Bryan Carey
CS432 Assignment 7

Q1) Find 3 users who are closest to you in terms of age, gender, and occupation. For each of those 3 users:

- what are their top 3 favorite films?

- bottom 3 least favorite films?

Based on the movie values in those 6 tables (3 users X (favorite +least)), choose a user that you feel is most like you. Feel free to note any outliers (e.g., "I mostly identify with user 123, except I did not like ``Ghost'' at all").

This user is the "substitute you".

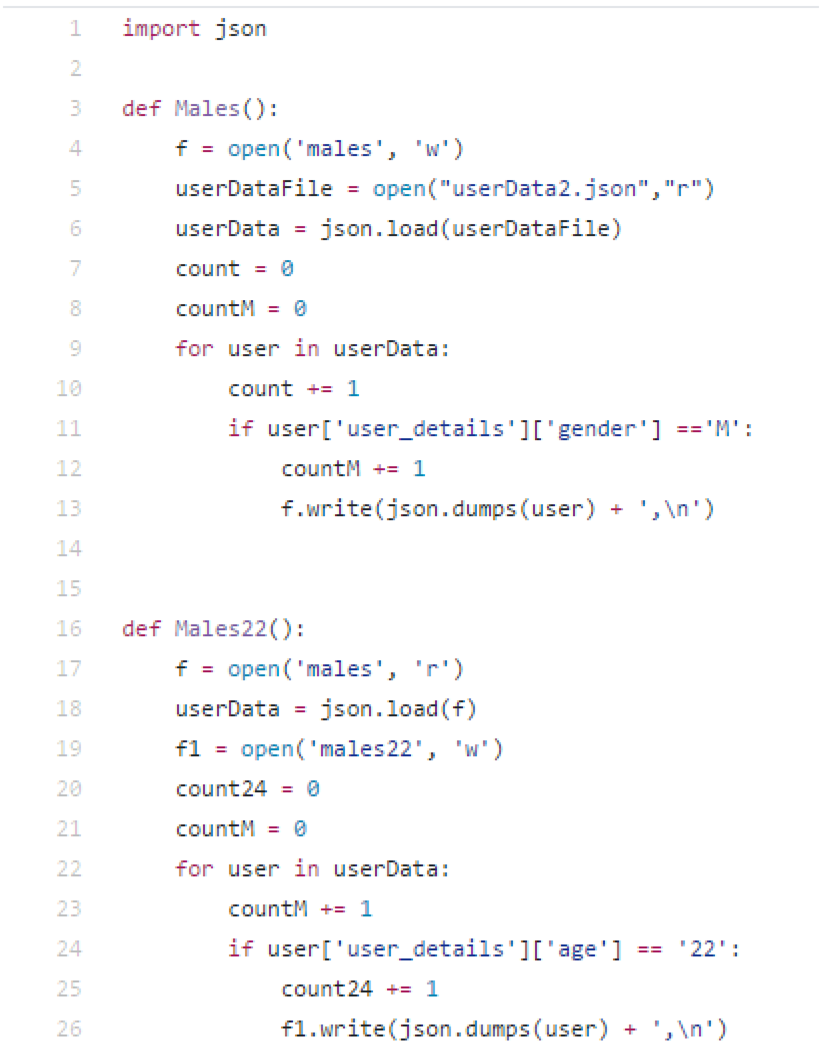
For my first attempt at problem 1, I utilized the recommendations.py file provided from the PCI text I found on GITHUB at https://github.com/uolter/PCI/tree/master/chapter2 . I created my first .py file, similarUsers.py and imported recommendations.py as a module. Doing so I was able to generate a list of users however, working with a text file would be a little more difficult when deducing the correlation in my opinion, so I scrapped this approach. Below is my first attempt and the output.

```
import recommendations as re
f = open('similarUsers.txt', 'w')
input = open("u.user", 'r')

data = re.loadMovieLens

users = []
#not acessing the input file
for line in input:
    u = (id, age, gender, occupation, zip) = line.split('|')[0:5]

    if (u[1] == '21' and u[2] == 'M' and u[3] == 'student'):
        users.append(u)
for item in users:
    print>> f, item
```

```
['81', '21', 'M', 'student', '21218\n']
['259', '21', 'M', 'student', '48823\n']
['276', '21', 'M', 'student', '95064\n']
['323', '21', 'M', 'student', '19149\n']
['542', '21', 'M', 'student', '60515\n']
['725', '21', 'M', 'student', '91711\n']
['923', '21', 'M', 'student', 'E2E3R\n']
['928', '21', 'M', 'student', '55408\n']
```

