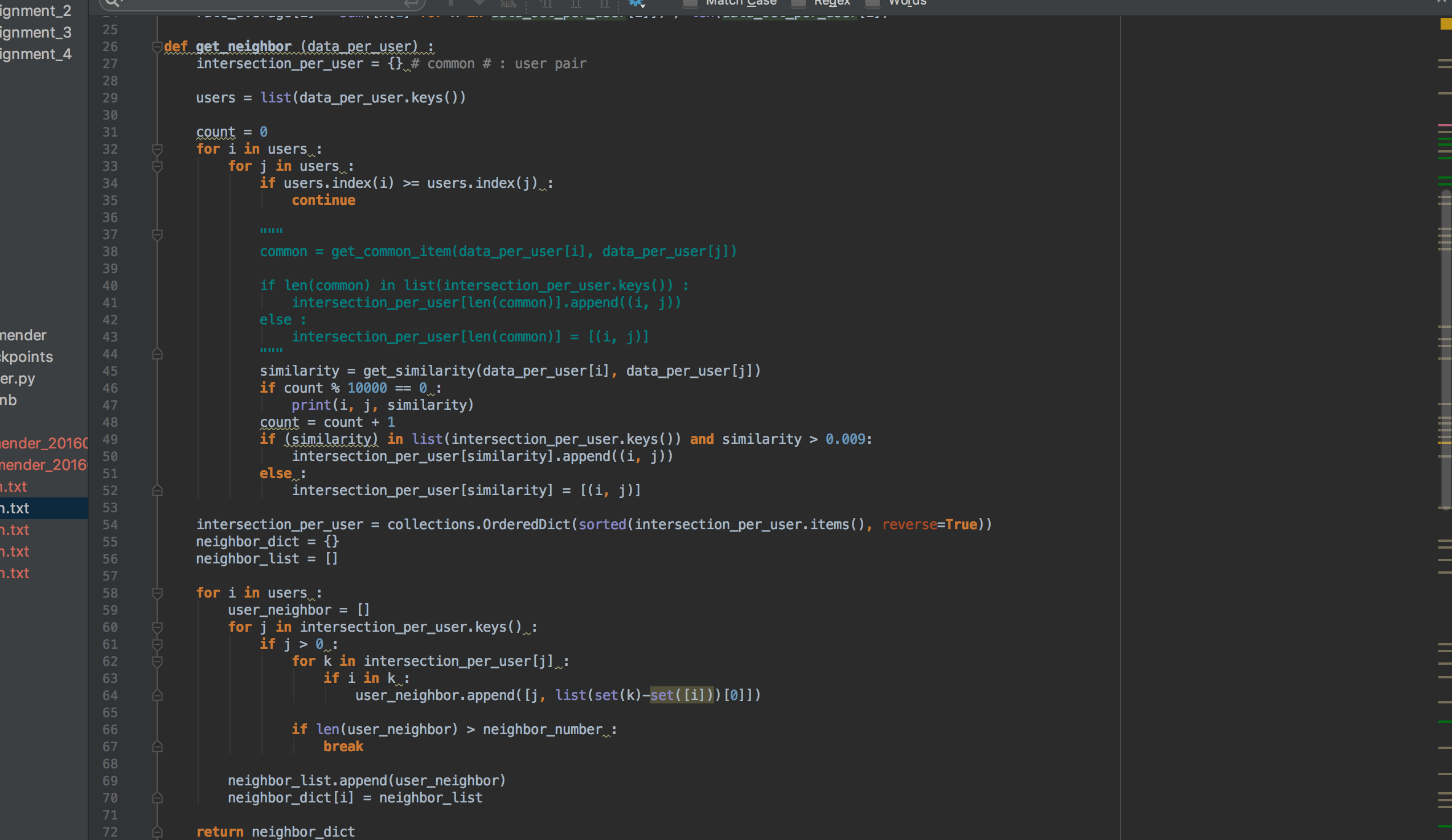
base file에서 user, item, rank, time stamp를 읽어 이를 [[user1, item1, rank1, time stamp1], [user1, item2, rank2, time stamp2]] 의 형태로 data set에 넣어줍니다.



위에서 얻은 data set을 user당 data로 표현합니다. 이는 {user : [[item1, rank1, time stamp1], [item2, rank2, time stamp2]]}의 형태가 됩니다.



