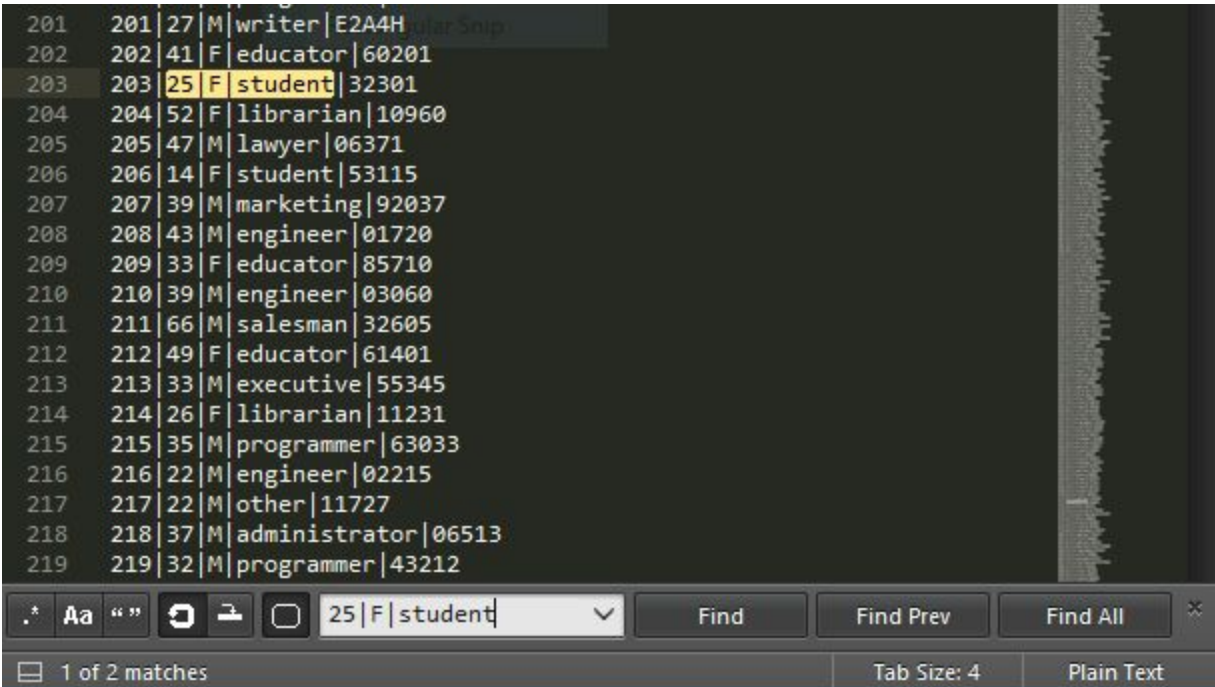


PART 1

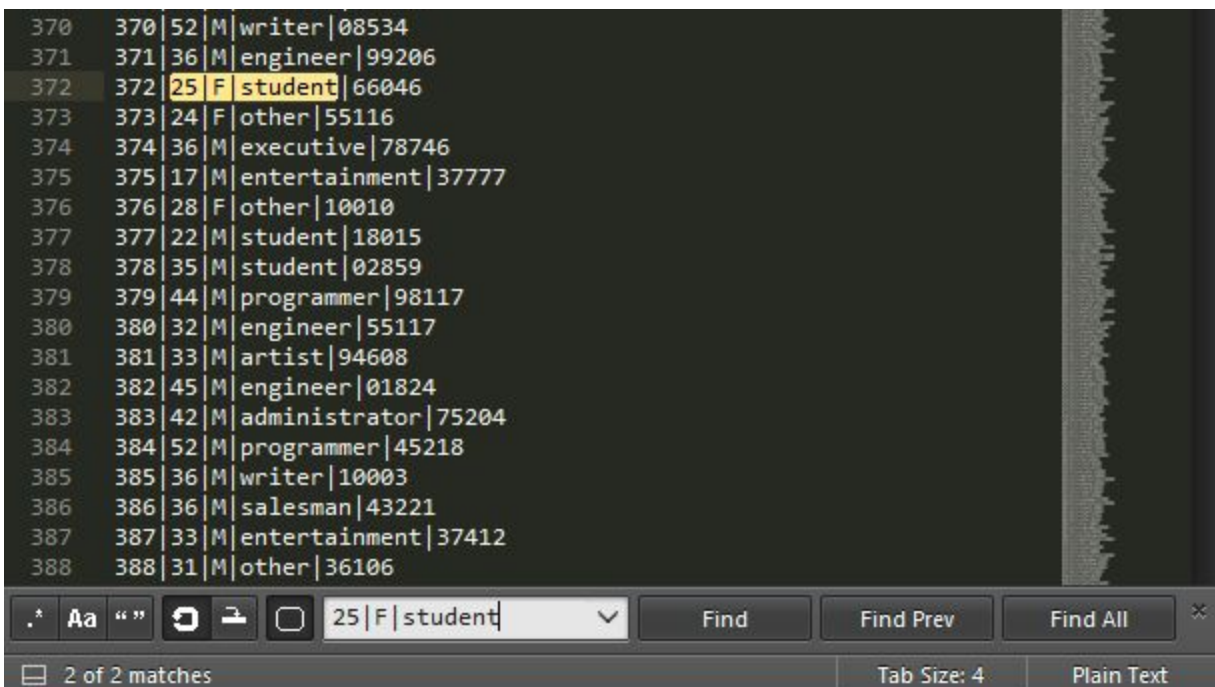
To find 3 users similar to myself in u.user, I just searched for "25|F|student" and found two exact matches:



```
201 201|27|M|writer|E2A4H...
202 202|41|F|educator|60201
203 203|25|F|student|32301
204 204|52|F|librarian|10960
205 205|47|M|lawyer|06371
206 206|14|F|student|53115
207 207|39|M|marketing|92037
208 208|43|M|engineer|01720
209 209|33|F|educator|85710
210 210|39|M|engineer|03060
211 211|66|M|salesman|32605
212 212|49|F|educator|61401
213 213|33|M|executive|55345
214 214|26|F|librarian|11231
215 215|35|M|programmer|63033
216 216|22|M|engineer|02215
217 217|22|M|other|11727
218 218|37|M|administrator|06513
219 219|32|M|programmer|43212
```

