**Artificial Intelligence**

Assignment2 : Word Clustering

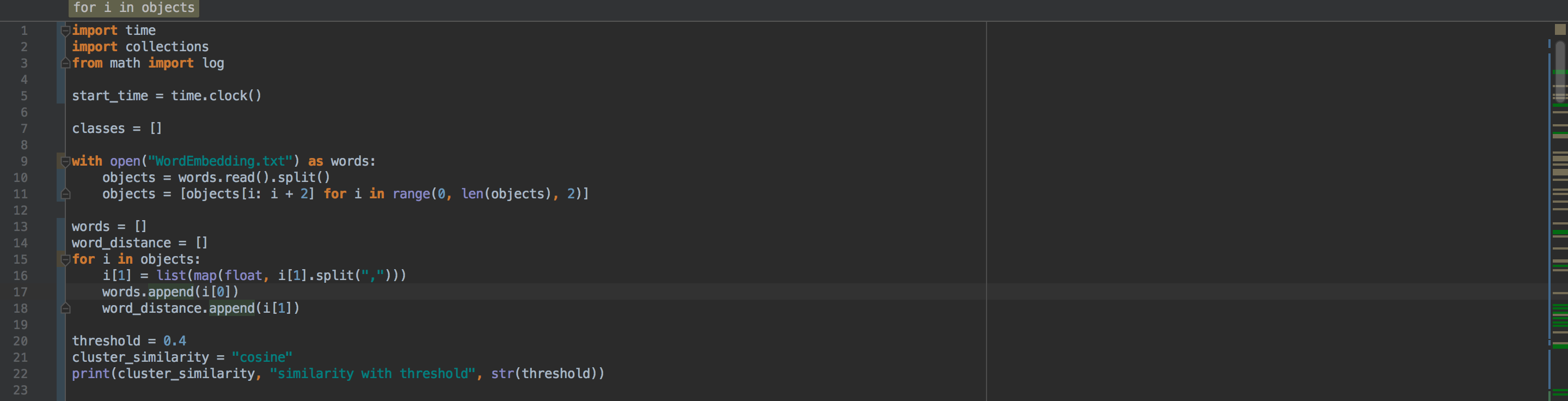
2016025532 컴퓨터전공 심수정

**1. Code Compile Version**

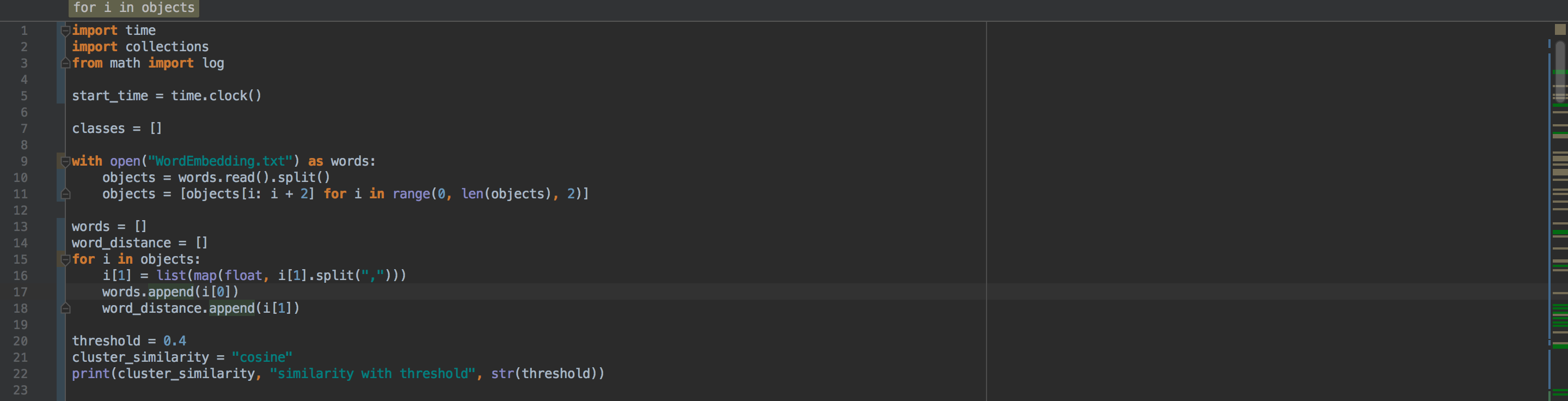
모든 코드는 python3.6 version을 기준으로 작성되었습니다. 따라서 python3 version이 설치되어 있지 않다면 이를 설치할 필요가 있습니다.

“**python3 [filename]**”을 입력한다면 해당 코드를 컴파일, 실행할 수 있습니다.

**2. Code Review**



start time을 확인하고, WordEmbedding.txt를 열어 data를 꺼내옵니다. 이때 data는 objects라는 list에 저장되는데, [[word, position], [word, position], …]의 형태로 저장하게 됩니다. 또한 word, position을 각각 words, word\_distance list에 저장하고, position을 float들의 list로 변경합니다.



threshold를 저장하고, cluster\_similarity에 어떠한 similarity를 사용할 것인지(e.g. cosine similarity, euclidean similarity) 저장합니다. 그리고 어떠한 similarity로 어떤 threshold와 함께 clustering을 진행할지 출력합니다.



similarity matrix를 계산하여 얻습니다. 각 classese list 안에 담겨 있는 cluster 간의 similarity를 계산하여 이 행렬에 집어넣습니다. 단, 이 행렬의 경우 하삼각행렬의 모양을 가집니다.



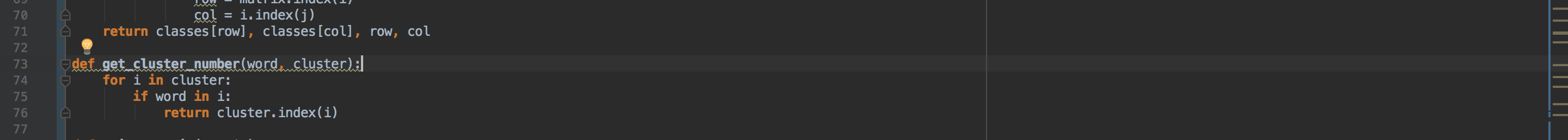
similarity를 계산합니다. 우선적으로 item set x와 item set y가 같은 경우 무조건 euclidean의 경우 0, cosine의 경우 1을 return하고, 더이상 계산하지 않습니다.