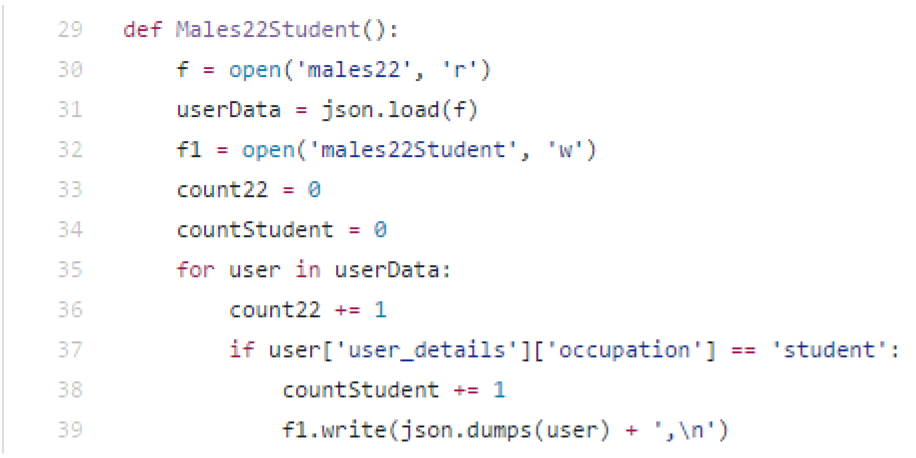
Bryan Carey
CS432 Assignment 7

I decided to rethink my approach. I spoke to anther classmate who had made use of JSON to solve the assignment. From that point I created match.py, which can be seen below. It contained three functions, which I used to find male users, male users my age(I'll be 22 very soon), and finally that list was shortened to students. The last piece of functionality of match.py was to generate a li st of the preferred movies for each user based upon their ranking. High rankings that were output were films assigned either a rating of 5 or 4, in order to increase the amount of options. Low rated films were ones who were rated a 1. When I used 1 & 2, the list became too extensive. Below are the contents of match.py. I heavily relied on recommendation.py to assist in generating this file as with the rest of the assignment for the sake of reusability. Each of the functions were ran independently and the JSON output was used for the next function when I commented out the call I just performed and uncommented the next one.

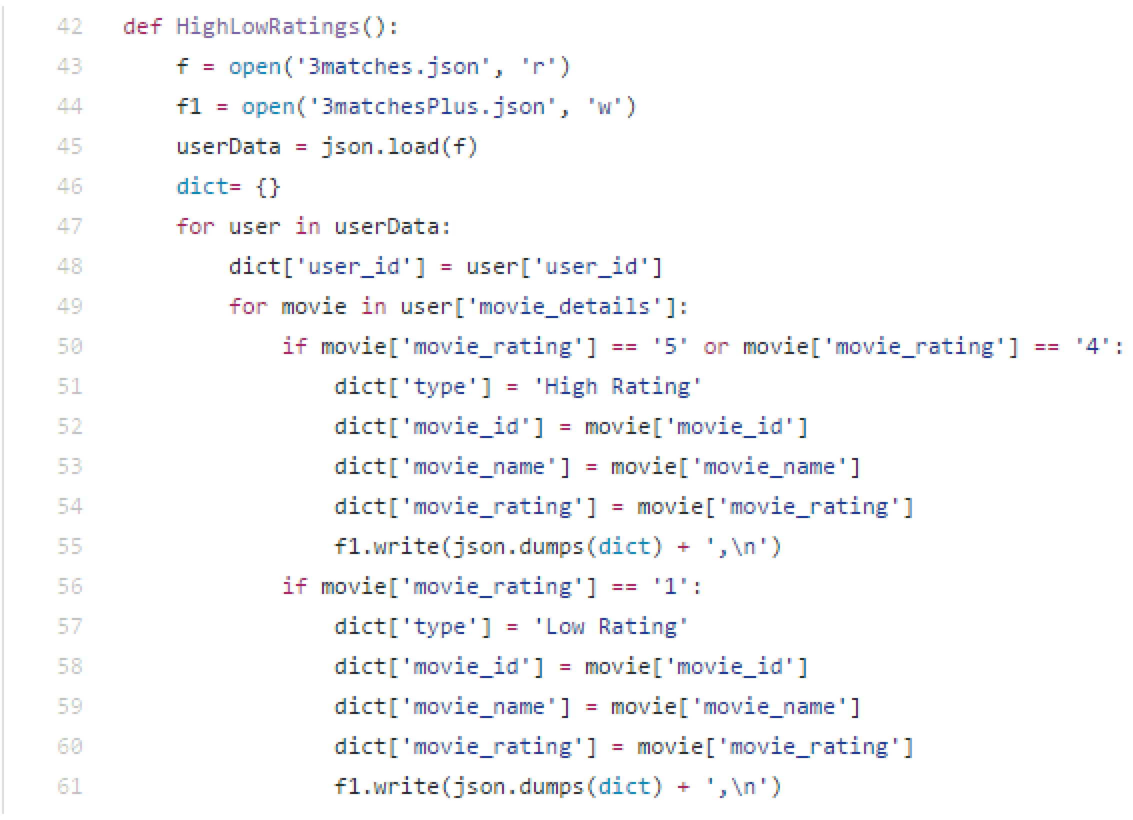


```python
import json

def Males():
    f = open('males', 'w')
    userDataFile = open("userData2.json","r")
    userData = json.load(userDataFile)
    count = 0
    countM = 0
    for user in userData:
        count += 1
        if user['user_details']['gender'] =='M':
            countM += 1
            f.write(json.dumps(user) + ',\n')


def Males22():
    f = open('males', 'r')
    userData = json.load(f)
    f1 = open('males22', 'w')
    count24 = 0
    countM = 0
    for user in userData:
        countM += 1
        if user['user_details']['age'] == '22':
            count24 += 1
            f1.write(json.dumps(user) + ',\n')
```