.* Aa " " Find Find Prev Find All

1 of 2 matches Tab Size: 4 Plain Text

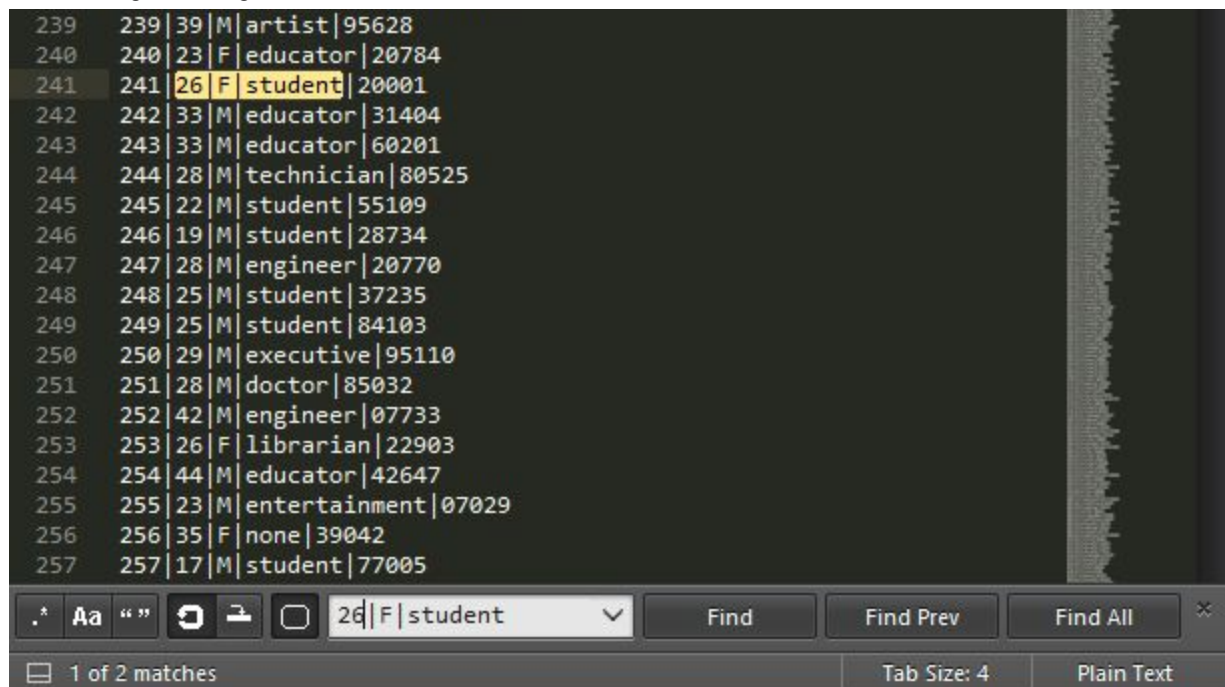


```
370 370|52|M|writer|08534
371 371|36|M|engineer|99206
372 372|25|F|student|66046
373 373|24|F|other|55116
374 374|36|M|executive|78746
375 375|17|M|entertainment|37777
376 376|28|F|other|10010
377 377|22|M|student|18015
378 378|35|M|student|02859
379 379|44|M|programmer|98117
380 380|32|M|engineer|55117
381 381|33|M|artist|94608
382 382|45|M|engineer|01824
383 383|42|M|administrator|75204
384 384|52|M|programmer|45218
385 385|36|M|writer|10003
386 386|36|M|salesman|43221
387 387|33|M|entertainment|37412
388 388|31|M|other|36106
```

.* Aa " " Find Find Prev Find All

2 of 2 matches Tab Size: 4 Plain Text

Then, changed the age to “26” for a third match:

