

INTRODUCTION TO WEB SCIENCES: Assignment 8

Babitha Bokka

15 November 2014

Contents

1	Question 1:	4
1.1	Approach	4
1.2	Source Code	5
1.2.1	averageMovies.py	5
1.3	Input Files	6
1.3.1	u.data	6
1.3.2	u.item	6
1.4	Output Files	7
1.4.1	averageMovies.png	7
2	Question 2:	8
2.1	Approach	8
2.2	Source Code	9
2.2.1	mostRatingMovies.py	9
2.3	Input Files	10
2.3.1	u.data	10
2.3.2	u.item	10
2.4	Output Files	11
2.4.1	mostRatingMovies.png	11
3	Question 3:	12
3.1	Approach	12
3.2	Source Code	13
3.2.1	averageMovies.py	13
3.3	Input Files	15
3.3.1	u.data	15
3.3.2	u.item	15
3.3.3	u.user	15
3.4	Output Files	16
3.4.1	averageGenderF.png	16
4	Question 4:	17
4.1	Approach	17
4.2	Source Code	18
4.2.1	averageMovies.py	18
4.3	Input Files	20
4.3.1	u.data	20
4.3.2	u.item	20
4.3.3	u.user	20
4.4	Output Files	21
4.4.1	averageGenderM.png	21
5	Question 5:	22
5.1	Approach	22
5.2	Approach	22
5.3	Source Code	23

5.3.1	recommendations.py	23
5.3.2	ratingsTopGun.py	24
5.4	Input Files	25
5.4.1	u.data	25
5.4.2	u.item	25
5.5	Output Files	26
6	Question 6:	27
6.1	Approach	27
6.2	Source Code	28
6.2.1	ratersMostRatedMovies.py	28
6.3	Input Files	29
6.3.1	u.data	29
6.4	Output Files	29
6.4.1	ratersMostRatedMovies.png	29
7	Question 7:	30
7.1	Approach	30
7.2	Description of	30
7.3	Source Code	31
7.4	Output Files	31
8	Question 8:	32
8.1	Approach	32
8.2	Description of	32
8.3	Source Code	33
8.4	Output Files	33
9	Question 9:	34
9.1	Approach	34
9.2	Source Code	35
9.2.1	averageOverUnderGender.py	35
9.3	Input Files	37
9.3.1	u.data	37
9.3.2	u.item	37
9.3.3	u.user	37
9.4	Output Files	38
9.4.1	averageOverM.png	38
9.4.2	averageUnderM.png	39
10	Question 10:	40
10.1	Approach	40
10.2	Source Code	41
10.2.1	averageOverUnderGender.py	41
10.3	Input Files	43
10.3.1	u.data	43
10.3.2	u.item	43
10.3.3	u.user	43

10.4	Output Files	44
10.4.1	averageOverF.png	44
10.4.2	averageUnderF.png	45

1 Question 1:

What 5 movies have the highest average ratings? Show the movies and their ratings sorted by their average ratings.

1.1 Approach

1. To solve this question the straight forward approach is to read u.data file get movie id, corresponding movie ratings and read u.item file to get movie id and corresponding movie name.
2. Extract the movie id and ratings. For each movie id build a dictionary that has all the ratings.
3. Now, for each each movie id calculate the average a using `sum()` and `len()` functions.
4. For each movie id get the movie name and append it to the dictionary which has average.
5. Now, sort the dictionary and print the top 5 movies with highest average ratings
6. Program `averageMovies.py` produces the top 5 movies with highest average ratings.
7. Figure 1 is the output showing the results of the program are limited to top 5 it can get more average ratings by changing a conditional statement.

1.2 Source Code

1.2.1 averageMovies.py

```
1 #!/usr/bin/env python
2
3 import sys
4
5 def main():
6     # Create a dictionary
7     movie_id_ratings = {}
8     movie_id_avg = {}
9     count = 1
10    # Read the input files.
11    readData = open('/home/bbokka/cs594/A8/u.data', 'r')
12    readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
13    # reading the u.data file for item and ratings.
14    for line in readData:
15        split_input_line = line.strip().split('\t')
16        movie_id = split_input_line[1]
17        movie_rating = float(split_input_line[2])
18        try:
19            movie_id_ratings[movie_id].append(movie_rating)
20        except KeyError:
21            movie_id_ratings[movie_id] = list()
22            movie_id_ratings[movie_id].append(movie_rating)
23    readData.close()
24    # Calculating the average.
25    for key in movie_id_ratings:
26        avg = sum(movie_id_ratings[key])/len(movie_id_ratings[key])
27        movie_id_avg[key] = [float(avg)]
28
29    # Reading the u.item file for movie name.
30    for each_line in readItem:
31        split_each_line = each_line.strip().split('|')
32        movie_item_id = split_each_line[0]
33        movie_name_split = split_each_line[1].split('(1')
34        movie_name = movie_name_split[0]
35
36        movie_id_avg[movie_item_id].append(movie_name)
37    readItem.close()
38    print "*" * 60
39    print "Top 5 movies having the highest average ratings"
40    print "*" * 60
41    print "Movie Name\t\t\t\t\t","Avg Rating"
42    print "-" * 60
43    # sorting the movies from highest to lowest based on the average value.
44    for key, value in sorted(movie_id_avg.items(), key=lambda e: e[1], reverse=True):
45        if (count <= 5):
46            print '{:<50}{:<0.1f} '.format(value[1], value[0])
47            count += 1
48
49 if __name__ == "__main__":
50     try:
51         main()
52     except KeyboardInterrupt:
53         sys.exit(1)
```

1.3 Input Files

1.3.1 u.data

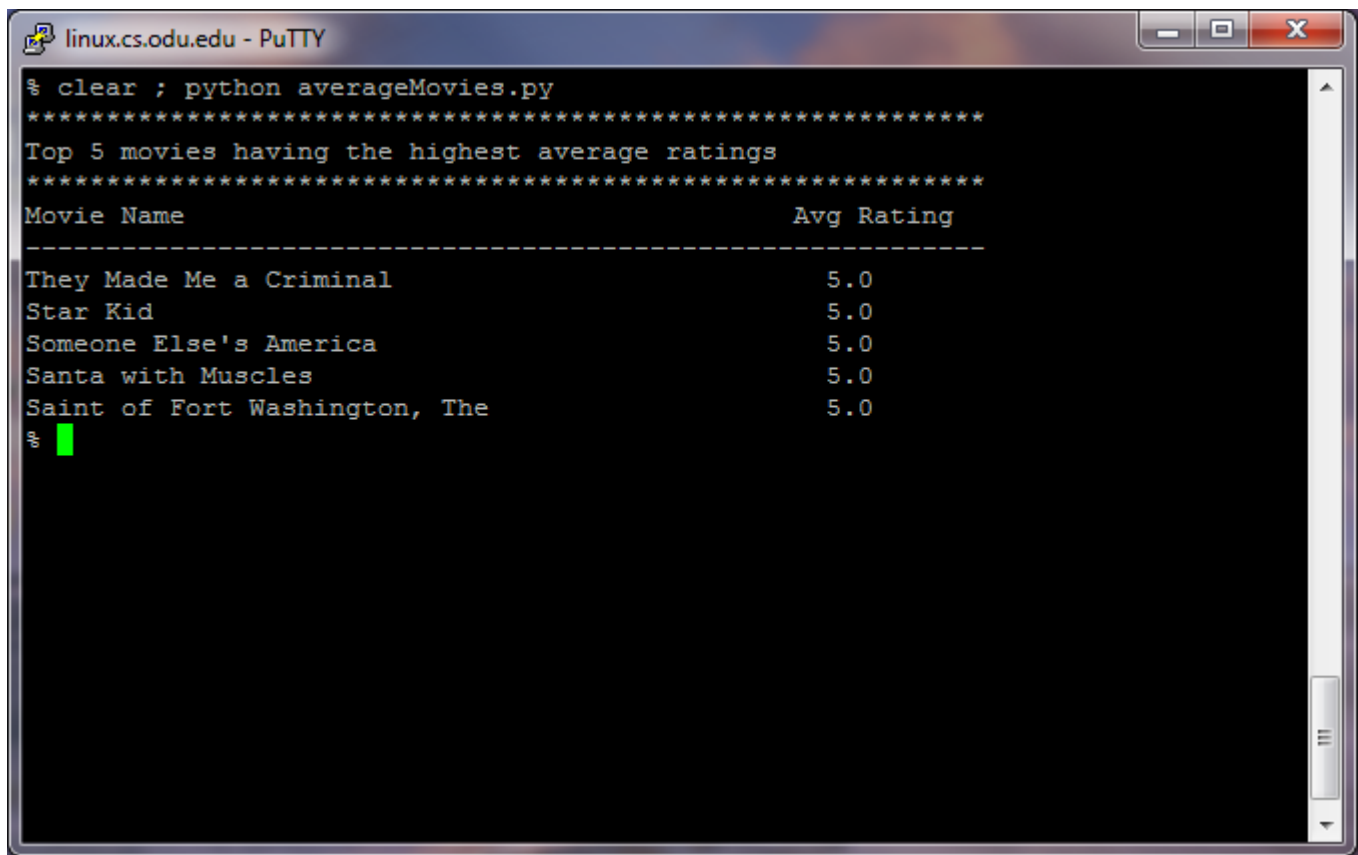
	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

1.3.2 u.item

[illegible]

1.4 Output Files

1.4.1 averageMovies.png



```
linux.cs.odu.edu - PuTTY
% clear ; python averageMovies.py
*****
Top 5 movies having the highest average ratings
*****
Movie Name                               Avg Rating
-----
They Made Me a Criminal                  5.0
Star Kid                                5.0
Someone Else's America                   5.0
Santa with Muscles                       5.0
Saint of Fort Washington, The            5.0
% 
```

Figure 1: Top 5 movies with highest average ratings.