Bryan Carey
CS432 Assignment 7



```python
def Males22Student():
    f = open('males22', 'r')
    userData = json.load(f)
    f1 = open('males22Student', 'w')
    count22 = 0
    countStudent = 0
    for user in userData:
        count22 += 1
        if user['user_details']['occupation'] == 'student':
            countStudent += 1
            f1.write(json.dumps(user) + ',\n')
```



```python
def HighLowRatings():
    f = open('3matches.json', 'r')
    f1 = open('3matchesPlus.json', 'w')
    userData = json.load(f)
    dict= {}
    for user in userData:
        dict['user_id'] = user['user_id']
        for movie in user['movie_details']:
            if movie['movie_rating'] == '5' or movie['movie_rating'] == '4':
                dict['type'] = 'High Rating'
                dict['movie_id'] = movie['movie_id']
                dict['movie_name'] = movie['movie_name']
                dict['movie_rating'] = movie['movie_rating']
                f1.write(json.dumps(dict) + ',\n')
            if movie['movie_rating'] == '1':
                dict['type'] = 'Low Rating'
                dict['movie_id'] = movie['movie_id']
                dict['movie_name'] = movie['movie_name']
                dict['movie_rating'] = movie['movie_rating']
                f1.write(json.dumps(dict) + ',\n')
```

Samples of the outputs from each of the functions are provided below.

Male()

```
[{"user_details": {"gender": "M", "age": "24", "occupation": "technician"},
"user_id": "1", "movie_details": [{"movie_rating": "4", "movie_id": "61",
"movie_name": "Three Colors: White (1994)"}, {"movie_rating": "3",
"movie_id": "189", "movie_name": "Grand Day Out, A (1992)"}, {"movie_rating":
"4", "movie_id": "33", "movie_name": "Desperado (1995)"}, {"movie_rating":
"4", "movie_id": "160", "movie_name": "Glengarry Glen Ross (1992)"},
```

Male(22)

Bryan Carey
CS432 Assignment 7

[{"user_details": {"gender": "M", "age": "22", "occupation": "executive"}, "user_id": "54", "movie_details": [{"movie_rating": "3", "movie_name": "Diabolique (1996)", "movie_id": "106"}, {"movie_rating": "3", "movie_name": "Fan, The (1996)", "movie_id": "595"}, {"movie_rating": "5", "movie_name": "Ransom (1996)", "movie_id": "742"}, {"movie_rating": "4", "movie_name": "L. A. Confidential (1997)", "movie_id": "302"}, {"movie_rating": "5", "movie_name": "Crucible, The (1996)", "movie_id": "676"},

Male22Student()

[{"user_details": {"gender": "M", "age": "22", "occupation": "student"}, "user_id": "245", "movie_details": [{"movie_rating": "5", "movie_id": "1028", "movie_name": "Grumpier Old Men (1995)"}, {"movie_rating": "4", "movie_id": "181", "movie_name": "Return of the Jedi (1983)"}, {"movie_rating": "4", "movie_id": "222", "movie_name": "Star Trek: First Contact (1996)"}, {"movie_rating": "2", "movie_id": "94", "movie_name": "Home Alone (1990)"},

Once I had a list of all 22 year old male students, I found three who gave Men and Black a high rating and narrowed it down that way to users 459, 501, and 793. I chose 501 because of both of his high ratings on M.I.B, as well as Star Wars. 501 is the substitute me.

One curious think I did notice when running each of my functions, is that the generated JSON files were "broken" as in they were missing the opening bracket [, and required me to delete the final comma close the bracket] at the end. Once I did this I simply ran each of the functions in succession to obtain the required data.