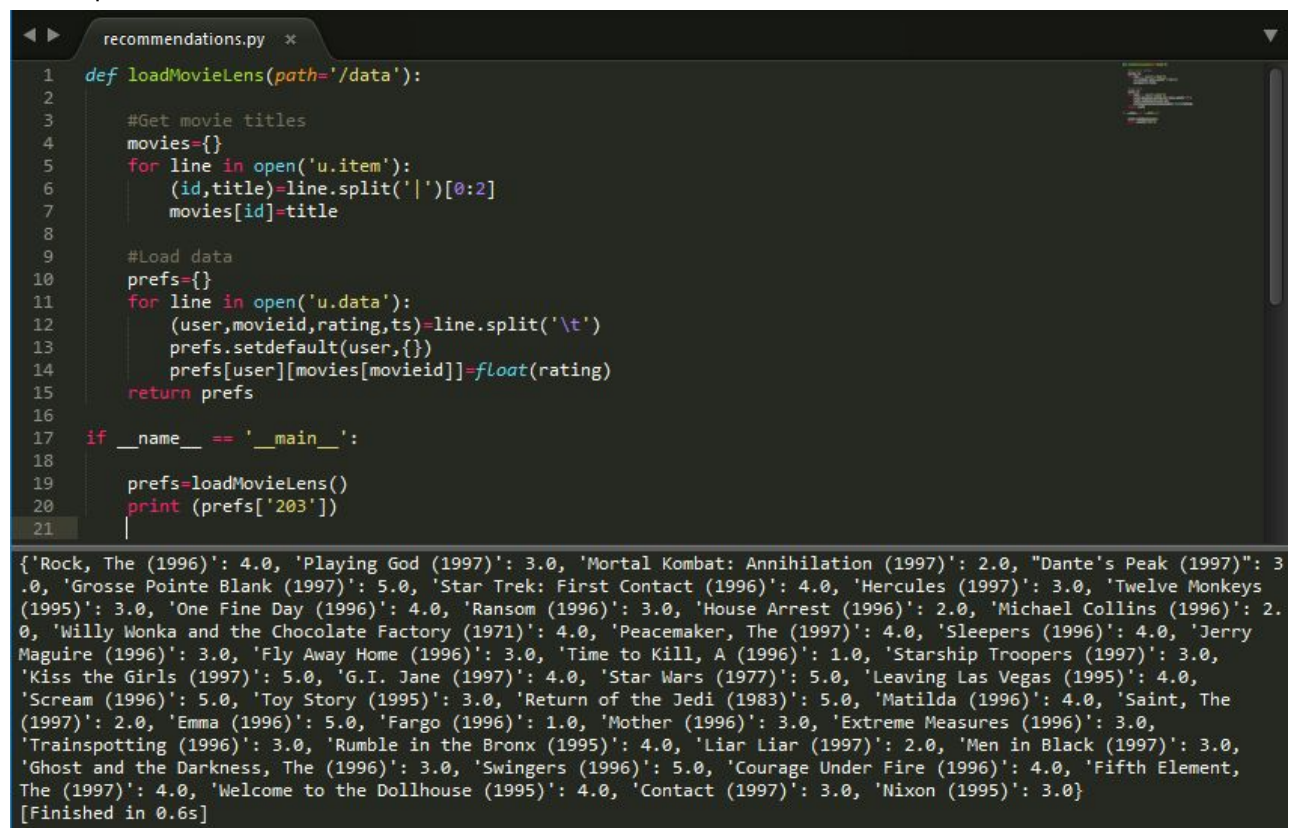


The screenshot shows a text editor with a list of user profiles. Each profile is a line of text in the format: `user_id | age | gender | occupation | user_id`. The search bar at the bottom contains the text `26 | F | student`. The search results show 2 matches.

User ID	Age	Gender	Occupation	User ID
239	39	M	artist	95628
240	23	F	educator	20784
241	26	F	student	20001
242	33	M	educator	31404
243	33	M	educator	60201
244	28	M	technician	80525
245	22	M	student	55109
246	19	M	student	28734
247	28	M	engineer	20770
248	25	M	student	37235
249	25	M	student	84103
250	29	M	executive	95110
251	28	M	doctor	85032
252	42	M	engineer	07733
253	26	F	librarian	22903
254	44	M	educator	42647
255	23	M	entertainment	07029
256	35	F	none	39042
257	17	M	student	77005

Users **203**, **372**, and **241** were matches. For this assignment, I modified the code on page 26 of the book Programming Collective Intelligence. I just changed the paths and entered the user IDs of my matches.

#203's preferences:



The screenshot shows a Python script named `recommendations.py` with the following code:

```
def loadMovieLens(path='/data'):
    #Get movie titles
    movies={}
    for line in open('u.item'):
        (id,title)=line.split('|')[0:2]
        movies[id]=title

    #Load data
    prefs={}
    for line in open('u.data'):
        (user,movieid,rating,ts)=line.split('\t')
        prefs.setdefault(user,{})
        prefs[user][movies[movieid]]=float(rating)
    return prefs

if __name__ == '__main__':
    prefs=loadMovieLens()
    print (prefs['203'])
```