2 Question 2:

What 5 movies received the most ratings? Show the movies and the number of ratings sorted by number of ratings.

2.1 Approach

1. The approach is same as question 1.
2. The only difference is we count the number of ratings for each movie and display the corresponding movie name and number of ratings for each movie.
3. The output displays only top 5 movies and their number of ratings.
4. Figure 2 is the output for the program.

2.2 Source Code

2.2.1 mostRatingMovies.py

```
1#!/usr/bin/env python
2
3import sys
4
5def main():
6    # Create a dictionary
7    movie_id_ratings = {}
8    movie_id_avg = {}
9    count = 1
10   # Read the input files.
11   readData = open('/home/bbokka/cs594/A8/u.data', 'r')
12   readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
13   # reading the u.data file for item and ratings.
14   for line in readData:
15       split_input_line = line.strip().split('\t')
16       movie_id = split_input_line[1]
17       movie_rating = int(split_input_line[2])
18       try:
19           movie_id_ratings[movie_id].append(movie_rating)
20       except KeyError:
21           movie_id_ratings[movie_id] = list()
22           movie_id_ratings[movie_id].append(movie_rating)
23   readData.close()
24   # Calculating the average.
25   for key in movie_id_ratings:
26       length = len(movie_id_ratings[key])
27       movie_id_avg[key] = [int(length)]
28
29   # Reading the u.item file for movie name.
30   for each_line in readItem:
31       split_each_line = each_line.strip().split('|')
32       movie_item_id = split_each_line[0]
33       movie_name_split = split_each_line[1].split('(1')
34       movie_name = movie_name_split[0]
35
36       movie_id_avg[movie_item_id].append(movie_name)
37   readItem.close()
38   print '*' * 60
39   print "Top 5 movies received the most ratings"
40   print "*" * 60
41   print "Movie Name\t\t\t\t\t", "Number of ratings"
42   print "-" * 60
43   # sorting the movies from highest to lowest based on the average value.
44   for key, value in sorted(movie_id_avg.items(), key=lambda e: e[1], reverse=True):
45       if (count <= 5):
46           print '{:<50}{} ' .format(value[1], value[0])
47           count += 1
48
49 if __name__ == "__main__":
50     try:
51         main()
52     except KeyboardInterrupt:
53         sys.exit(1)
```

2.3 Input Files

2.3.1 u.data

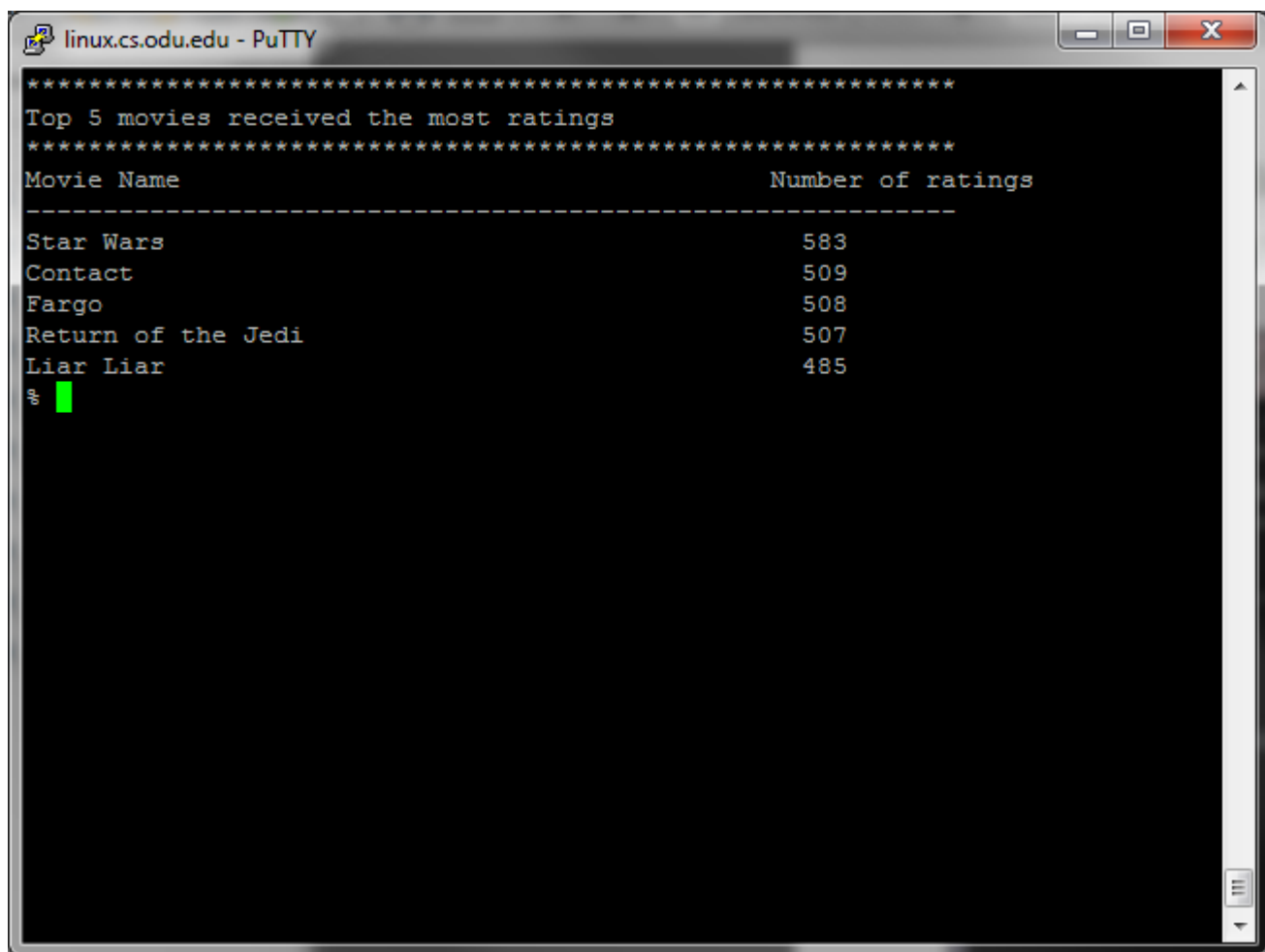
	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

2.3.2 u.item

movie id	movie title	release date	video release date	IMDb URL	unknown
	Action Adventure Animation Children's Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western				
1	Toy Story (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)	
		000011110000000000000000			
2	GoldenEye (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?GoldenEye%20(1995)	
		0011100000000000000000001000			
3	Four Rooms (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)	
		0000000000000000000000001000			
4	Get Shorty (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)	
		0010000100010000000000000000			
5	Copycat (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Copycat%20(1995)	
		0000000010010000000000001000			
6	Shanghai Triad (Yao a yao dao waipo qiao) (1995)	01-Jan-1995		http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)	
		000000000000000000100000000000			
7	Twelve Monkeys (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995)	
		00000000000010000000000010000			

2.4 Output Files

2.4.1 mostRatingMovies.png



```
linux.cs.odu.edu - PuTTY
*****
Top 5 movies received the most ratings
*****
Movie Name                                Number of ratings
-----
Star Wars                                583
Contact                                  509
Fargo                                    508
Return of the Jedi                       507
Liar Liar                                485
% 
```

Figure 2: Top 5 movies received the most ratings.

3 Question 3:

What 5 movies were rated the highest on average by women? Show the movies and their ratings sorted by ratings.

3.1 Approach

1. To solve this question the straight forward approach is to read u.user file get the user id, gender. Read u.data file get movie id, corresponding movie ratings. Read u.item file to get movie id and corresponding movie name.
2. From u.user file extract the user id and gender store them in a dictionary.
3. Now, Extract the movie id and ratings. For each user id build a dictionary that has all the ratings for the respective gender.
4. Now, for each each movie id calculate the average a using `sum()` and `len()` functions.
5. For each movie id get the movie name and append it to the dictionary which has average.
6. Now, sort the dictionary and print the top 5 movies rated the highest on average by women.
7. Program `averageGender.py` is designed in such a that it can get the highest averages by men and women just by changing the arguments to F or M while executing the program.
8. Figure 3 is the output showing top 5 movies rated the highest on average by women the results of the program are limited to top 5 it can be modified to get more average ratings.