The movies from my three matches were placed in 3matchesPlus.JSON. It is my general consensus that 501 had pretty decent taste, as it gave the Nutty Professor & the Fifth Element both scores of 4.

Q2) Which 5 users are most correlated to the substitute you? Which 5 users are least correlated (i.e., negative correlation)?

For the remainder of the assignment, I simply used the PCI text code with a few modifications, including the addition of a few required functions. I took my colleague's advice once again as I did for the remainder of the assignment. Below is correlation.py

```
import json


  import re
  from math import sqrt

  #code taken directly from the PCI text, found at https://github.com/arthur-
  e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py
  def sim_pearson(prefs, p1, p2):
      '''
      Returns the Pearson correlation coefficient for p1 and p2.
```

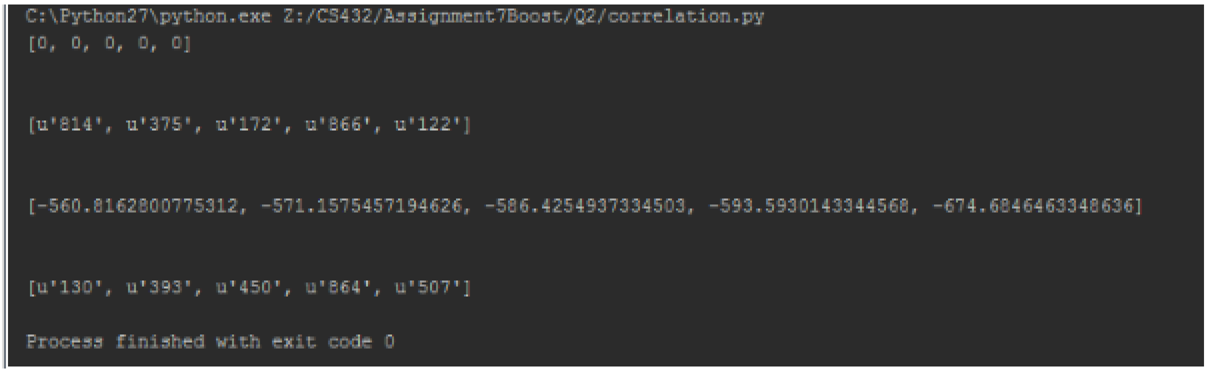
```
    '''

    # Get the list of mutually rated items
    si = {}
    for item in prefs[p1]:
        if item in prefs[p2]:
            si[item] = 1
    # If they are no ratings in common, return 0
    if len(si) == 0:
        return 0
    # Sum calculations
    n = len(si)
    # Sums of all the preferences
    sum1 = 0
    for it in si:
        sum1 = sum1 + int(prefs[p1][it])
    #repeated above code
    sum2 = 0
    for it in si:
        sum2 = sum2 + int(prefs[p2][it])
    # Sums of the squares
    sum1Sq = 0
    for it in si:
        sum1Sq = sum1Sq + pow(int(prefs[p1][it]), 2)
    sum2Sq = 0
    for it in si:
        sum2Sq = sum2Sq + pow(int(prefs[p2][it]), 2)
    # Sum of the products
    pSum = 0
    for it in si:
        pSum = pSum + (int(prefs[p2][it]) * int(prefs[p1][it]))
    # Calculate r (Pearson score)
    num = pSum - ((sum1 * sum2) / 2)
    den = sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
    if den == 0:
        return 0
    r = num / den
    return r


def findCorrelation():
    userDataFile = open("userData2.json","r")
```

Bryan Carey
CS432 Assignment 7

```
        userData = json.load(userDataFile)
        correlation_dic = {}
        for user in userData:
            user_id = user['user_id']
            movie_details_dic = {}
            for movie in user['movie_details']:
                movie_id = movie['movie_id']
                movie_rating = movie['movie_rating']
                movie_details_dic[movie_id] = movie_rating

            correlation_dic[user_id] = movie_details_dic
        return correlation_dic

    correlation_dic = findCorrelation()
    user_min = []
    user_max = []
    max = [-10, -10, -10, -10, -10]
    min = [10, 10, 10, 10, 10]

    #compare substitute me, user 459 to the other users in the data set
    for user in correlation_dic:
        if(user != '501'):
            result = sim_pearson(correlation_dic, '501', user)
            if(max[0] < result):
                max[0] = result
                max.sort()
                user_max.append(user)
            if(min[0] > result):
                min[0] = result
                min.sort()
                min.reverse()
                user_min.append(user)

    print max
    print '\n'
    print user_max[-5:]
    print '\n'
    print min
    print '\n'
    print user_min[-5:]
```

Bryan Carey
CS432 Assignment 7

The most correlated users were 814, 375, 172, 866, and 122. The least correlated were 130, 393,450, 84, and 507.The output was as follows:



```
C:\Python27\python.exe Z:/CS432/Assignment7Boost/Q2/correlation.py
[0, 0, 0, 0, 0]


[u'814', u'375', u'172', u'866', u'122']


[-560.8162800775312, -571.1575457194626, -586.4254937334503, -593.5930143344568, -674.6846463348636]


[u'130', u'393', u'450', u'864', u'507']

Process finished with exit code 0
```

Q3) Compute ratings for all the films that the substitute you have not seen. Provide a list of the top 5 recommendations for films that the substitute you should see. Provide a list of the bottom 5 recommendations (i.e., films the substitute you is almost certain to hate).