The output of the script is a list of movie titles and their ratings for user 203:

```
{'Rock, The (1996)': 4.0, 'Playing God (1997)': 3.0, 'Mortal Kombat: Annihilation (1997)': 2.0, "Dante's Peak (1997)": 3.0, 'Grosse Pointe Blank (1997)': 5.0, 'Star Trek: First Contact (1996)': 4.0, 'Hercules (1997)': 3.0, 'Twelve Monkeys (1995)': 3.0, 'One Fine Day (1996)': 4.0, 'Ransom (1996)': 3.0, 'House Arrest (1996)': 2.0, 'Michael Collins (1996)': 2.0, 'Willy Wonka & the Chocolate Factory (1971)': 4.0, 'Peacemaker, The (1997)': 4.0, 'Sleepers (1996)': 4.0, 'Jerry Maguire (1996)': 3.0, 'Fly Away Home (1996)': 3.0, 'Time to Kill, A (1996)': 1.0, 'Starship Troopers (1997)': 3.0, 'Kiss the Girls (1997)': 5.0, 'G.I. Jane (1997)': 4.0, 'Star Wars (1977)': 5.0, 'Leaving Las Vegas (1995)': 4.0, 'Scream (1996)': 5.0, 'Toy Story (1995)': 3.0, 'Return of the Jedi (1983)': 5.0, 'Matilda (1996)': 4.0, 'Saint, The (1997)': 2.0, 'Emma (1996)': 5.0, ' Fargo (1996)': 1.0, 'Mother (1996)': 3.0, 'Extreme Measures (1996)': 3.0, 'Trainspotting (1996)': 3.0, 'Rumble in the Bronx (1995)': 4.0, 'Liar Liar (1997)': 2.0, 'Men in Black (1997)': 3.0, 'Ghost and the Darkness, The (1996)': 3.0, 'Swingers (1996)': 5.0, 'Courage Under Fire (1996)': 4.0, 'Fifth Element, The (1997)': 4.0, 'Welcome to the Dollhouse (1995)': 4.0, 'Contact (1997)': 3.0, 'Nixon (1995)': 3.0}
[Finished in 0.6s]
```


#372's preferences:

```
recommendations.py x
1 def loadMovieLens(path='/data'):
2
3     #Get movie titles
4     movies={}
5     for line in open('u.item'):
6         (id,title)=line.split('|')[0:2]
7         movies[id]=title
8
9     #Load data
10    prefs={}
11    for line in open('u.data'):
12        (user,movieid,rating,ts)=line.split('\t')
13        prefs.setdefault(user,{})
14        prefs[user][movies[movieid]]=float(rating)
15    return prefs
16
17 if __name__ == '__main__':
18
19     prefs=loadMovieLens()
20     print (prefs['372'])
21
{'Career Girls (1997)': 4.0, 'Abyss, The (1989)': 4.0, 'Crash (1996)': 4.0, 'Jaws 2 (1978)': 4.0, 'Carrie (1976)': 5.0,
'Candyman (1992)': 5.0, 'Interview with the Vampire (1994)': 4.0, 'Jaws (1975)': 5.0, 'Psycho (1960)': 5.0, 'Sliver
(1993)': 5.0, 'Firm, The (1993)': 5.0, 'Amityville Horror, The (1979)': 4.0, 'Omen, The (1976)': 4.0, 'American
Werewolf in London, An (1981)': 5.0, 'Birds, The (1963)': 4.0, 'Color of Night (1994)': 4.0, 'Cape Fear (1991)': 5.0,
'Volcano (1997)': 4.0, 'Boxing Helena (1993)': 4.0, 'Pulp Fiction (1994)': 4.0, 'Fugitive, The (1993)': 5.0, 'Sleepers
(1996)': 4.0, 'Fog, The (1980)': 5.0, 'Usual Suspects, The (1995)': 4.0, 'Taxi Driver (1976)': 5.0, 'Evil Dead II
(1987)': 2.0, 'Burnt Offerings (1976)': 4.0, 'Murder at 1600 (1997)': 3.0, 'Twelve Monkeys (1995)': 3.0, 'Hoodlum
(1997)': 4.0, 'Rosewood (1997)': 5.0, 'Kiss the Girls (1997)': 4.0, 'Silence of the Lambs, The (1991)': 5.0, 'Natural
Born Killers (1994)': 5.0, 'Heat (1995)': 5.0, 'In the Company of Men (1997)': 4.0, 'Albino Alligator (1996)': 3.0,
'Fargo (1996)': 3.0, 'Game, The (1997)': 5.0, 'Basic Instinct (1992)': 5.0, 'Love Jones (1997)': 4.0, 'Copycat (1995)':
4.0, 'Howling, The (1981)': 4.0, 'Cop Land (1997)': 5.0, 'Flesh and Bone (1993)': 4.0, 'Nightmare on Elm Street, A
(1984)': 5.0, 'Shining, The (1980)': 5.0, 'Mary Shelley's Frankenstein (1994)': 5.0, 'English Patient, The (1996)': 5.
0, 'Once Upon a Time in America (1984)': 3.0, 'Cat People (1982)': 5.0, 'Ghost and the Darkness, The (1996)': 5.0,
'Scream (1996)': 5.0, 'G.I. Jane (1997)': 4.0, 'Dolores Claiborne (1994)': 4.0, 'Bound (1996)': 4.0, 'City Hall
(1996)': 4.0, 'She's So Lovely (1997)': 4.0, 'Kalifornia (1993)': 5.0, 'Alien (1979)': 5.0, 'Fan, The (1996)': 4.0,
'Young Poisoner's Handbook, The (1995)': 5.0, 'Freeway (1996)': 4.0, 'Aliens (1986)': 3.0, 'Mimic (1997)': 4.0, 'Death
and the Maiden (1994)': 4.0}
[Finished in 0.4s]
```