3.2 Source Code

3.2.1 averageMovies.py

```
1 #!/usr/bin/env python
2
3
4 # program to get the highest average ratings by men and women
5 import sys
6 import operator
7 def main():
8     # take the arguments from the command line
9     numOfArgs=len(sys.argv)
10    if numOfArgs<2 or numOfArgs>2:
11
12        print 'Usage: averageGender.py <F or M> '
13        print 'e.g. : averageGender.py F'
14        sys.exit(1)
15    gender = sys.argv[1]
16    # Create a dictionary
17    movie_userid_avg = {}
18    movie_id_ratings = {}
19    movie_userid_avg = {}
20    users = {}
21    count = 1
22
23    readData = open('/home/bbokka/cs594/A8/u.data', 'r')
24    readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
25    readUser = open('/home/bbokka/cs594/A8/u.user', 'r')
26
27    # Reading u.user file to get the male or female
28    for line in readUser:
29        split_input_line = line.strip().split('|')
30        user_id_from_user= split_input_line[0]
31        user_gender = split_input_line[2]
32
33        if(gender.upper() == user_gender):
34            users[user_id_from_user] = [user_gender]
35
36    readUser.close()
37
38    # reading the u.data file for item and ratings.
39    for line in readData:
40        split_input_line = line.strip().split('\t')
41        user_id_from_data = split_input_line[0]
42        movie_id_from_data= split_input_line[1]
43        movie_rating = float(split_input_line[2])
44
45        if user_id_from_data in users:
46            try:
47                movie_id_ratings[movie_id_from_data].append(movie_rating)
48            except KeyError:
49                movie_id_ratings[movie_id_from_data] = list()
50                movie_id_ratings[movie_id_from_data].append(movie_rating)
51    readData.close()
52
53    # Calculating the average.
```

```

54     for key in movie_id_ratings:
55         avg = sum(movie_id_ratings[key])/len(movie_id_ratings[key])
56         movie_userid_avg[key] = [float(avg)]
57
58     # Reading the u.item file for movie name.
59     for each_line in readItem:
60         split_each_line = each_line.strip().split('|')
61         movie_item_id = split_each_line[0]
62         movie_name_split = split_each_line[1].split('(1')
63         movie_name = movie_name_split[0]
64
65         try:
66             movie_userid_avg[movie_item_id].append(movie_name)
67         except KeyError:
68             pass
69     readItem.close()
70     print "*" * 60
71     print "Top 5 movies rated the highest on average by women"
72     print "*" * 60
73     print "Movie Name\t\t\t\t\t", "Avg Rating"
74     print "-" * 60
75     # sorting the movies from highest to lowest based on the average value.
76     for key, value in sorted(movie_userid_avg.items(), key=lambda e: e[1], reverse=
77 True):
78         if (count <=5 ):
79             print '{:<50}{:<0.1f} '.format(value[1], value[0])
80             count += 1
81
82 if __name__ == "__main__":
83     try:
84         main()
85     except KeyboardInterrupt:
86         sys.exit(1)

```

3.3 Input Files

3.3.1 u.data

	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

3.3.2 u.item

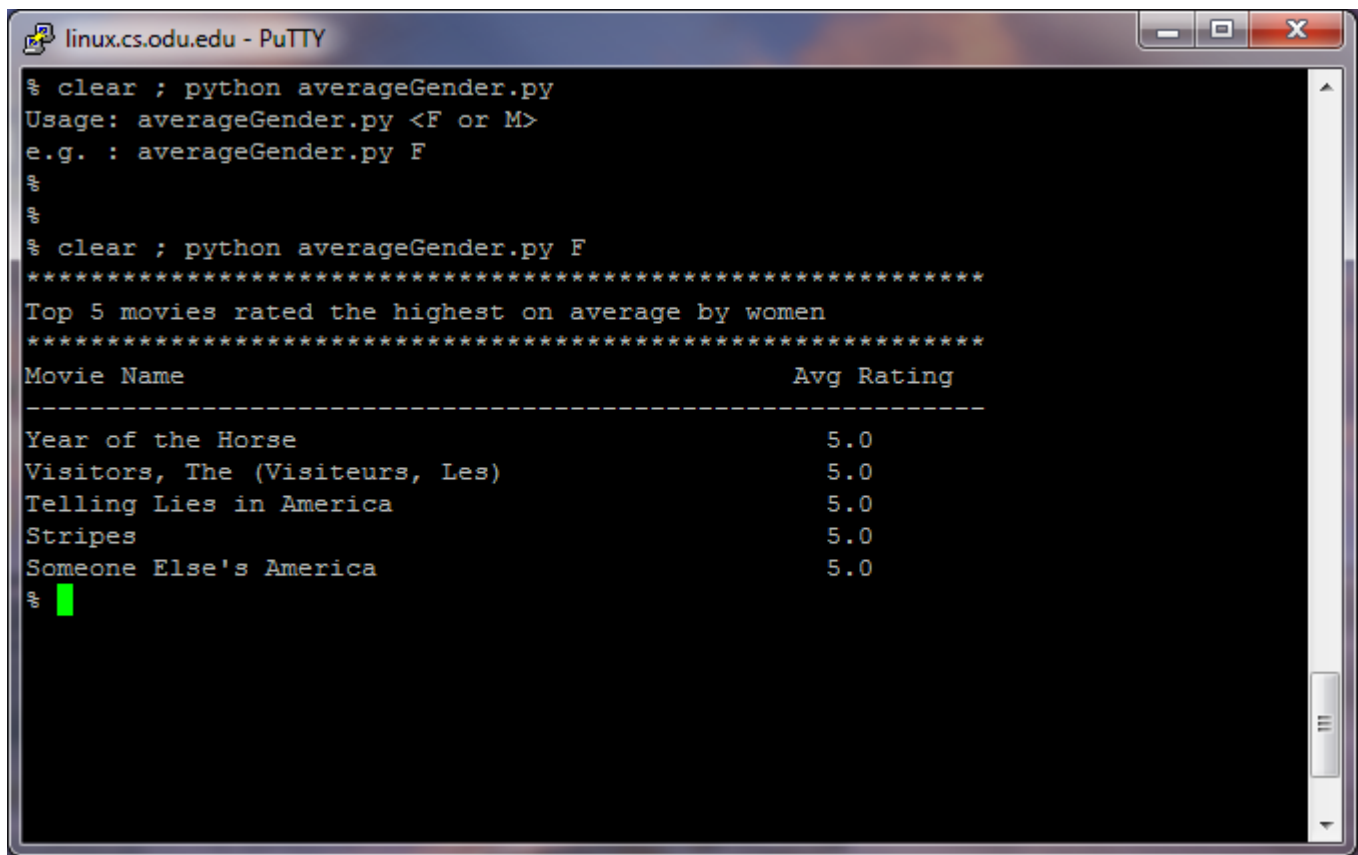
[illegible]

3.3.3 u.user

	userid	age	gender	occupation	zip_code
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344

3.4 Output Files

3.4.1 averageGenderF.png



The screenshot shows a PuTTY terminal window titled "linux.cs.odu.edu - PuTTY". The user has entered the command `% clear ; python averageGender.py`. The script displays usage instructions: `Usage: averageGender.py <F or M>` and `e.g. : averageGender.py F`. The user then enters `% clear ; python averageGender.py F`. The script outputs a list of the top 5 movies rated highest on average by women, each with a 5.0 rating. The output is formatted as a table with columns "Movie Name" and "Avg Rating".

```
% clear ; python averageGender.py
Usage: averageGender.py <F or M>
e.g. : averageGender.py F
%
%
% clear ; python averageGender.py F
*****
Top 5 movies rated the highest on average by women
*****
Movie Name                               Avg Rating
-----
Year of the Horse                         5.0
Visitors, The (Visiteurs, Les)           5.0
Telling Lies in America                  5.0
Stripes                                  5.0
Someone Else's America                   5.0
% █
```

Figure 3: Top 5 movies rated the highest on average by women.

4 Question 4:

What 5 movies were rated the highest on average by men? Show the movies and their ratings sorted by ratings.

4.1 Approach

1. The approach is same as question 3.
2. The only difference is changing the command line argument Which can be seen in the output produced.
3. The output displays only top 5 movies rated the highest on average by men.
4. Figure 4 is the output for the program.