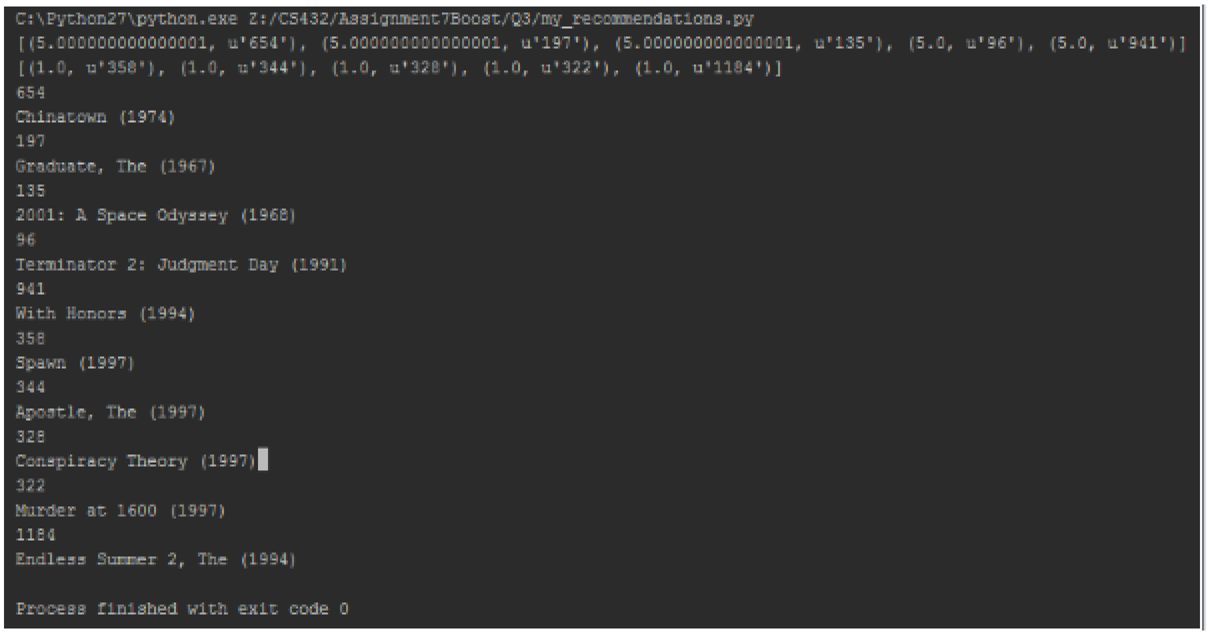
In my_recommendations.py I only changed the lower half of the code. Here is the new portion.

Bryan Carey
CS432 Assignment 7

```python
#compare substitute me, user 459 to the other users in the data set
for user in correlation_dic:
    if (user != '501'):
        result = sim_pearson(correlation_dic, '501', user)
        if (max[0] < result):
            max[0] = result
            max.sort()
            user_max.append(user)
        if (min[0] > result):
            min[0] = result
            min.sort()
            min.reverse()
            user_min.append(user)

result = getRecommendations(correlation_dic, '501', similarity=sim_pearson)

Top_Movies = result[0:5]
Least_Movies = result[-5:]
print Top_Movies
print Least_Movies

for movie in Top_Movies:
    getMovieName(movie[1])

for movie in Least_Movies:
    getMovieName(movie[1])
```