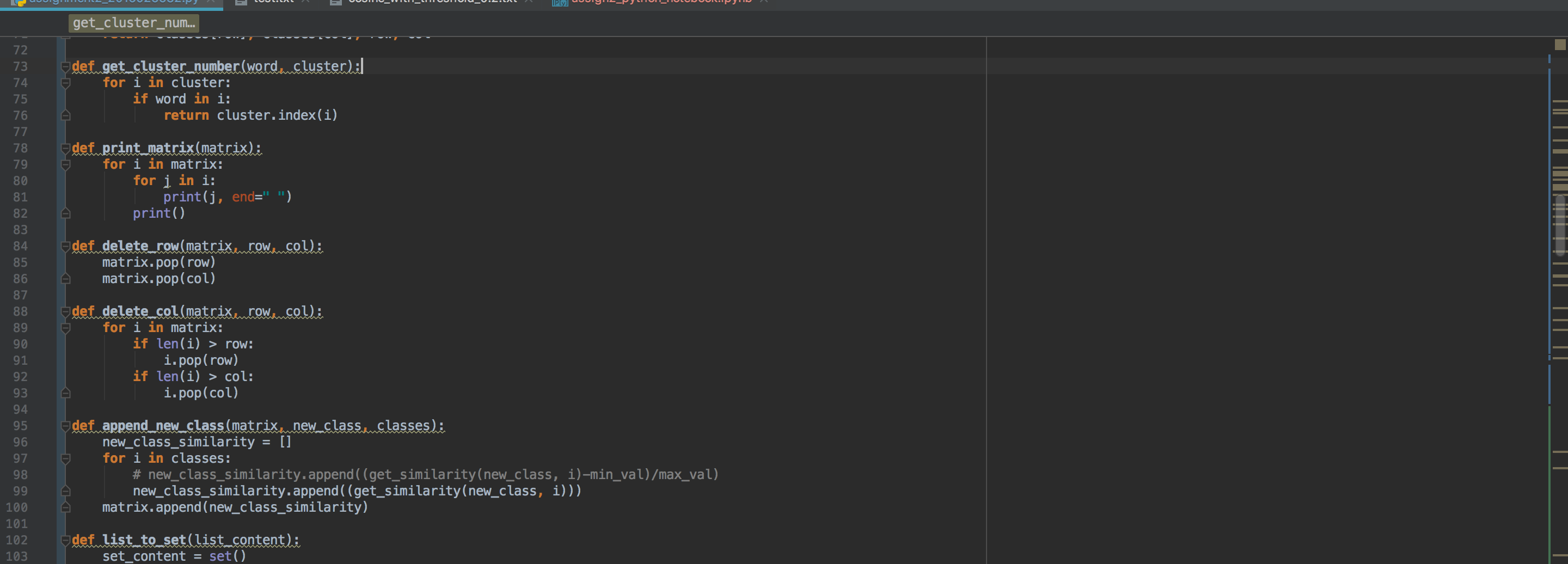
후에 item set x, item set y의 모든 item 조합을 보며 euclidean, cosine similarity를 각각 계산하여 similarity list에 넣어줍니다.

최종적으로, similarity list에 아무것도 없다면 euclidean의 경우 0, cosine의 경우 1을 return합니다.

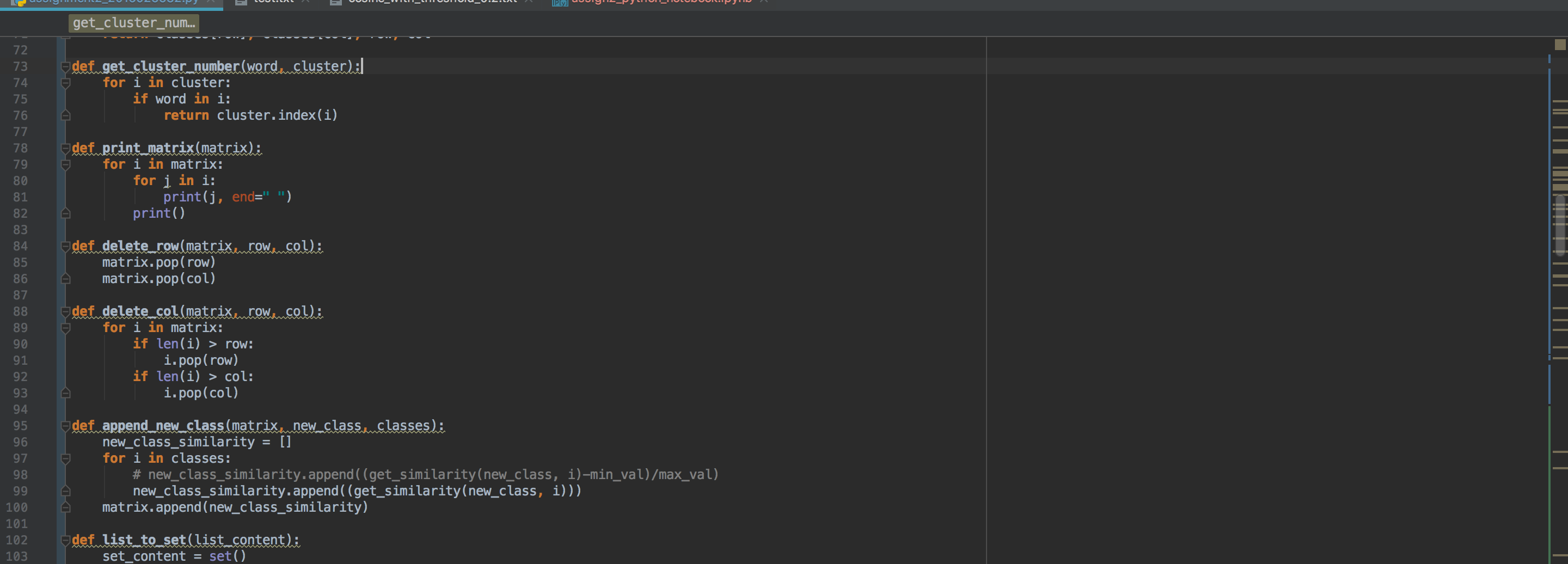


similarity가 가장 높은 cluster를 찾습니다. 그리고 classes list에서 similarity가 가장 높은 2개의 cluster를 return하고, 이게 matrix에서 어떤 row, 어떤 col에 해당했는지를 return합니다.