4.2 Source Code

4.2.1 averageMovies.py

```
1 #!/usr/bin/env python
2
3
4 # program to get the highest average ratings by men and women
5 import sys
6 import operator
7 def main():
8     # take the arguments from the command line
9     numOfArgs=len(sys.argv)
10    if numOfArgs<2 or numOfArgs>2:
11
12        print 'Usage: averageGender.py <F or M> '
13        print 'e.g. : averageGender.py F'
14        sys.exit(1)
15    gender = sys.argv[1]
16    # Create a dictionary
17    movie_userid_avg = {}
18    movie_id_ratings = {}
19    movie_userid_avg = {}
20    users = {}
21    count = 1
22
23    readData = open('/home/bbokka/cs594/A8/u.data', 'r')
24    readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
25    readUser = open('/home/bbokka/cs594/A8/u.user', 'r')
26
27    # Reading u.user file to get the male or female
28    for line in readUser:
29        split_input_line = line.strip().split('|')
30        user_id_from_user= split_input_line[0]
31        user_gender = split_input_line[2]
32
33        if(gender.upper() == user_gender):
34            users[user_id_from_user] = [user_gender]
35
36    readUser.close()
37
38    # reading the u.data file for item and ratings.
39    for line in readData:
40        split_input_line = line.strip().split('\t')
41        user_id_from_data = split_input_line[0]
42        movie_id_from_data= split_input_line[1]
43        movie_rating = float(split_input_line[2])
44
45        if user_id_from_data in users:
46            try:
47                movie_id_ratings[movie_id_from_data].append(movie_rating)
48            except KeyError:
49                movie_id_ratings[movie_id_from_data] = list()
50                movie_id_ratings[movie_id_from_data].append(movie_rating)
51    readData.close()
52
53    # Calculating the average.
```

```

54     for key in movie_id_ratings:
55         avg = sum(movie_id_ratings[key])/len(movie_id_ratings[key])
56         movie_userid_avg[key] = [float(avg)]
57
58     # Reading the u.item file for movie name.
59     for each_line in readItem:
60         split_each_line = each_line.strip().split('|')
61         movie_item_id = split_each_line[0]
62         movie_name_split = split_each_line[1].split('(1')
63         movie_name = movie_name_split[0]
64
65         try:
66             movie_userid_avg[movie_item_id].append(movie_name)
67         except KeyError:
68             pass
69     readItem.close()
70     print "*" * 60
71     print "Top 5 movies rated the highest on average by women"
72     print "*" * 60
73     print "Movie Name\t\t\t\t\t", "Avg Rating"
74     print "-" * 60
75     # sorting the movies from highest to lowest based on the average value.
76     for key, value in sorted(movie_userid_avg.items(), key=lambda e: e[1], reverse=
77 True):
78         if (count <=5 ):
79             print '{:<50}{:<0.1f} '.format(value[1], value[0])
80             count += 1
81
82 if __name__ == "__main__":
83     try:
84         main()
85     except KeyboardInterrupt:
86         sys.exit(1)

```

4.3 Input Files

4.3.1 u.data

	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

4.3.2 u.item

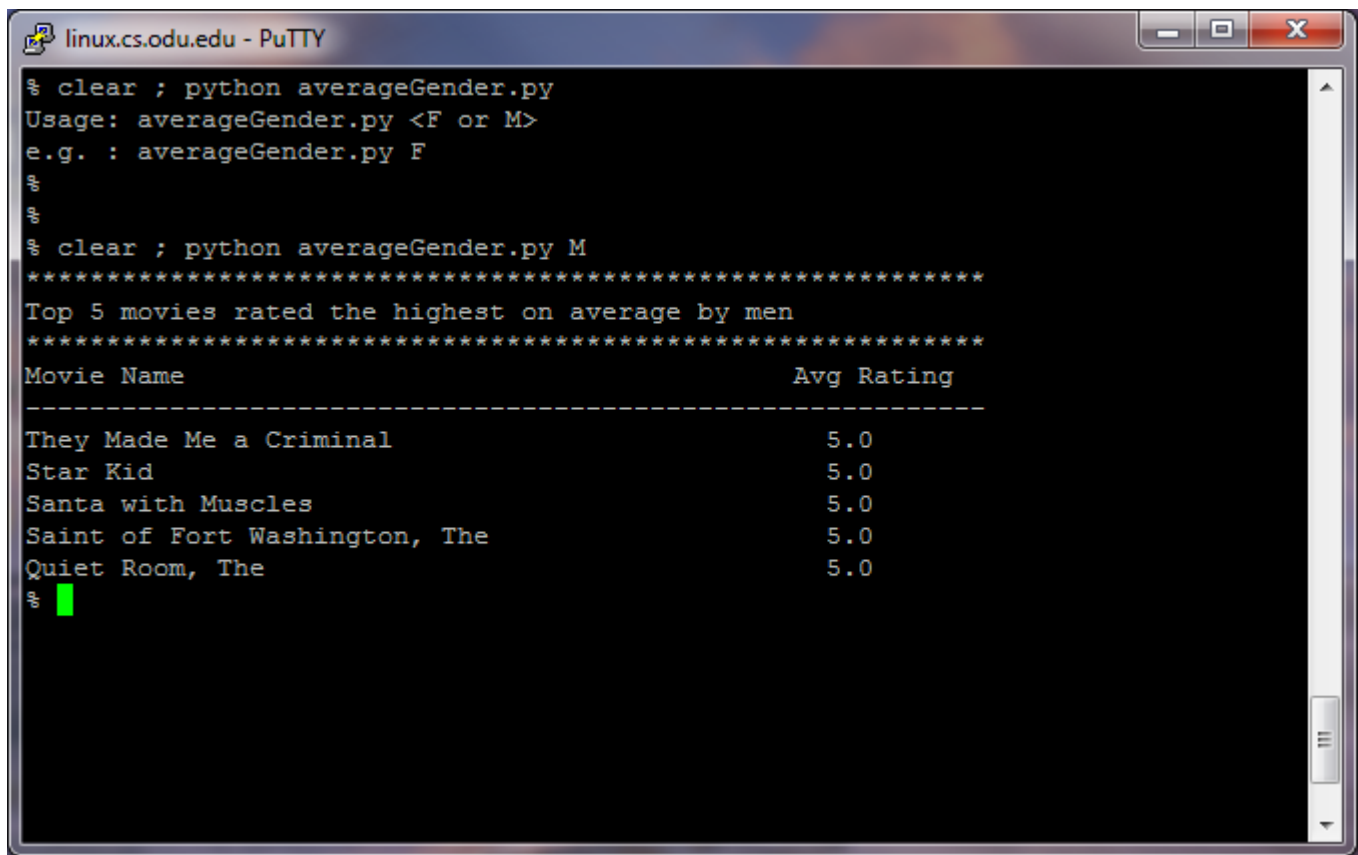
[illegible]

4.3.3 u.user

	userid	age	gender	occupation	zip_code
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344

4.4 Output Files

4.4.1 averageGenderM.png



```
linux.cs.odu.edu - PuTTY
% clear ; python averageGender.py
Usage: averageGender.py <F or M>
e.g. : averageGender.py F
%
%
% clear ; python averageGender.py M
*****
Top 5 movies rated the highest on average by men
*****
Movie Name                               Avg Rating
-----
They Made Me a Criminal                   5.0
Star Kid                                 5.0
Santa with Muscles                        5.0
Saint of Fort Washington, The             5.0
Quiet Room, The                          5.0
% █
```

Figure 4: Top 5 movies rated the highest on average by men.

5 Question 5:

What movie received ratings most like Top Gun? Which movie received ratings that were least like Top Gun (negative correlation)?

5.1 Approach

5.2 Approach

- 1.

5.3 Source Code

5.3.1 recommendations.py

```
1 def calculateSimilarItems(prefs,n=80):
2     # Create a dictionary of items showing which other items they
3     # are most similar to.
4     result={}
5     item1 = 'Top Gun (1986)'
6     # Invert the preference matrix to be item-centric
7     itemPrefs=transformPrefs(prefs)
8     c=0
9     for item in itemPrefs:
10         # Status updates for large datasets
11         # c+=1
12         # if c%100==0: print "%d / %d" % (c,len(itemPrefs))
13         # Find the most similar items to this one
14         scores=topMatches(itemPrefs,item1,n=n,similarity=sim_pearson)
15         result[item]=scores
16     return result
17
18
19 def loadMovieLens():
20     # Get movie titles
21     readData = open('/home/bbokka/cs594/A8/u.data', 'r')
22     readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
23     movies={}
24     for line in readItem:
25         (id,title)=line.split('|')[0:2]
26         movies[id]=title
27     # Load data
28     prefs={}
29     for line in readData:
30         (user,movieid,rating,ts)=line.split('\t')
31         prefs.setdefault(user,{})
32         prefs[user][movies[movieid]]=float(rating)
33
34     return prefs
```


5.3.2 ratingsTopGun.py

```
1#!/usr/bin/env python
2
3import sys
4import recommendations
5
6def main():
7    count = 'Top Gun (1986)'
8    resultsOfloadMovieLens = recommendations.loadMovieLens()
9    resultsOfcalculateSimilarItems = recommendations.calculateSimilarItems(
10    resultsOfloadMovieLens,n=80)
11    print "*" * 60
12    print "Movies received ratings most like Top Gun"
13    print "*" * 60
14    print "value\t\t\t\t\t", "Movie Name"
15    print "-" * 60
16
17    for key, value in sorted(resultsOfcalculateSimilarItems.items(), key=lambda e: e
18    [1], reverse=True):
19        variable = key
20        if (count == variable):
21            for value , movie in value:
22                print value , movie
23
24if __name__ == "__main__":
25    try:
26        main()
27    except KeyboardInterrupt:
28        sys.exit(1)
```

5.4 Input Files

5.4.1 u.data

	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

5.4.2 u.item

movie id	movie title	release date	video release date	IMDb URL	unknown
	Action Adventure Animation Children's Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western				
1	Toy Story (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)	
		000011110000000000000000			
2	GoldenEye (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?GoldenEye%20(1995)	
		0011100000000000000000001000			
3	Four Rooms (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)	
		0000000000000000000000001000			
4	Get Shorty (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)	
		0010000100010000000000000000			
5	Copycat (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Copycat%20(1995)	
		0000000010010000000000001000			
6	Shanghai Triad (Yao a yao dao waipo qiao) (1995)	01-Jan-1995		http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)	
		000000000000000010000000000000			
7	Twelve Monkeys (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995)	
		00000000000010000000000010000			

5.5 Output Files

6 Question 6:

Which 5 raters rated the most films? Show the raters' IDs and the number of films each rated.