The output

Bryan Carey
CS432 Assignment 7

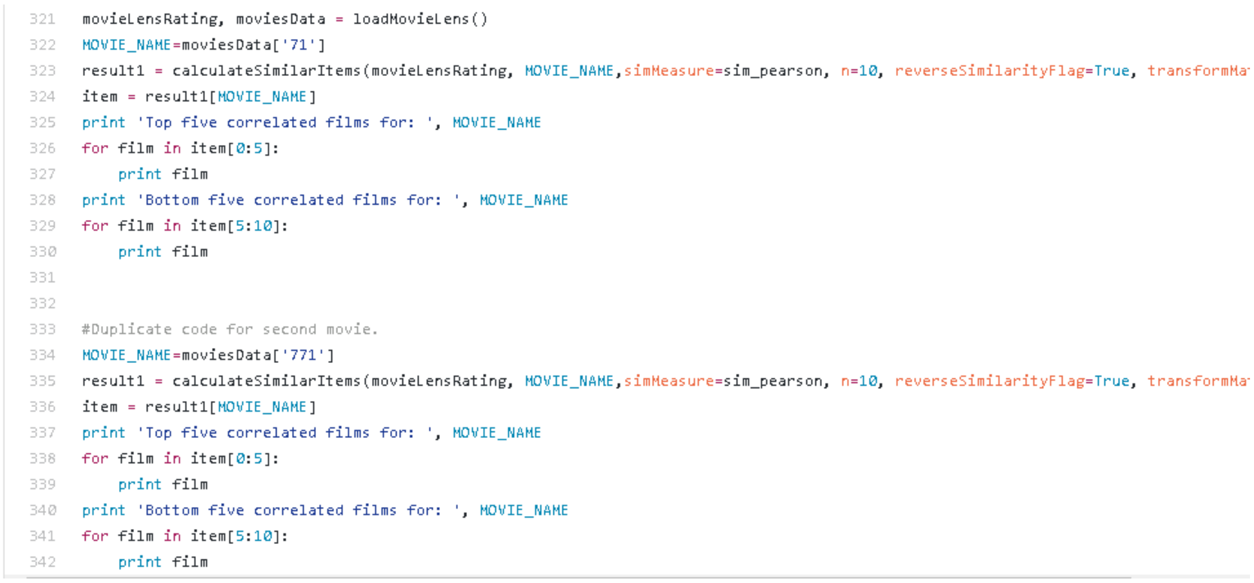


```
C:\Python27\python.exe Z:/CS432/Assignment7Boost/Q3/my_recommendations.py
[(5.000000000000001, u'654'), (5.000000000000001, u'197'), (5.000000000000001, u'135'), (5.0, u'96'), (5.0, u'941')]
[(1.0, u'358'), (1.0, u'344'), (1.0, u'328'), (1.0, u'322'), (1.0, u'1184')]
654
Chinatown (1974)
197
Graduate, The (1967)
135
2001: A Space Odyssey (1968)
96
Terminator 2: Judgment Day (1991)
941
With Honors (1994)
358
Spawn (1997)
344
Apostle, The (1997)
328
Conspiracy Theory (1997)
322
Murder at 1600 (1997)
1184
Endless Summer 2, The (1994)

Process finished with exit code 0
```

The first 5 are top recommendations, the bottom 5 are the bottom recommendations. I was a bit shocked to see Spawn on the bottom list, as I like the movie myself and thought the substitute me would too.

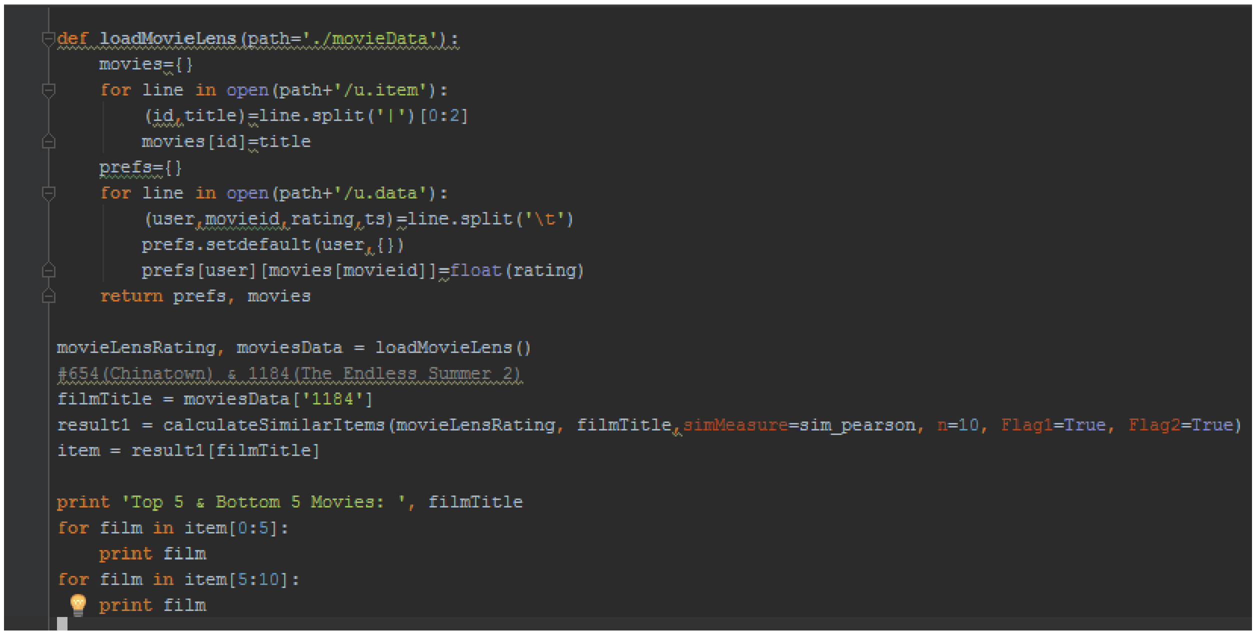
Q4) Choose your (the real you, not the substitute you) favorite and least favorite film from the data. For each film, generate a list of the top 5 most correlated and bottom 5 least correlated films. Based on your knowledge of the resulting films, do you agree with the results? In other words, do you personally like / dislike the resulting films?