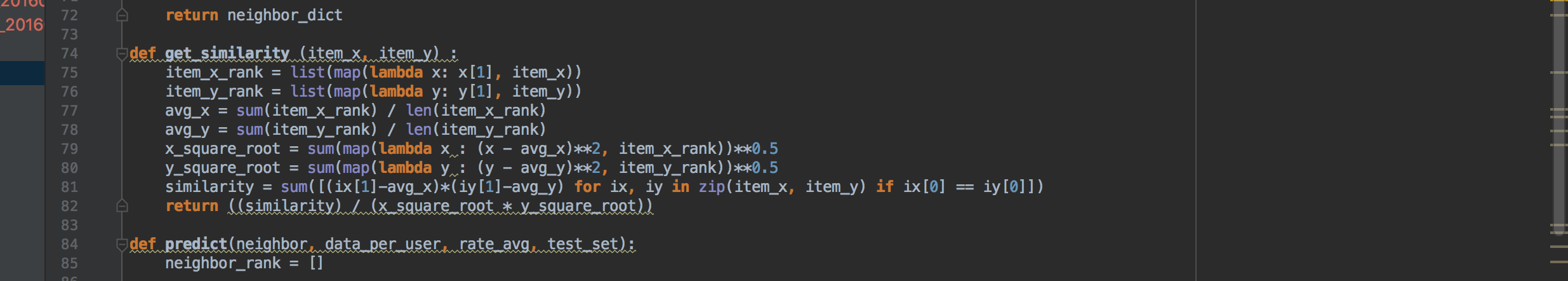
user 가 평가한 item들의 평균이 몇점인지 표시합니다. 이는 {user : rank}의 형태가 됩니다.



간단하게 말하면 다른 user 들과의 similarity를 확인하고, 이중 similarity가 높은 항목들을 이웃으로 넣어 반환하는 함수입니다. 우선적으로 user들의 pair를 simillarity를 key로 저장하는 dictionary를 만듭니다. 이 dictionary는 {similarity : [user pair1, user pair2]}의 형태를 가집니다.

후에는 이를 similarity 순으로 정렬하고, user별 neighbor를 neighbor\_dict에 {user : [[similarity1, neighbor1], [similarity2, neighbor2]]}의 형태로 저장합니다. 만약 같은 similarity에서 이 작업을 한 후 이웃의 수가 neighbor number로 지정된 수를 넘는다면 이 user에 대해서 종료합니다.

최종적으로는 이렇게 형성된 neighbor dict를 return하게 됩니다.

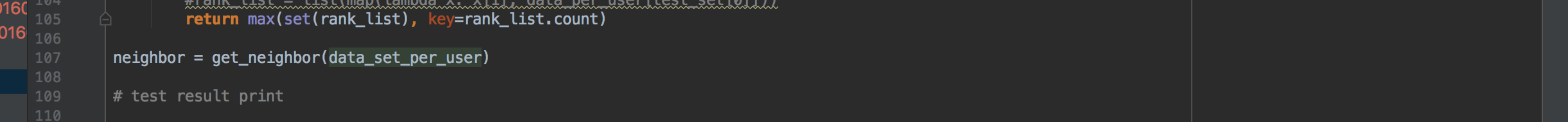


간단하게 similarity를 구해서 return하는 함수입니다. 위에서 언급한 식이였던 “”를 활용해 계산하여 return합니다.

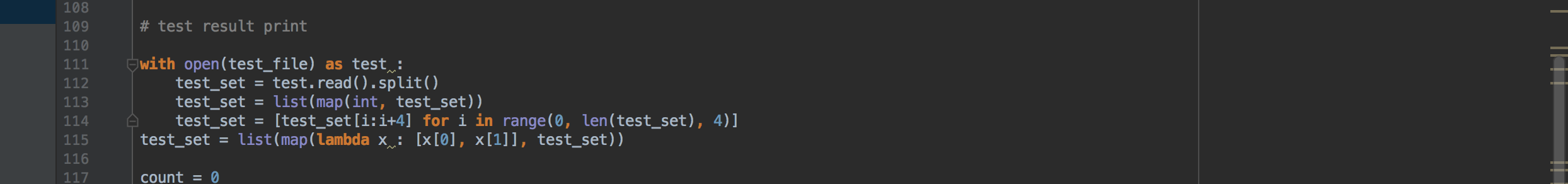
rank를 예측하는 함수입니다. neighbor를 확인하고, 이들 중 평가할 아이템에 대해 rank를 가지고 있는 항목들을 이 neighbor의 평균에 상대적인 값으로 저장합니다.

그리고 나의 평균과 time stamp, similarity를 weight로 더한 평균을 반올림하여 새로 저장합니다. 만약 1보다 작다면 1로, 5보다 크다면 5로 저장합니다.

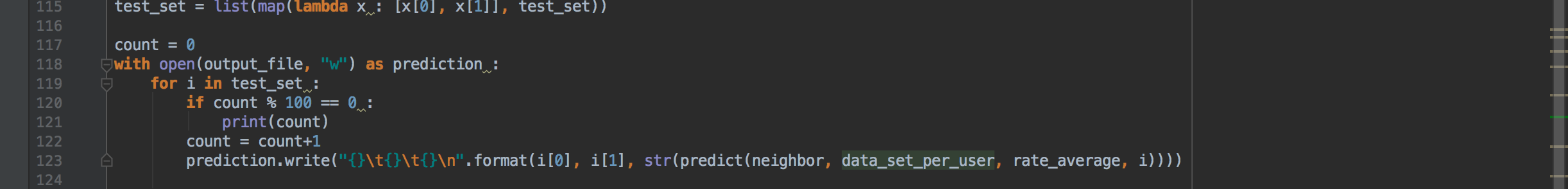
하지만 이웃 중 내가 평가하려는 item의 정보를 가진 이웃이 없다면 내가 가장 많이 준 rank를 부여합니다.