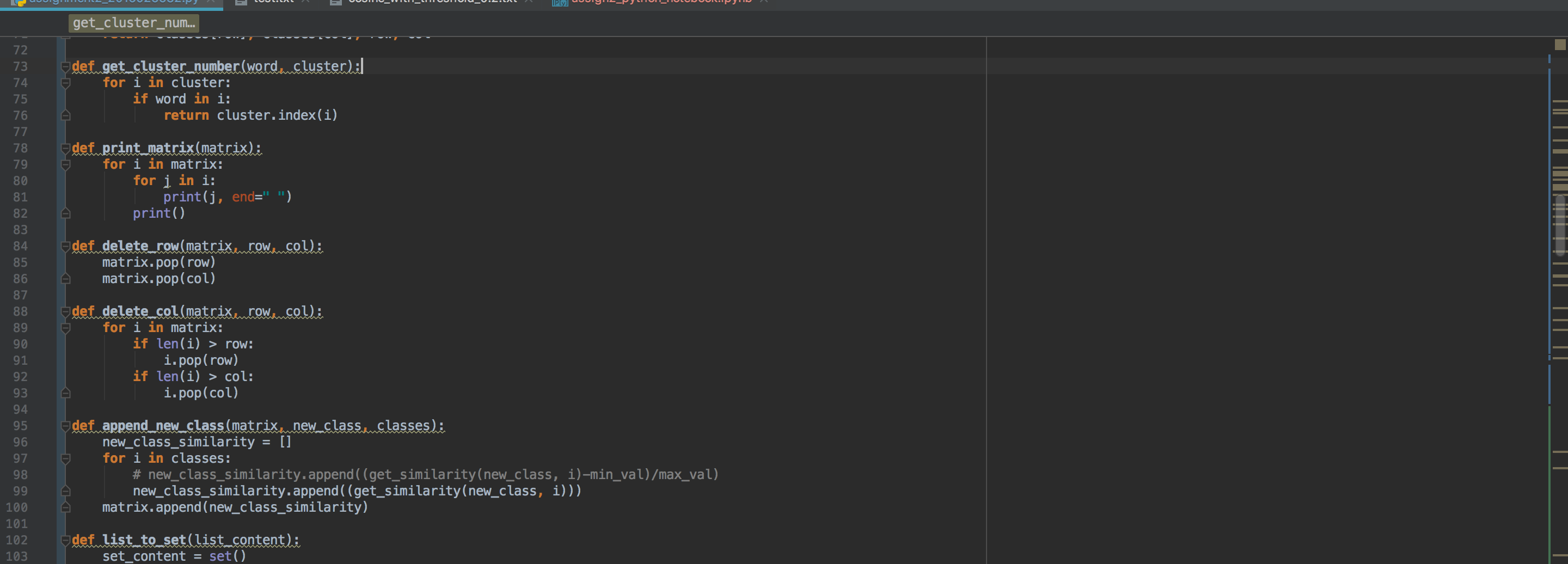


현재 word가 몇번째 cluster에 들어가있는지 찾아 index를 return합니다.

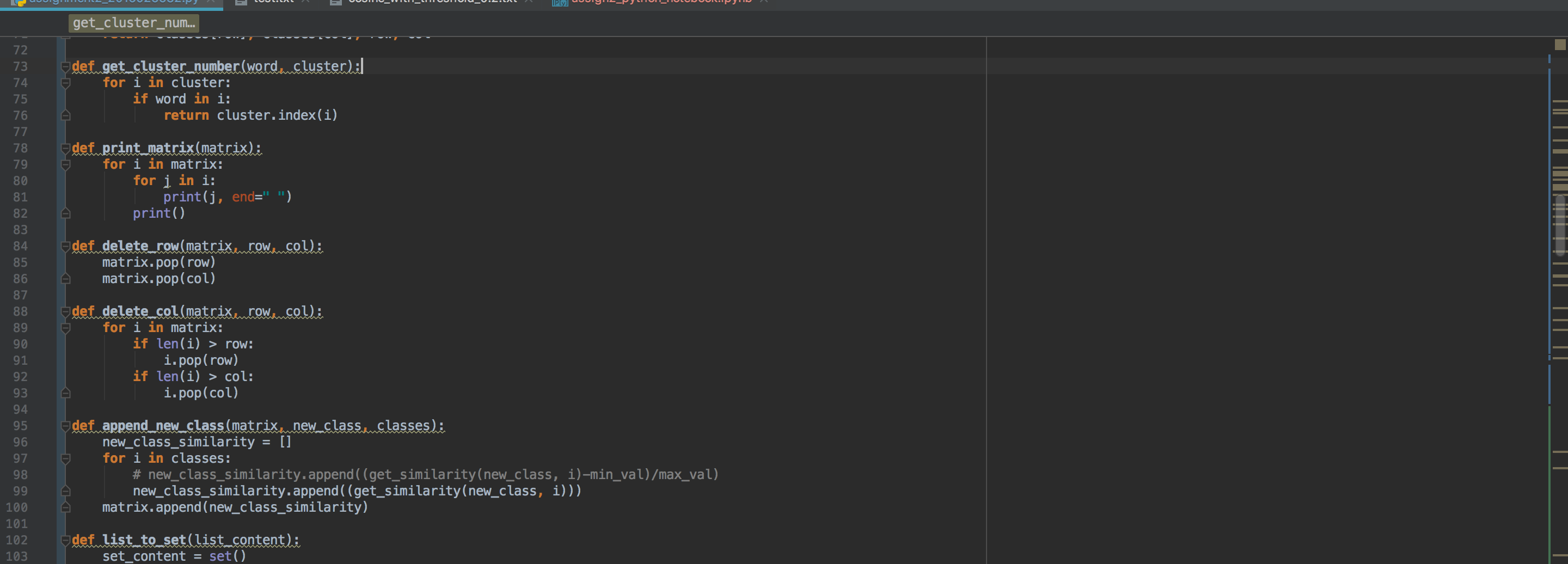
현재 matrix를 출력합니다.



row, col에 해당하는 matrix의 row를 삭제합니다.

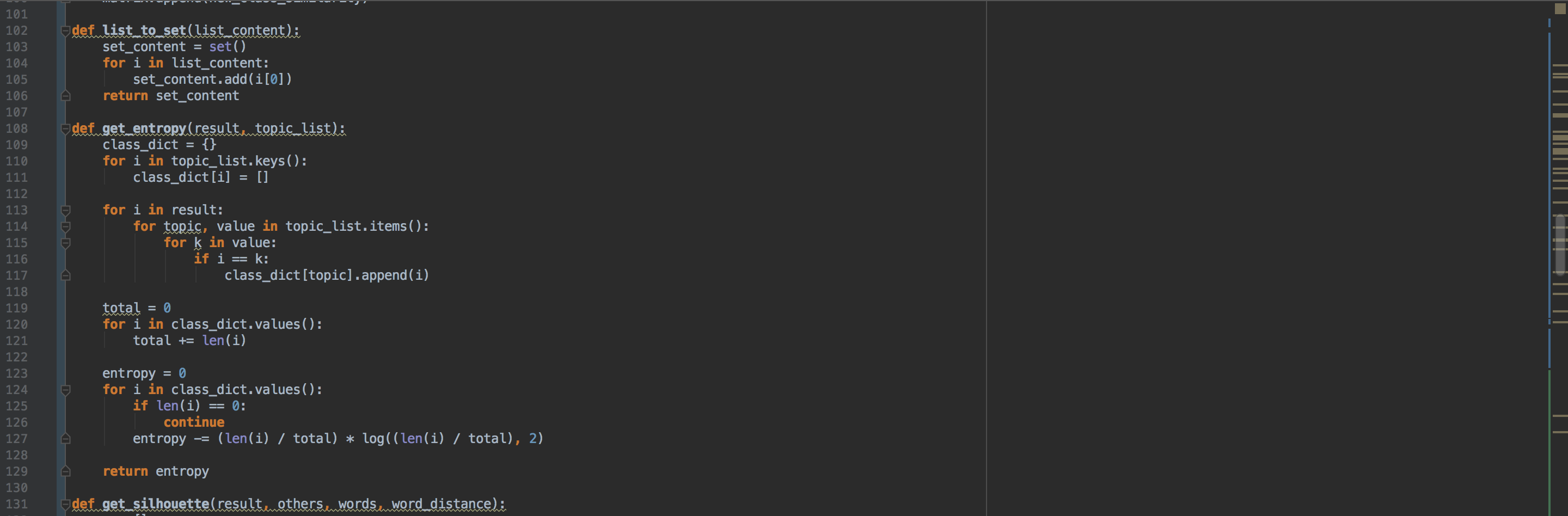


row, col에 해당하는 matrix의 column이 있다면, 삭제합니다.



새로운 cluster similarity를 기존의 matrix에 추가합니다. 