6.1 Approach

1. The approach is to read U.data file get user id and for the corresponding user id store all the ratings .
2. Now, we count the number of ratings for each user by using the len() function.
3. The output displays only top 5 users and their number of ratings.
4. Figure 5 is the output for the program.

6.2 Source Code

6.2.1 ratersMostRatedMovies.py

```
1#!/usr/bin/env python
2
3import sys
4
5def main():
6    # Create a dictionary
7    movie_userid_ratings = {}
8    movie_userid_len = {}
9    count = 1
10   # Read the input files.
11   readData = open('/home/bbokka/cs594/A8/u.data', 'r')
12   # reading the u.data file for item and ratings.
13   for line in readData:
14       split_input_line = line.strip().split('\t')
15       user_id_from_data = split_input_line[0]
16       movie_rating = float(split_input_line[2])
17
18       try:
19           movie_userid_ratings[user_id_from_data].append(movie_rating)
20       except KeyError:
21           movie_userid_ratings[user_id_from_data] = list()
22           movie_userid_ratings[user_id_from_data].append(movie_rating)
23   readData.close()
24   # Calculating the number of movies each user rated.
25   for key in movie_userid_ratings:
26       length = len(movie_userid_ratings[key])
27       movie_userid_len[key] = int(length)
28
29
30   print '*' * 60
31   print "Top 5 raters rated movies."
32   print "*" * 60
33   print "User id\t\t", "Number of movies rated"
34   print "-" * 60
35   # sorting the movies from highest to lowest based on the number of
36   # ratings for each movie.
37   for key, value in sorted(movie_userid_len.items(), key=lambda e: e[1], reverse=
True):
38       if (count <= 5):
39           print '{:<20}{} ' .format(key, value)
40           count += 1
41
42   if __name__ == "__main__":
43       try:
44           main()
45       except KeyboardInterrupt:
46           sys.exit(1)
```

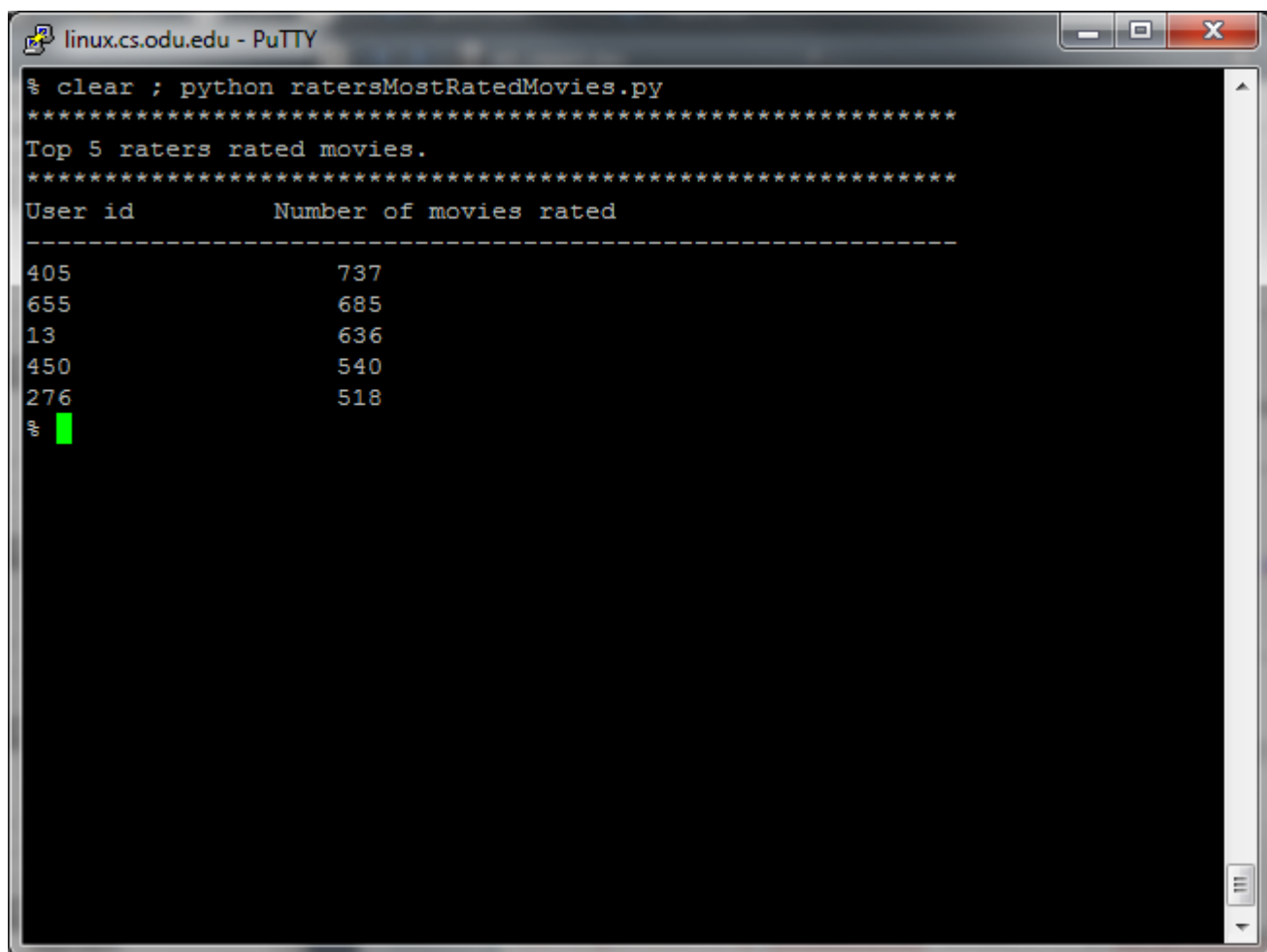
6.3 Input Files

6.3.1 u.data

```
1 userid itemid rating timestamp
2
3 196      242      3    881250949
4 186      302      3    891717742
5 22       377      1    878887116
6 244      51       2    880606923
7 166      346      1    886397596
8 298      474      4    884182806
9 115      265      2    881171488
```

6.4 Output Files

6.4.1 ratersMostRatedMovies.png



```
linux.cs.odu.edu - PuTTY
% clear ; python ratersMostRatedMovies.py
*****
Top 5 raters rated movies.
*****
User id      Number of movies rated
-----
405          737
655          685
13           636
450          540
276          518
% █
```

Figure 5: Top 5 raters rated movies.

7 Question 7:

Which 5 raters most agreed with each other? Show the raters' IDs and Pearson's r , sorted by r .

7.1 Approach

7.2 Description of

7.3 Source Code

7.4 Output Files

8 Question 8:

Which 5 raters most disagreed with each other (negative correlation)? Show the raters' IDs and Pearson's r , sorted by r .

8.1 Approach

8.2 Description of

8.3 Source Code

8.4 Output Files

9 Question 9:

What movie was rated highest on average by men over 40? By men under 40?

9.1 Approach

1. To solve this question the straight forward approach is to read u.user file get the user id, gender, age of each user. Read u.data file get movie id, corresponding movie ratings. Read u.item file to get movie id and corresponding movie name.
2. Program averageOverUnderGender.py is designed in such a way that it can get the highest averages by men and women over and under 40 just by changing the arguments to F or M and “over” or “under” while executing the program.
3. From u.user file extract the user id, gender, age based on arguments given from command line either men over 40 or men under 40 or women over 40 or women under 40 is stored into dictionary.
4. Now, Extract the movie id and ratings. For each user id build a dictionary that has all the ratings for the respective gender with over or under condition.
5. Now, for each each movie id calculate the average a using sum() and len() functions.
6. For each movie id get the movie name and append it to the dictionary which has average.
7. Now, sort the dictionary and get the movies which are rated highest on average by men or women over 40 or under 40.
8. Figure 6 is the output showing how to execute the program and shows the top 5 movies which are rated highest on average by men “over” 40.
9. Figure 7 is the output showing how to execute the program and shows the top 5 movies which are rated highest on average by men “under” 40.