My last .py file, 5TopBottom.py I once again called on the PCI text, however this time with a few different modifications. The idea of using Boolean "flags" for this section was suggested to me by a fellow classmate and while searching google to see if anyone else had done anything similar with the data I came across another GitHub user, Majetsiri. This is an exert from the code that I used.



```
movieLensRating, moviesData = loadMovieLens()
MOVIE_NAME=moviesData['71']
result1 = calculateSimilarItems(movieLensRating, MOVIE_NAME,simMeasure=sim_pearson, n=10, reverseSimilarityFlag=True, transformMa
item = result1[MOVIE_NAME]
print 'Top five correlated films for: ', MOVIE_NAME
for film in item[0:5]:
    print film
print 'Bottom five correlated films for: ', MOVIE_NAME
for film in item[5:10]:
    print film


#Duplicate code for second movie.
MOVIE_NAME=moviesData['771']
result1 = calculateSimilarItems(movieLensRating, MOVIE_NAME,simMeasure=sim_pearson, n=10, reverseSimilarityFlag=True, transformMa
item = result1[MOVIE_NAME]
print 'Top five correlated films for: ', MOVIE_NAME
for film in item[0:5]:
    print film
print 'Bottom five correlated films for: ', MOVIE_NAME
for film in item[5:10]:
    print film
```

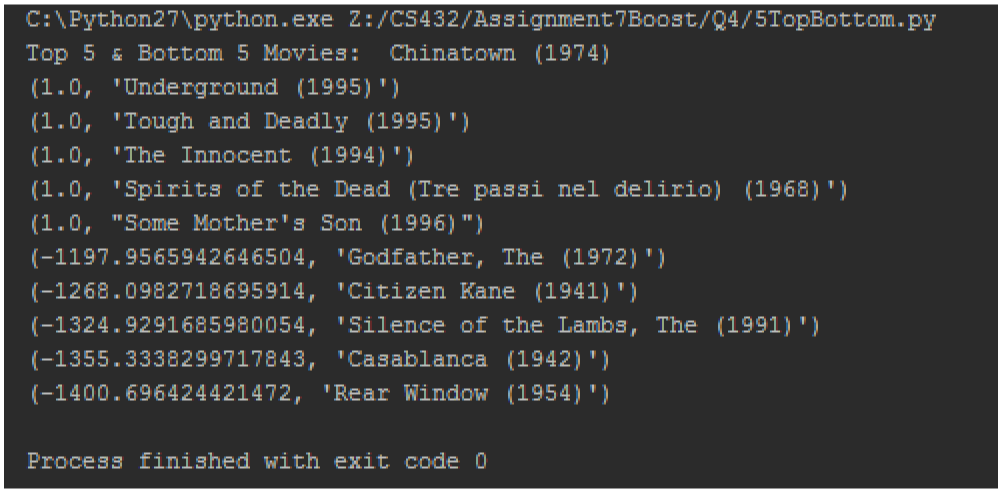
I replicated their use of flags for this portion of the assignment. I entered these flags as an argument in a prior line in the PCI text, then I was able to utilize them as arguments within a modified portion of the code from the sim_pearson function. When ran, the code generates 10 movie titles that are correlated to the movie I entered as a parameter, the first being movie 654 & the second being 1164. Those two movies were my top and bottom suggestions for the substitute me.