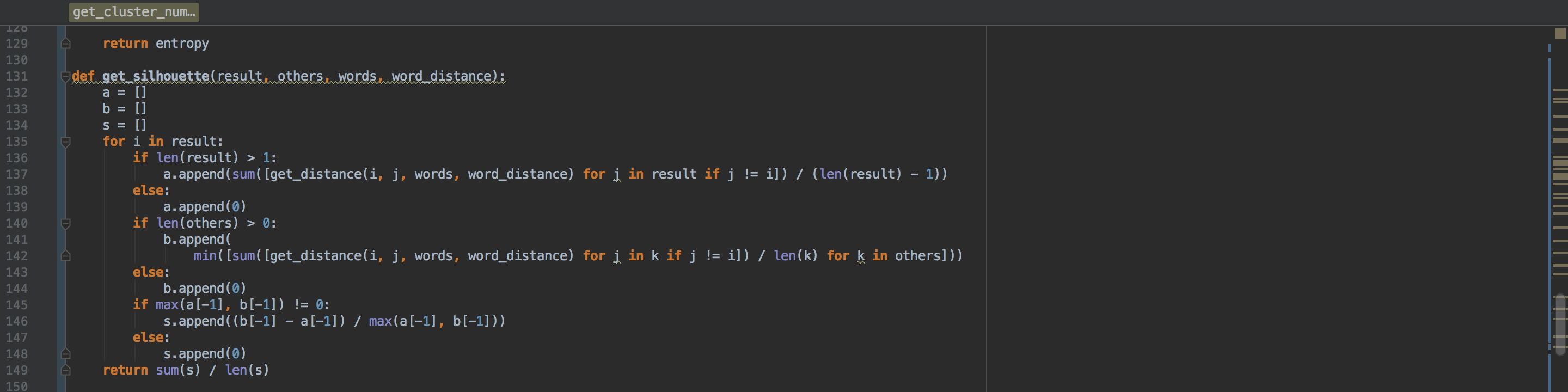
cluster내에 존재하는 word만 set으로 만들어 return합니다.



entropy를 계산하는 함수입니다. Class와 이에 해당하는 word를 저장하는 dictionary인 class\_dict를 만들어 이에 정확한 word, class가 입력된 topic\_list의 key만큼 key를 빈 list와 mapping하여 넣어둡니다.

그리고 result를 확인하여 각 class에 해당하는 dictionary word list에 들어가도록 해주고, 같은 cluster내에 있는 word가 몇개인지 확인합니다.

마지막으로 entropy를 계산하여 return합니다.

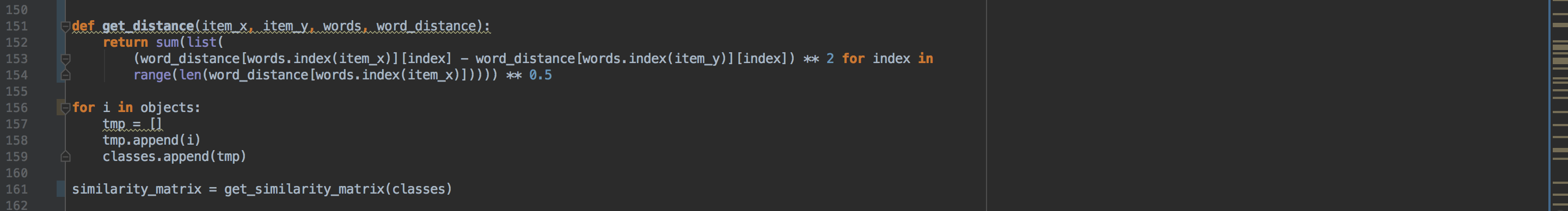


silhouette 지표를 계산합니다. 우선 a(i)에 해당하는 값을 계산하여, list a에 넣고, b(i)에 해당하는 값을 계산하여 b(i)에 넣습니다.

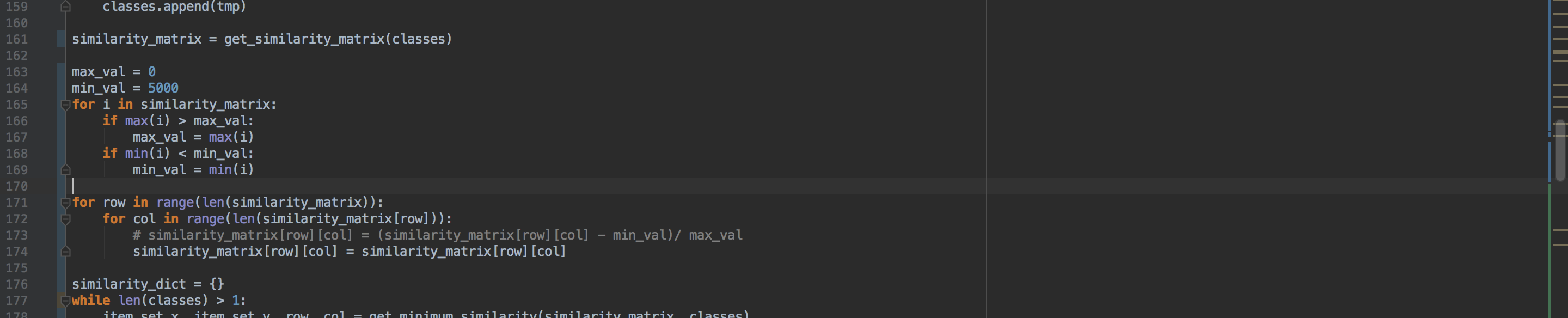
a(i), b(i)는 각각 같은 cluster 내의 다른 item과 i간 거리의 평균, 다른 cluster의 속한 요소들 간 거리의 평균을 다른 모든 cluster에 대해 구한 후, 가장 작은 값을 선택한 것입니다.

s(i)는 로 표현되는 값입니다. 이는 해당 cluster내의 모든 word에 대해 계산됩니다. 그리고 구해진 silhouette 값의 평균을 구해 return합니다.