9.2 Source Code

9.2.1 averageOverUnderGender.py

```
1#!/usr/bin/env python
2
3
4# program to get the highest average ratings by men under and over 40.
5#
6import sys
7import operator
8def main():
9    # Take the arguments from the command line
10    numOfArgs=len(sys.argv)
11    if numOfArgs<3 or numOfArgs>3:
12
13        print 'Usage: averageOverUnderGender.py <F or M> <under or over>'
14        print 'e.g. : averageOverUnderGender.py F over '
15        sys.exit(1)
16    gender          = sys.argv[1]
17    gender_under_over= sys.argv[2]
18
19    # Create a dictionary
20    movie_userid_avg = {}
21    movie_id_ratings = {}
22    movie_userid_avg = {}
23    users             = {}
24    count             = 1
25
26    # Read the files
27    readData = open('/home/bbokka/cs594/A8/u.data', 'r')
28    readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
29    readUser = open('/home/bbokka/cs594/A8/u.user', 'r')
30
31    # Reading u.user file to get the male or female
32    for line in readUser:
33        split_input_line = line.strip().split('|')
34        user_id_from_user= split_input_line[0]
35        user_age         = int(split_input_line[1])
36        user_gender      = split_input_line[2]
37
38        if gender_under_over == 'over' :
39            if(gender.upper() == user_gender and user_age > 40):
40                users[user_id_from_user] = [user_age]
41
42        if gender_under_over == 'under' :
43            if(gender.upper() == user_gender and user_age < 40):
44                users[user_id_from_user] = [user_age]
45
46    readUser.close()
47
48    # reading the u.data file for item and ratings.
49    for line in readData:
50        split_input_line = line.strip().split('\t')
51        user_id_from_data = split_input_line[0]
52        movie_id_from_data= split_input_line[1]
53        movie_rating      = float(split_input_line[2])
```

```

54
55     if user_id_from_data in users:
56         try:
57             movie_id_ratings[movie_id_from_data].append(movie_rating)
58         except KeyError:
59             movie_id_ratings[movie_id_from_data] = list()
60             movie_id_ratings[movie_id_from_data].append(movie_rating)
61 readData.close()
62
63 # Calculating the average.
64 for key in movie_id_ratings:
65     avg = sum(movie_id_ratings[key])/len(movie_id_ratings[key])
66     movie_userid_avg[key] = [float(avg)]
67
68 # Reading the u.item file for movie name.
69 for each_line in readItem:
70     split_each_line = each_line.strip().split('|')
71
72     movie_item_id = split_each_line[0]
73     movie_name_split = split_each_line[1].split('(1')
74     movie_name = movie_name_split[0]
75
76     try:
77         movie_userid_avg[movie_item_id].append(movie_name)
78     except KeyError:
79         pass
80 readItem.close()
81 print '*' * 55
82 print "Movies rated highest on average by men"
83 print "*" * 55
84 print "Movie Name\t\t\t\t\t", "Avg Rating"
85 print "-" * 55
86 # sorting the movies from highest to lowest based on the average value.
87 for key, value in sorted(movie_userid_avg.items(), key=lambda e: e[1], reverse=
True):
88     if (count <=5 ):
89         print '{:<50}{:<0.1f} '.format(value[1], value[0])
90         count += 1
91
92 if __name__ == "__main__":
93     try:
94         main()
95     except KeyboardInterrupt:
96         sys.exit(1)

```

9.3 Input Files

9.3.1 u.data

	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

9.3.2 u.item

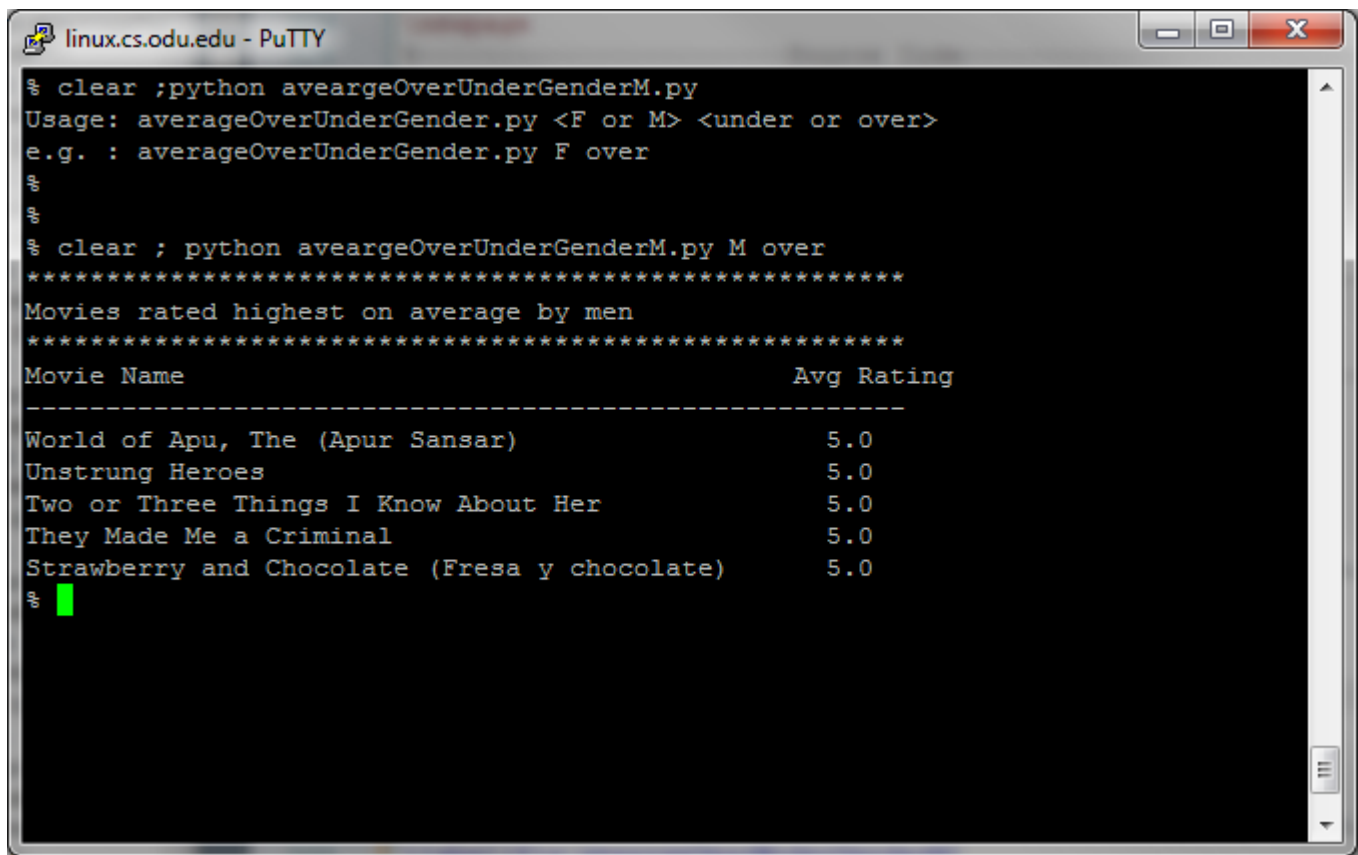
[illegible]

9.3.3 u.user

	userid	age	gender	occupation	zip_code
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344