이웃 dictionary를 가져옵니다.



test file을 열여 test set에 data set과 같은 형태로 넣어줍니다.



output file을 열어 예측한 값을 user, item와 함께 적어줍니다. 이의 형태는 “{user}\t{item}\t{predicted rank}\n” 입니다.

**3. Instruction for Compiling Source Code**

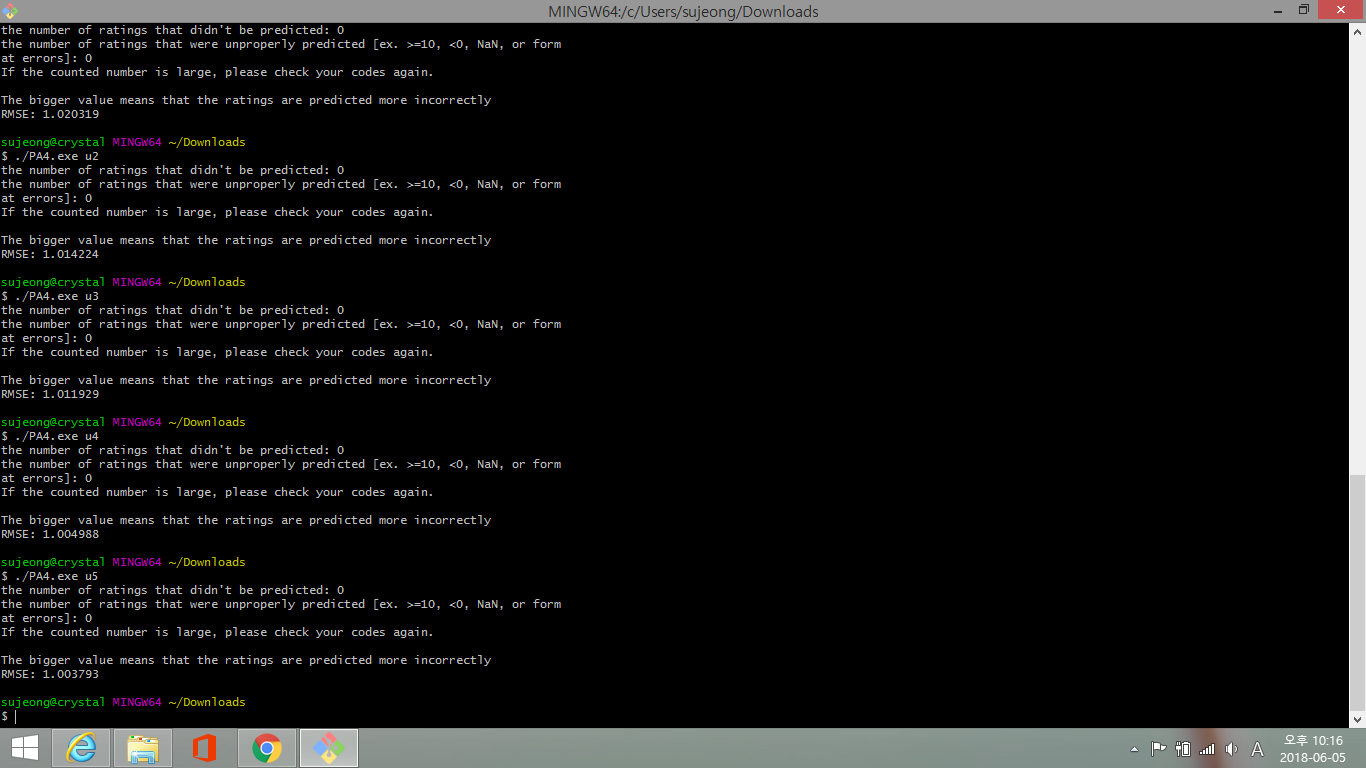
이 코드는 python3을 기반으로 작성되었습니다. 따라서 python3가 설치되어 있어야 합니다.

“**python3 recommender.py [base file] [test file]**”의 형태로 complie, 실행하면 됩니다. output file은 실행한 현재 폴더의 상위 폴더에 생성됩니다.

**4. Other Sepcification of Implementation and Testing**

우선적으로, 실행 시간이 매우 깁니다. 하지만 실행을 하는 동안은 현재 진행중인 user-user similarity, test set이 몇 번째 등인지를 계속 일정 간격으로 출력합니다.

아래 그림은 주어진 test file, base file과 주어진 test program을 바탕으로 성능을 측정한 결과입니다.



대부분 1에 근접하는 RMSE 값을 가지는 것을 확인할 수 있습니다.