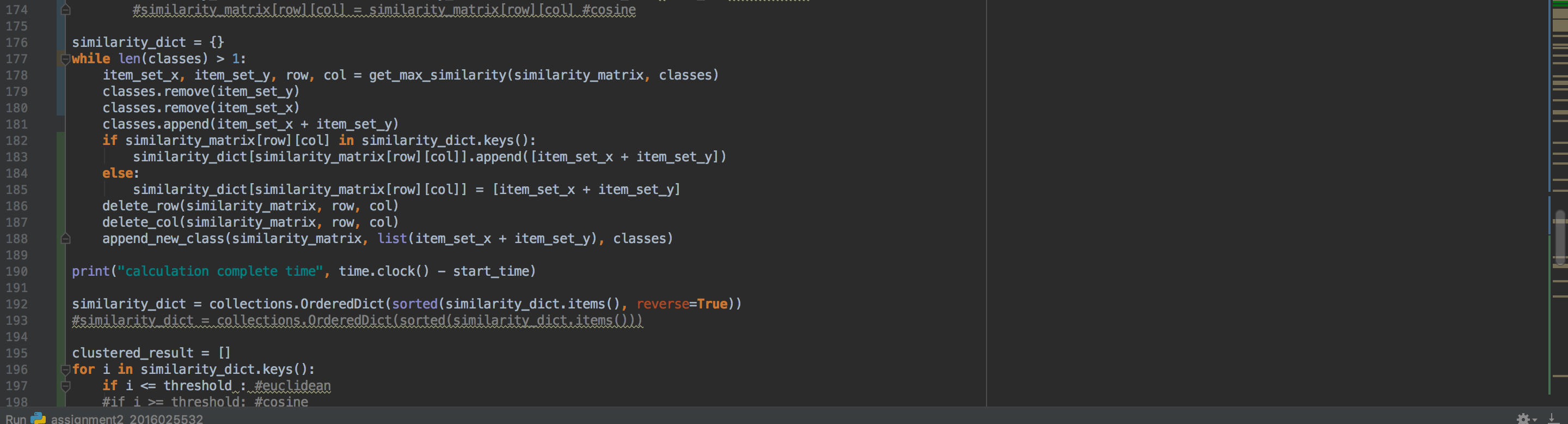


item\_set\_x, item\_set\_y 간의 euclidean distance를 구하여 return합니다.

cluster를 [[item], [item], …]의 형태로 표현하기 위해 objects의 item을 list에 넣고, 이를 다시 cluster list에 넣어줍니다.



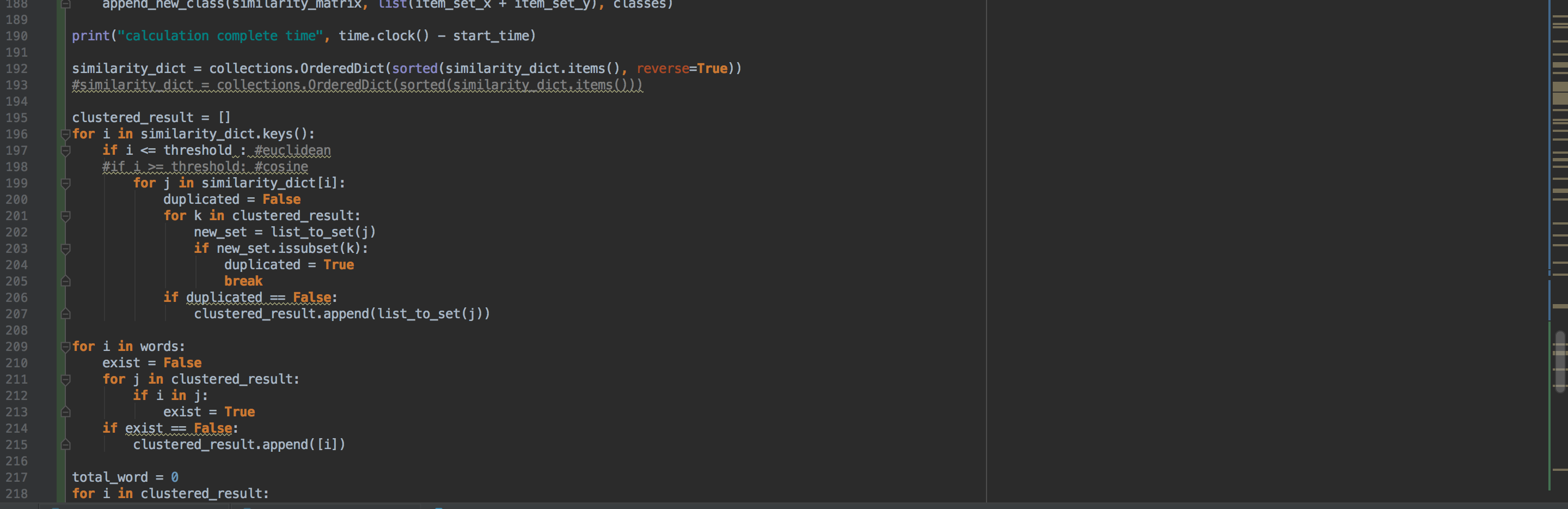
similarity matrix를 먼저 구합니다. 그리고 euclidean인 경우 max값, min값을 구하여 의 형태로 만들어 0~1 사이의 값으로 나타날 수 있도록 합니다. cosine의 경우는 그대로 유지합니다.



complete link clustering을 수행합니다. 우선 가장 similarity가 높은 cluster 2개를 찾고, 이를 classes list에서 제거합니다. 또한 이에 해당하는 similarity matrix 값도 지우고, classes, similarity matrix에 이들이 합쳐진 cluster, 이에 대한 similarity를 계산하여 넣습니다.

그리고 similarity dict에 거리를 key로 cluster list안의 cluster로 넣습니다. 이는 {distance : [[clsuter1], [cluster2]]}의 형태로 표현됩니다.

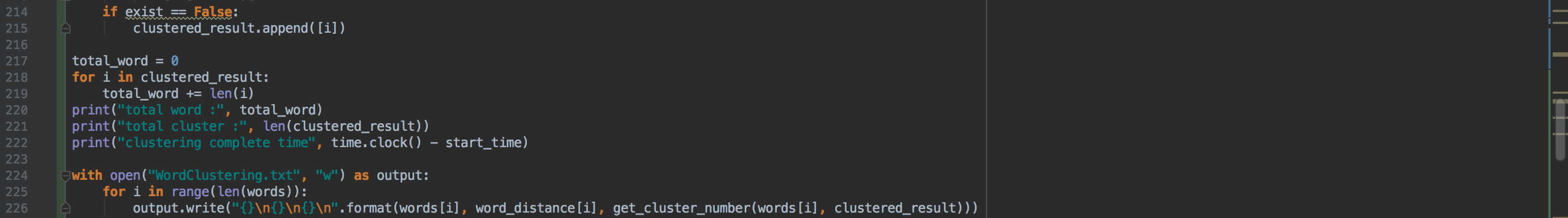
그리고 여기까지의 과정을 하는 동안 몇초의 시간이 소요되었는지 출력합니다.



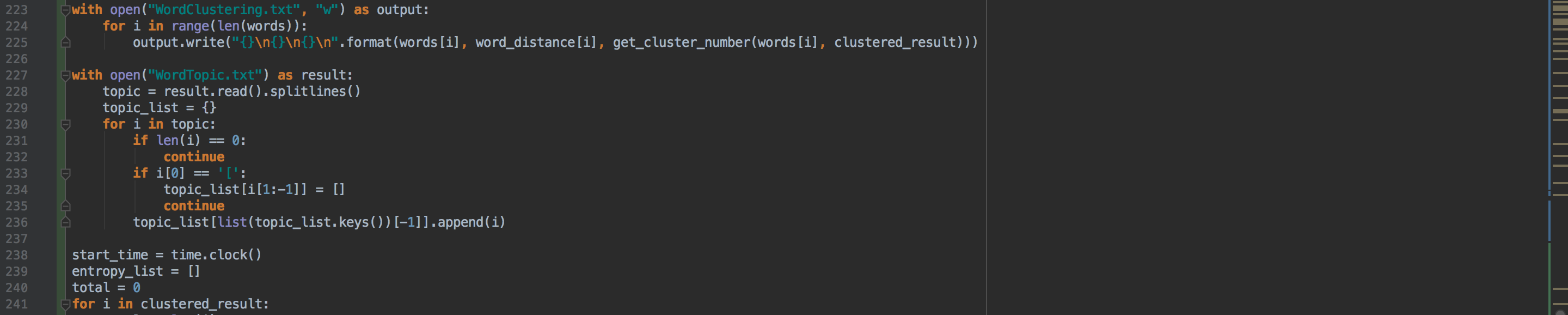
우선적으로 eulidean similarity를 활용하는 경우라면 similarity의 내림차순으로, cosine을 이용하는 경우라면 similarity 오름차순으로 정렬합니다.