9.4 Output Files

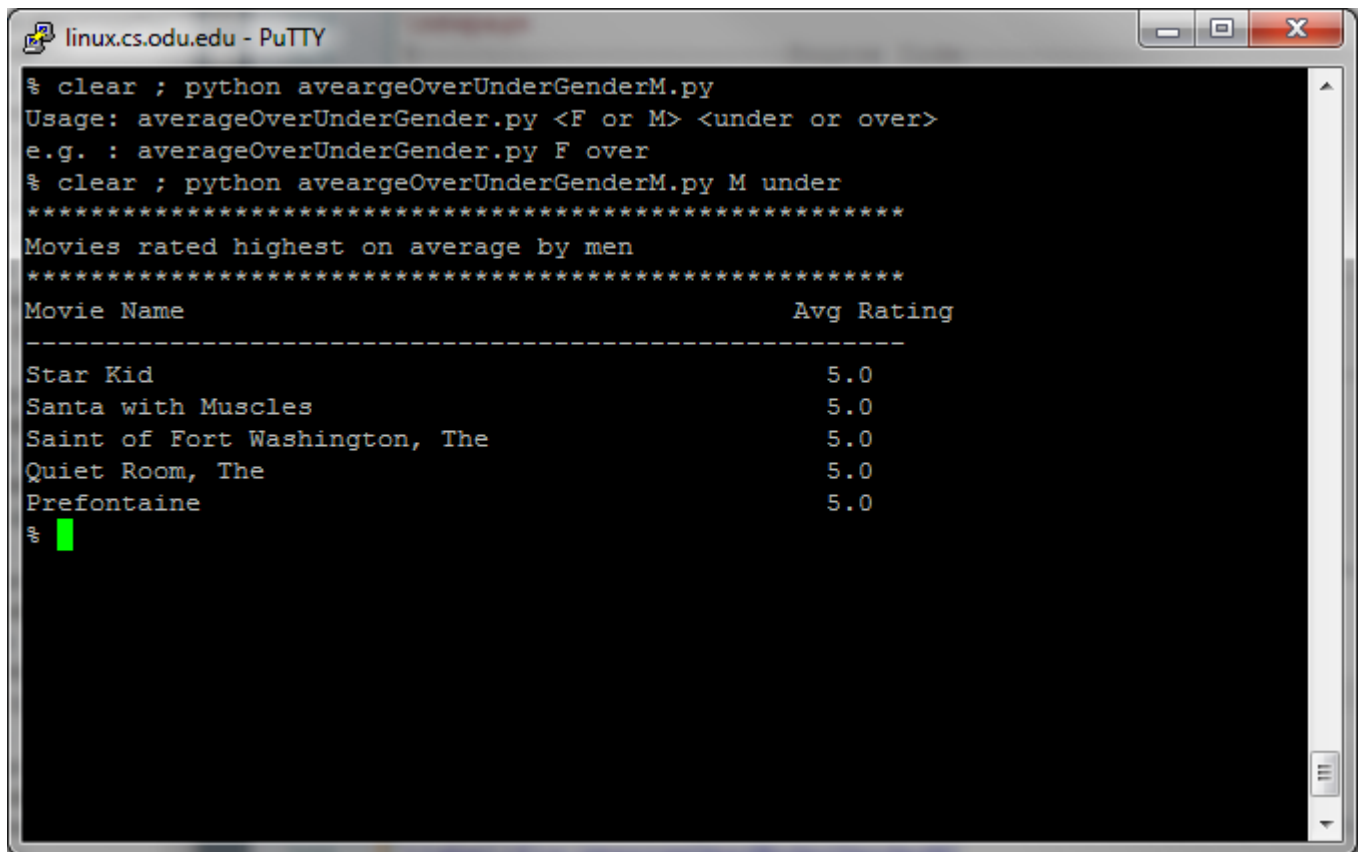
9.4.1 averageOverM.png



```
linux.cs.odu.edu - PuTTY
% clear ;python aveargeOverUnderGenderM.py
Usage: averageOverUnderGender.py <F or M> <under or over>
e.g. : averageOverUnderGender.py F over
%
%
% clear ; python aveargeOverUnderGenderM.py M over
*****
Movies rated highest on average by men
*****
Movie Name                               Avg Rating
-----
World of Apu, The (Apu Sansar)           5.0
Unstrung Heroes                           5.0
Two or Three Things I Know About Her      5.0
They Made Me a Criminal                   5.0
Strawberry and Chocolate (Fresa y chocolate) 5.0
% █
```

Figure 6: Movies rated highest on average by men over 40.

9.4.2 averageUnderM.png



```
linux.cs.odu.edu - PuTTY
% clear ; python aveargeOverUnderGenderM.py
Usage: averageOverUnderGender.py <F or M> <under or over>
e.g. : averageOverUnderGender.py F over
% clear ; python aveargeOverUnderGenderM.py M under
*****
Movies rated highest on average by men
*****
Movie Name                               Avg Rating
-----
Star Kid                                5.0
Santa with Muscles                      5.0
Saint of Fort Washington, The           5.0
Quiet Room, The                         5.0
Prefontaine                             5.0
% █
```

Figure 7: Movies rated highest on average by men under 40.

10 Question 10:

What movie was rated highest on average by women over 40? By women under 40?

10.1 Approach

1. The approach is same as question 9.
2. To get the corresponding output for this question it will still use `averageOverUnderGender.py` but the only difference is changing the arguments.
3. Figure 8 is the output showing how to execute the program and shows the top 5 movies which are rated highest on average by men “over” 40.
4. Figure 9 is the output showing how to execute the program and shows the top 5 movies which are rated highest on average by men “under” 40.

10.2 Source Code

10.2.1 averageOverUnderGender.py

```
1 #!/usr/bin/env python
2
3
4 # program to get the highest average ratings by women under and over 40.
5 import sys
6 import operator
7 def main():
8     # Take the arguments from the command line
9     numOfArgs=len(sys.argv)
10    if numOfArgs<3 or numOfArgs>3:
11
12        print 'Usage: averageOverUnderGender.py <F or M> <under or over>'
13        print 'e.g. : averageOverUnderGender.py F over '
14        sys.exit(1)
15    gender          = sys.argv[1]
16    gender_under_over= sys.argv[2]
17
18    # Create a dictionary
19    movie_userid_avg = {}
20    movie_id_ratings = {}
21    movie_userid_avg = {}
22    users            = {}
23    count            = 1
24
25    # Read the files
26    readData = open('/home/bbokka/cs594/A8/u.data', 'r')
27    readItem = open('/home/bbokka/cs594/A8/u.item', 'r')
28    readUser = open('/home/bbokka/cs594/A8/u.user', 'r')
29
30    # Reading u.user file to get the male or female
31    for line in readUser:
32        split_input_line = line.strip().split('|')
33        user_id_from_user= split_input_line[0]
34        user_age         = int(split_input_line[1])
35        user_gender      = split_input_line[2]
36
37        if gender_under_over == 'over' :
38            if(gender.upper() == user_gender and user_age > 40):
39                users[user_id_from_user] = [user_age]
40
41        if gender_under_over == 'under' :
42            if(gender.upper() == user_gender and user_age < 40):
43                users[user_id_from_user] = [user_age]
44
45    readUser.close()
46
47    # reading the u.data file for item and ratings.
48    for line in readData:
49        split_input_line = line.strip().split('\t')
50        user_id_from_data = split_input_line[0]
51        movie_id_from_data= split_input_line[1]
52        movie_rating      = float(split_input_line[2])
53
```

```

54         if user_id_from_data in users:
55             try:
56                 movie_id_ratings[movie_id_from_data].append(movie_rating)
57             except KeyError:
58                 movie_id_ratings[movie_id_from_data] = list()
59                 movie_id_ratings[movie_id_from_data].append(movie_rating)
60 readData.close()
61
62 # Calculating the average.
63 for key in movie_id_ratings:
64     avg = sum(movie_id_ratings[key])/len(movie_id_ratings[key])
65     movie_userid_avg[key] = [float(avg)]
66
67 # Reading the u.item file for movie name.
68 for each_line in readItem:
69     split_each_line = each_line.strip().split('|')
70
71     movie_item_id = split_each_line[0]
72     movie_name_split = split_each_line[1].split('(1')
73     movie_name = movie_name_split[0]
74
75     try:
76         movie_userid_avg[movie_item_id].append(movie_name)
77     except KeyError:
78         pass
79 readItem.close()
80 print '*' * 55
81 print "Movies rated highest on average by women"
82 print "*" * 55
83 print "Movie Name\t\t\t\t\t","Avg Rating"
84 print "-" * 55
85 # sorting the movies from highest to lowest based on the average value.
86 for key, value in sorted(movie_userid_avg.items(), key=lambda e: e[1], reverse=
True):
87     if (count <=5 ):
88         print '{:<50}{:<0.1f} '.format(value[1], value[0])
89         count += 1
90
91 if __name__ == "__main__":
92     try:
93         main()
94     except KeyboardInterrupt:
95         sys.exit(1)

```

10.3 Input Files

10.3.1 u.data

	userid	itemid	rating	timestamp
1				
2				
3	196	242	3	881250949
4	186	302	3	891717742
5	22	377	1	878887116
6	244	51	2	880606923
7	166	346	1	886397596
8	298	474	4	884182806
9	115	265	2	881171488