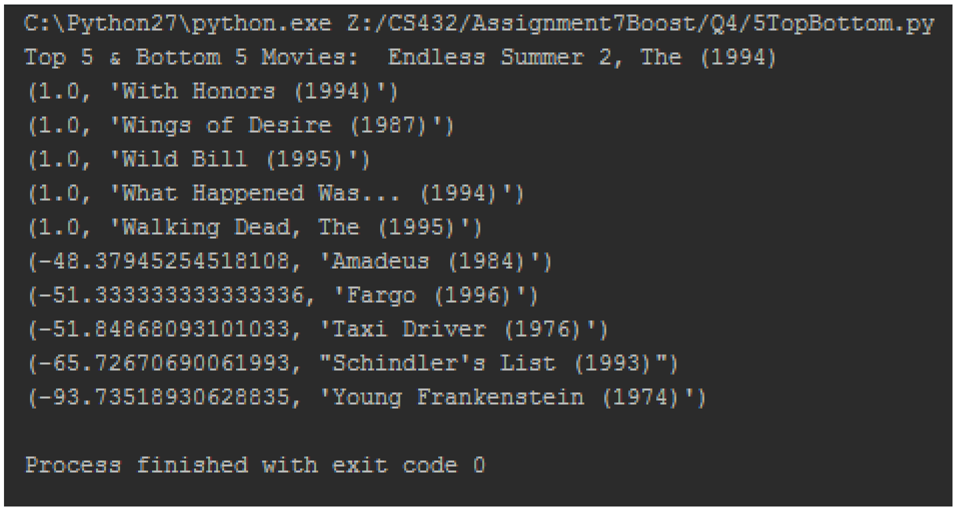
```
def loadMovieLens(path='./movieData'):
    movies={}
    for line in open(path+'/u.item'):
        (id,title)=line.split('|')[0:2]
        movies[id]=title
    prefs={}
    for line in open(path+'/u.data'):
        (user,movieid,rating,ts)=line.split('\t')
        prefs.setdefault(user,{})
        prefs[user][movies[movieid]]=float(rating)
    return prefs, movies

movieLensRating, moviesData = loadMovieLens()
#654(Chinatown) & 1184(The Endless Summer 2)
filmTitle = moviesData['1184']
result1 = calculateSimilarItems(movieLensRating, filmTitle,simMeasure=sim_pearson, n=10, Flag1=True, Flag2=True)
item = result1[filmTitle]

print 'Top 5 & Bottom 5 Movies: ', filmTitle
for film in item[0:5]:
    print film
for film in item[5:10]:
    print film
```

The output is as follows-

Bryan Carey
CS432 Assignment 7



```
C:\Python27\python.exe Z:/CS432/Assignment7Boost/Q4/5TopBottom.py
Top 5 & Bottom 5 Movies:  Chinatown (1974)
(1.0, 'Underground (1995)')
(1.0, 'Tough and Deadly (1995)')
(1.0, 'The Innocent (1994)')
(1.0, 'Spirits of the Dead (Tre passi nel delirio) (1968)')
(1.0, "Some Mother's Son (1996)")
(-1197.9565942646504, 'Godfather, The (1972)')
(-1268.0982718695914, 'Citizen Kane (1941)')
(-1324.9291685980054, 'Silence of the Lambs, The (1991)')
(-1355.3338299717843, 'Casablanca (1942)')
(-1400.696424421472, 'Rear Window (1954)')

Process finished with exit code 0
```



```
C:\Python27\python.exe Z:/CS432/Assignment7Boost/Q4/5TopBottom.py
Top 5 & Bottom 5 Movies:  Endless Summer 2, The (1994)
(1.0, 'With Honors (1994)')
(1.0, 'Wings of Desire (1987)')
(1.0, 'Wild Bill (1995)')
(1.0, 'What Happened Was... (1994)')
(1.0, 'Walking Dead, The (1995)')
(-48.37945254518108, 'Amadeus (1984)')
(-51.333333333333336, 'Fargo (1996)')
(-51.84868093101033, 'Taxi Driver (1976)')
(-65.72670690061993, "Schindler's List (1993)")
(-93.73518930628835, 'Young Frankenstein (1974)')

Process finished with exit code 0
```

I found the most correlated movies in relation to the top suggestion(Chinatown) & the bottom suggestion(The Endless Summer 2). I disagree with the Godfather not being recommended, as it was a phenomenal film however for the results of the correlated films from the least suggested category I was satisfied. One exception to this was the Walking Dead, which I would love to see and compare it to today's series of the same name.

References: https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py

https://github.com/majetisiri/cs532-s16/network

https://docs.python.org/2/library/stdtypes.html

http://stackoverflow.com/questions/35391728/a-toy-example-of-mapreduce

Bryan Carey
CS432 Assignment 7

http://stackoverflow.com/questions/14899506/displaying-better-error-message-than-no-json-object-could-be-decoded

http://stackoverflow.com/questions/3949226/calculating-pearson-correlation-and-significance-in-python

https://grouplens.org/datasets/movielens/100k/

http://stackoverflow.com/questions/6483611/python-typeerror-required-argument-source-pos-1-not-found

http://jsonlint.com/