또한 similarity\_dict를 보며 similarity가 threshold를 만족하는 경우를 찾고, 이미 이 cluster의 item이 기존의 clustered\_result로 분류된 cluster에 속하지는 않는지 확인 후 속하지 않는다면 clustered\_result에 추가합니다.

마지막으로, 기존의 word들을 확인하며 현재 word가 어떠한 cluster에도 속해 clustered\_result에 들어가지 못했다면, word만 존재하는 cluster를 만들어 clustered\_result에 넣어줍니다.



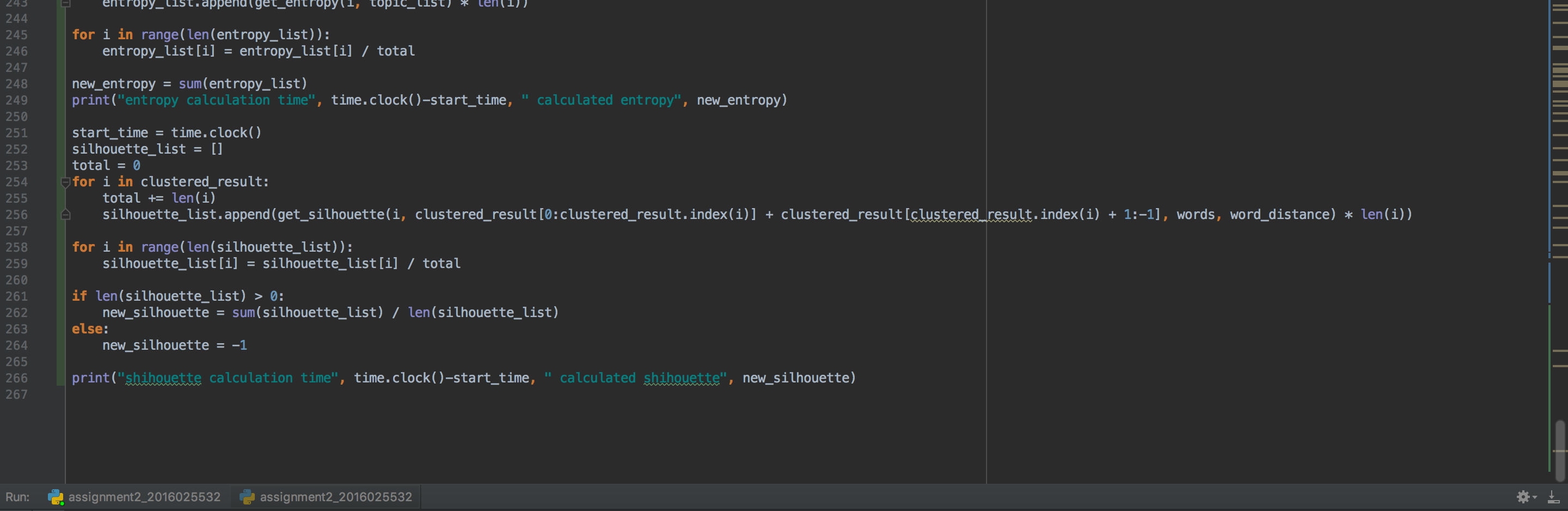
전체 단어가 다 들어갔는지 확인하기 위해 clustered\_result의 전체 단어수를 확인하고, cluster의 수도 확인합니다. 그리고 이 과정을 위해 몇초의 시간이 소요되었는지 확인합니다.



처음에 읽은 word와 vector값을 clustering을 통해 얻은 결과를 WordClustering.txt를 열어 \n 단위로 씁니다.

WordTopic.txt를 열어 정확한 cluster당 word들을 읽습니다. 그리고 이를 topic\_list에 {cluster : [word, word, …]}의 형태로 집어넣습니다.

entropy를 각 cluster마다 계산해 해당 cluster의 길이를 곱하고, 이의 평균을 전체적인 entropy로 지정합니다. 가 전체적인 entropy가 되는 셈입니다. 그리고 이를 entropy를 계산하는데 걸리는 시간과 함께 출력합니다.



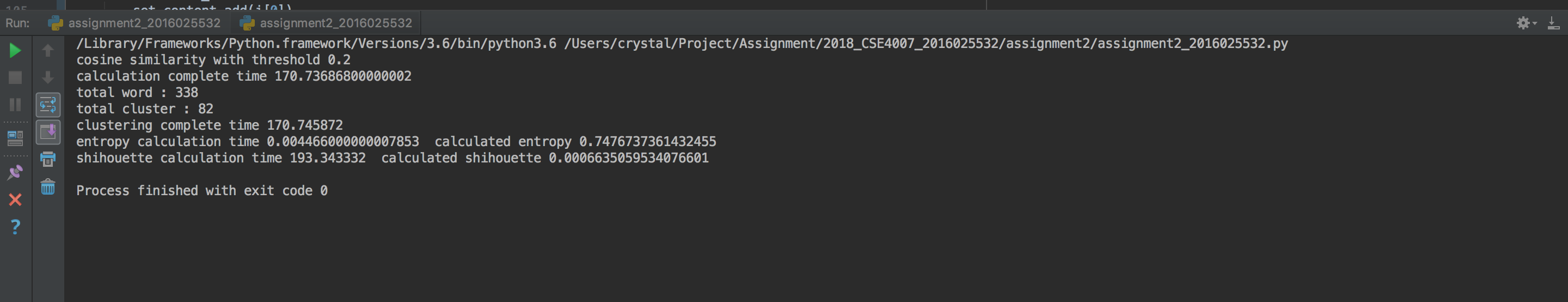
각 cluster마다 silhouette 값을 계산합니다. 그리고 이를 silhouette list에 넣고, 로 가중치 평균을 구합니다. 그리고 구해진 값을 출력하고, 이를 계산하는데 걸린 시간도 출력합니다.

**3. Result**

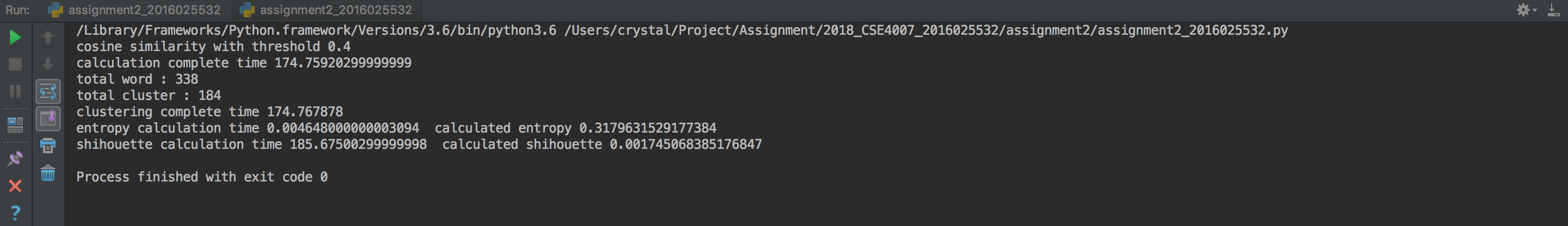
**1) Cosine**

cosine의 경우 계산된 값이 클수록 similarity가 크기에 threshold가 클수록 더 유사한 cluster만 허용합니다.