10.3.2 u.item

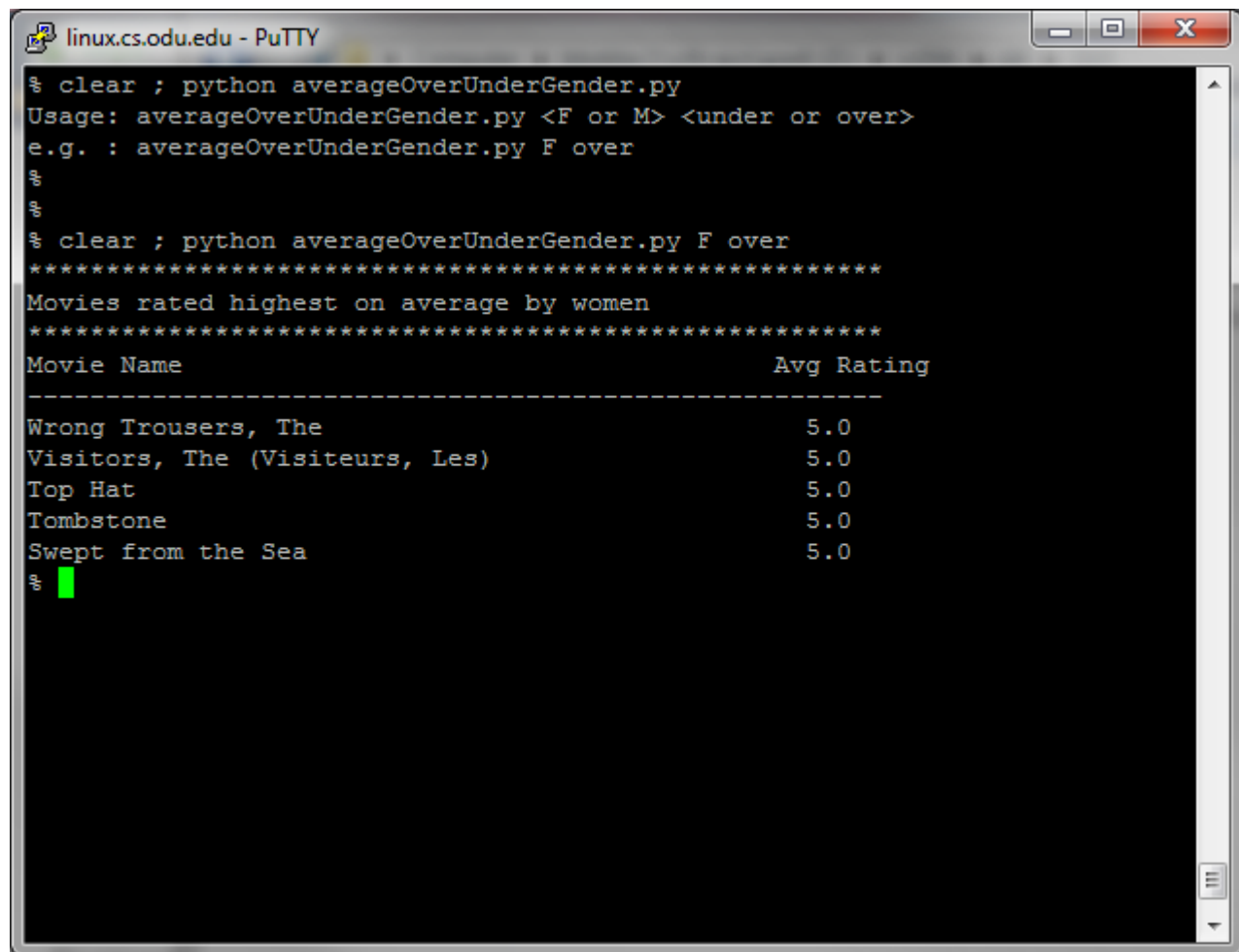
movie id	movie title	release date	video release date	IMDb URL	unknown
	Action Adventure Animation Children's Comedy Crime Documentary Drama Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western				
1	Toy Story (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Toy%20Story%20(1995)	
2	GoldenEye (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?GoldenEye%20(1995)	
3	Four Rooms (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Four%20Rooms%20(1995)	
4	Get Shorty (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Get%20Shorty%20(1995)	
5	Copycat (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Copycat%20(1995)	
6	Shanghai Triad (Yao a yao yao dao waipo qiao) (1995)	01-Jan-1995		http://us.imdb.com/Title?Yao+a+yao+yao+dao+waipo+qiao+(1995)	
7	Twelve Monkeys (1995)	01-Jan-1995		http://us.imdb.com/M/title-exact?Twelve%20Monkeys%20(1995)	

10.3.3 u.user

	userid	age	gender	occupation	zip_code
1	1	24	M	technician	85711
2	2	53	F	other	94043
3	3	23	M	writer	32067
4	4	24	M	technician	43537
5	5	33	F	other	15213
6	6	42	M	executive	98101
7	7	57	M	administrator	91344

10.4 Output Files

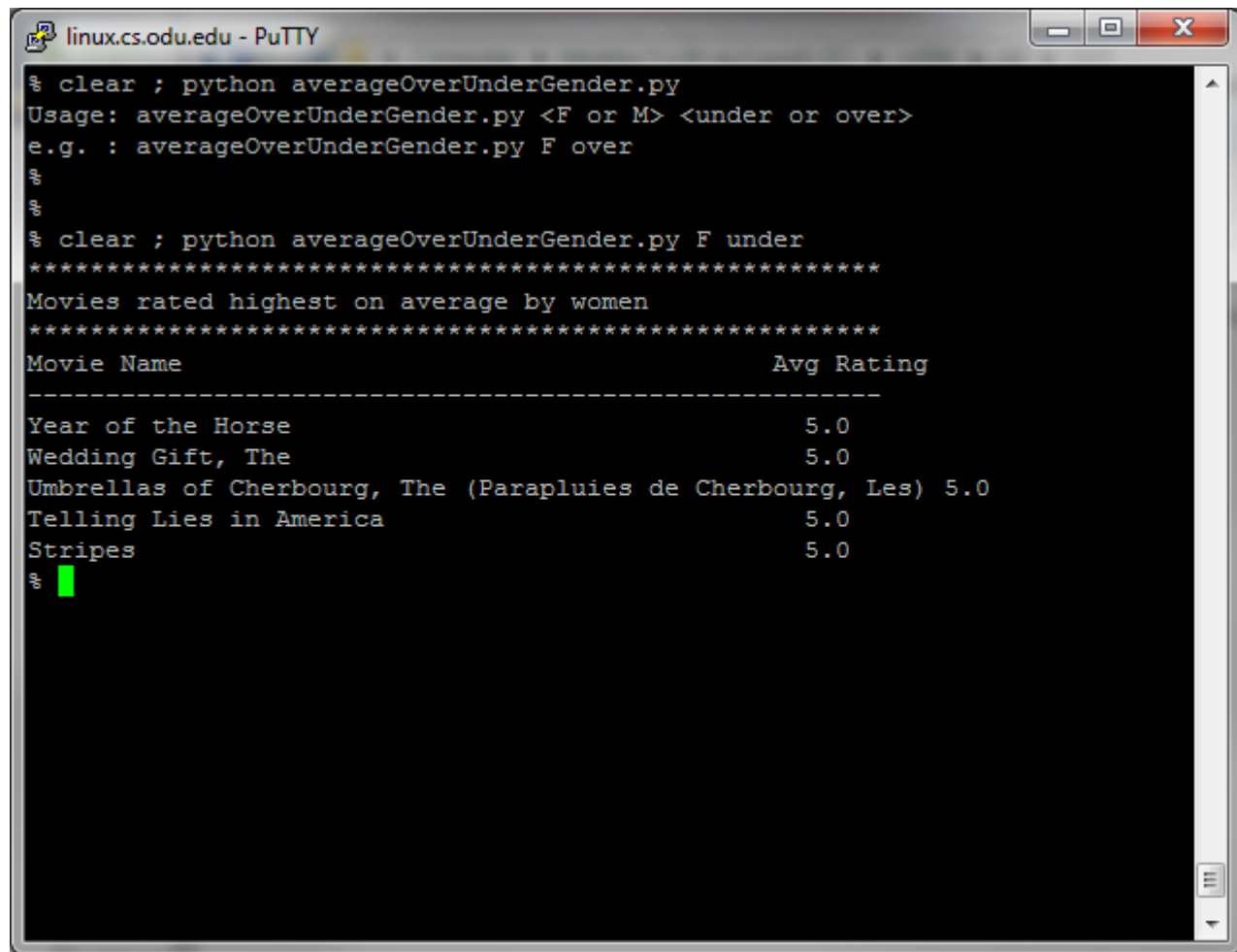
10.4.1 averageOverF.png



```
linux.cs.odu.edu - PuTTY
% clear ; python averageOverUnderGender.py
Usage: averageOverUnderGender.py <F or M> <under or over>
e.g. : averageOverUnderGender.py F over
%
%
% clear ; python averageOverUnderGender.py F over
*****
Movies rated highest on average by women
*****
Movie Name                               Avg Rating
-----
Wrong Trousers, The                      5.0
Visitors, The (Visiteurs, Les)           5.0
Top Hat                                  5.0
Tombstone                                5.0
Swept from the Sea                       5.0
% █
```

Figure 8: Movies rated highest on average by women over 40.

10.4.2 averageUnderF.png



```
linux.cs.odu.edu - PuTTY
% clear ; python averageOverUnderGender.py
Usage: averageOverUnderGender.py <F or M> <under or over>
e.g. : averageOverUnderGender.py F over
%
%
% clear ; python averageOverUnderGender.py F under
*****
Movies rated highest on average by women
*****
Movie Name                               Avg Rating
-----
Year of the Horse                        5.0
Wedding Gift, The                        5.0
Umbrellas of Cherbourg, The (Parapluies de Cherbourg, Les) 5.0
Telling Lies in America                  5.0
Stripes                                  5.0
% █
```

Figure 9: Movies rated highest on average by women under 40.

References

- [1] Data files used for assignment. <http://files.grouplens.org/datasets/movielens/ml-100k/>, April 1998. Data collected by GroupLens Research Project.
- [2] K. Arthur Endsley. recommendations.py program. <https://github.com/arthur-e/Programming-Collective-Intelligence/blob/master/chapter2/recommendations.py>, December 2012.
- [3] Frederick Hamidi. Python: Run function from the command line. <http://stackoverflow.com/questions/3987041/python-run-function-from-the-command-line>, October 2010.
- [4] Devin Jeanpierre. Sort a python dictionary by value. <http://stackoverflow.com/questions/613183/sort-a-python-dictionary-by-value>, March 2009.
- [5] Toby Segaran. *Programming Collective Intelligence*. O'Reilly Media, August 2007.

□