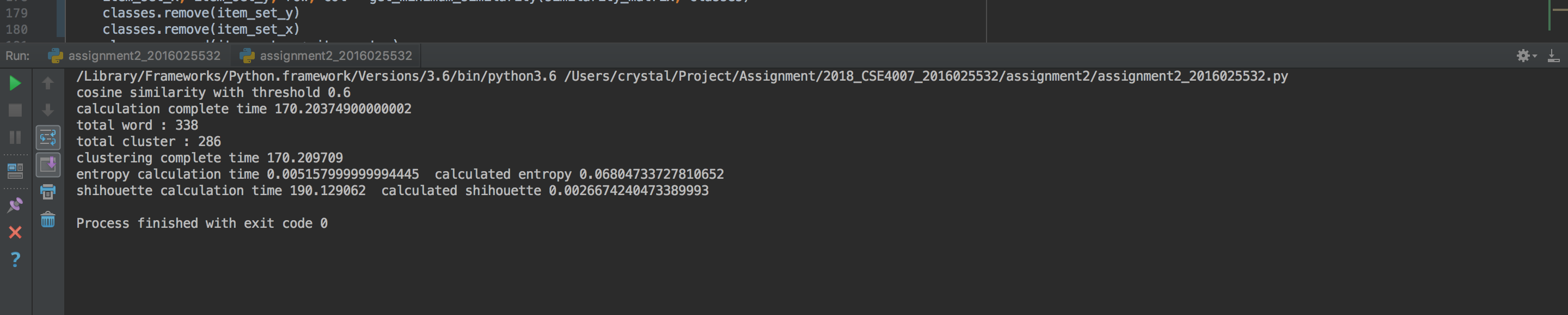
(1) threshold = 0.2



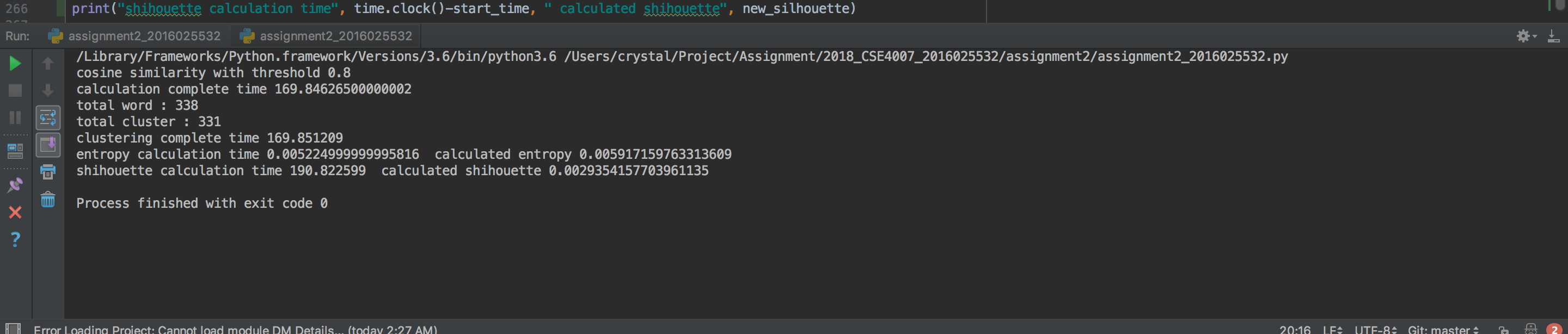
(2) threshold = 0.4



(3) threshold = 0.6



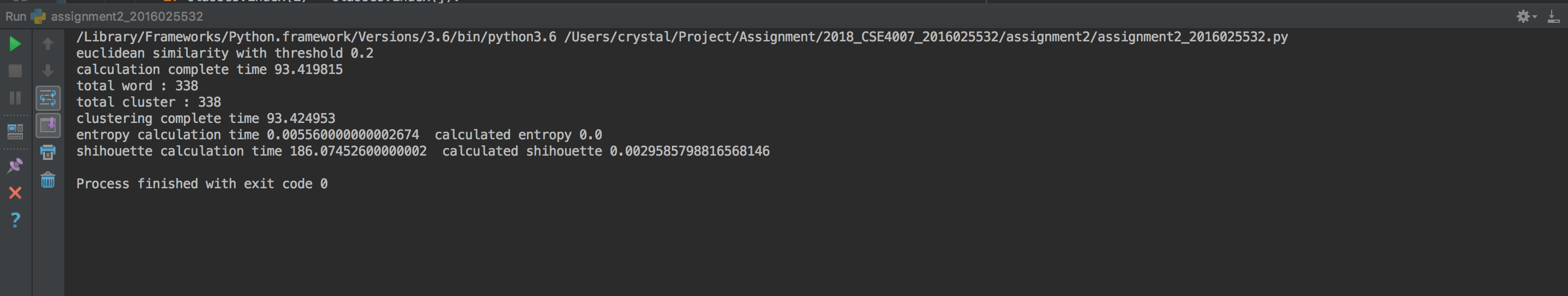
(4) threshold = 0.8



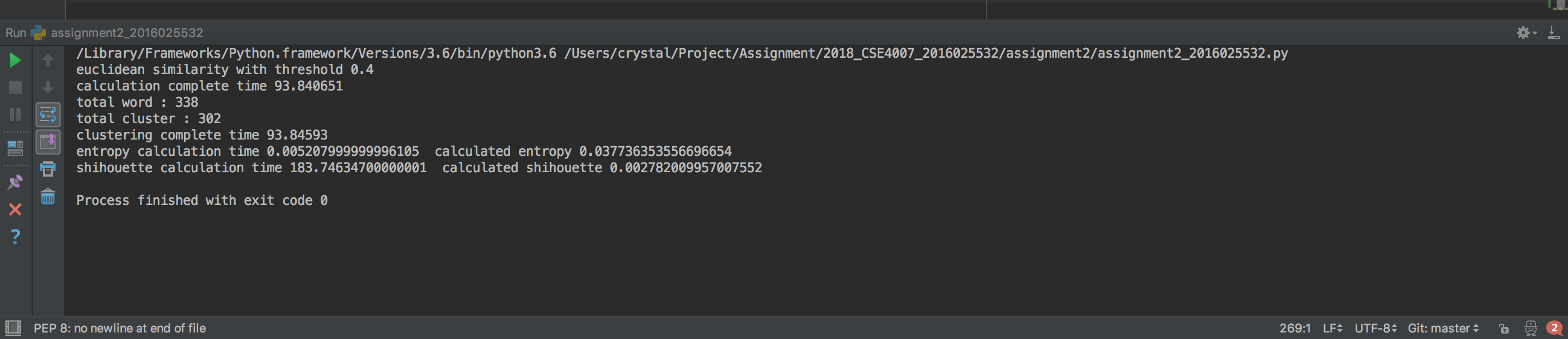
**2) Euclidean**

euclidean의 경우 계산된 값이 similarity가 클수록 작기에 threshold가 작을수록 더 유사한 cluster만 허용합니다.

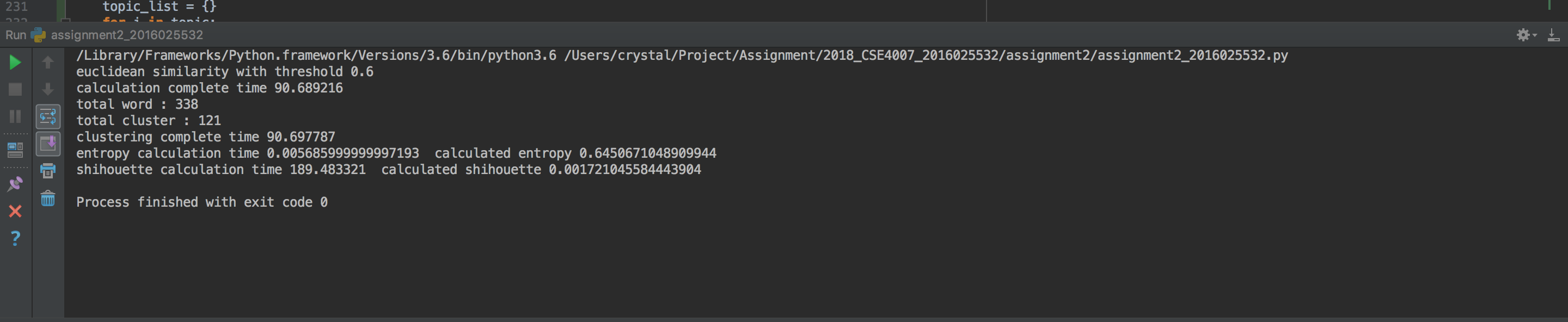
(1) threshold = 0.2



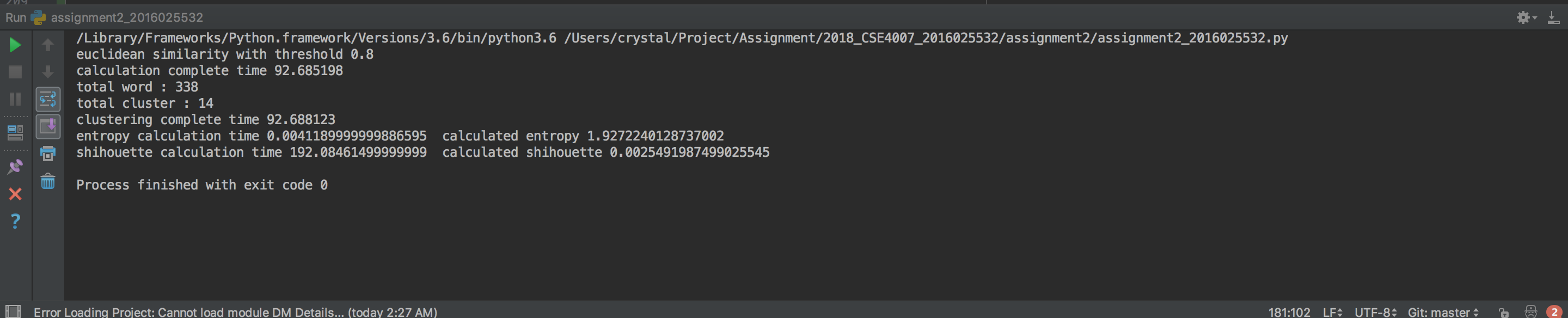
(2) threshold = 0.4



(3) threshold = 0.6



(4) threshold = 0.8



**3) Summary**

|  |  |  |  |
| --- | --- | --- | --- |
| threshold |  | Cosine Similarity | Euclidean Similarity |
| 0.2 | Cluster Number | 82 | 338 |
| Entropy | 0.748 | 0.0 |
| Silhouette | 0.000663 | 0.002959 |
| 0.4 | Cluster Number | 184 | 302 |
| Entropy | 0.318 | 0.038 |
| Silhouette | 0.001745 | 0.002782 |
| 0.6 | Cluster Number | 286 | 121 |
| Entropy | 0.068 | 0.645 |
| Silhouette | 0.002667 | 0.001721 |
| 0.8 | Cluster Number | 331 | 14 |
| Entropy | 0.006 | 1.927 |
| Silhouette | 0.002935 | 0.002549 |

**4. Analysis**

cosine의 경우 threshold가 높을수록 더 유사한 cluster만 허용합니다. euclidean의 경우 similarity가 낮을수록 더 유사한 cluster만 허용합니다. 그래서 cosine에서는 threshold가 높아질수록 entropy가 감소하고, silhouette가 증가하는 반면, euclidean에서는 threshold가 높아질수록 entropy가 증가하고, silhouette이 감소합니다.

보통 entropy가 낮을수록 비슷한 성질의 것들끼리 clustering이 잘 되었다고 말할 수 있습니다. 이 경우 cosine에서는 threshold가 0.8 경우, euclidean에서는 threshold가 0.2인 경우가 가장 좋습니다. 이 지표만 봤을 때 가장 좋은 것은 entropy가 0인 Euclidean similarity, threshold 0.2인 경우입니다. 이 경우도 entropy와 마찬가지로 가장 clustering이 잘 되었다고 보는 경우는 euclidean similarity, threshold 0.2인 경우입니다.

silhouette가 1에 가까울수록 비슷한 성질의 것들끼리 clustering이 잘 되었다고 할 수 있습니다. 이 경우는 cosine에서 threshold가 0.8인 경우, euclidean에서 threshold가 0.2인 경우가 가장 좋습니다.

하지만 cluster가 너무 세세하게 분할되어, word 1개가 된다면 clustering을 하는 의미가 없어집니다. 비슷한 Word를 묶어 같은 특성의 아이를 찾고 싶은 건데, 그렇지 못하는 경우이기 때문입니다. 하지만 위에서 봤듯이, entropy, silhouette 지표에서는 모든 word를 각각 하나의 cluster로 묶는 Euclidean similarity, threshold 0.2를 지지합니다.

그래서 cluster의 수도 중요한 지표입니다. 이것만 고려하는 경우는 cosine에서 threshold 0.2인 경우가 가장 좋고, euclidean에서 threshold가 0.8인 경우가 가장 좋습니다. 전체적으로 봤을 때는 압도적으로 Euclidean similarity, threshold 0.8인 경우가 좋습니다.

이들을 종합하여 봤을 때, cosine similarity에서 threshold가 0.6인 경우가 가장 좋다고 판단됩니다. Entropy도 높지 않고, silhouette 지표도 높은 편이고, cluster의 수가 높은 편인 것이 아쉽기는 하지만 그 이상이 되면 정확도가 떨어지리라 생각됩니다. 그래서 어느 정도의 정확도를 고려하며 clustering을 하기 위해서는 이게 가장 좋다고 판단됩니다.