#241's preferences:

```
recommendations.py x
1 def loadMovieLens(path='/data'):
2
3     #Get movie titles
4     movies={}
5     for line in open('u.item'):
6         (id,title)=line.split('|')[0:2]
7         movies[id]=title
8
9     #Load data
10    prefs={}
11    for line in open('u.data'):
12        (user,movieid,rating,ts)=line.split('\t')
13        prefs.setdefault(user,{})
14        prefs[user][movies[movieid]]=float(rating)
15    return prefs
16
17 if __name__ == '__main__':
18
19     prefs=loadMovieLens()
20     print (prefs['241'])
21
{'Seven Years in Tibet (1997)': 2.0, 'Amistad (1997)': 5.0, 'Fallen (1998)': 2.0, 'Rainmaker, The (1997)': 4.0,
'Rosewood (1997)': 4.0, 'English Patient, The (1996)': 5.0, 'Chasing Amy (1997)': 4.0, 'Liar Liar (1997)': 3.0, 'How to
Be a Player (1997)': 3.0, 'I Know What You Did Last Summer (1997)': 2.0, 'Jackie Brown (1997)': 3.0, 'Alien:
Resurrection (1997)': 2.0, 'Kiss the Girls (1997)': 3.0, 'Jackal, The (1997)': 3.0, 'Gattaca (1997)': 3.0, 'Devil's
Advocate, The (1997)': 4.0, 'Titanic (1997)': 4.0, 'Scream 2 (1997)': 2.0, 'L.A. Confidential (1997)': 3.0, 'Air Force
One (1997)': 4.0, 'Soul Food (1997)': 5.0, 'Eve's Bayou (1997)': 4.0, 'Scream (1996)': 5.0}
[Finished in 0.4s]
```

Here is a table of the users with their top and bottom 3 films:

USER ID	TOP 3 FILMS	BOTTOM 3 FILMS
203	Grosse Pointe Blank (5) Scream (5) Star Wars (5)	A Time to Kill (1) Fargo (1) Mortal Kombat: Annihilation (2)
372	Psycho (5) The Shining (5) The Silence of the Lambs (5)	Evil Dead II (2) Aliens (3) Fargo (3)
241	Amistad (5) Scream (5) Soul Food (5)	Alien: Resurrection (2) Fallen (2) Scream II (2)

I identify most with **#372**.

PART 2

To find each user's correlation score with #372, I used the code for the Pearson Correlation Score on page 13 of Programming Collective Intelligence, which returns a value between -1 and 1. If a value of 1 is returned, then those two people have identical ratings.

An example of getting the correlation scores for all 943 users, just to make sure it works:

The screenshot shows a terminal window on the left and a code editor on the right. The terminal displays the output of a Python script, showing correlation scores for each user from 907 to 943. The code editor shows the Python script used to calculate these scores.

```

rachel@Rachel:~/Desktop/CS_432/A7$ python recommendations.py > user_correlations.txt
rachel@Rachel:~/Desktop/CS_432/A7$

user_correlations.txt (~/Desktop/CS_432/A7) - gedit
907 -0.283069258536
908 0.350542603629
909 0
910 0.0917249232132
911 -0.172413793103
912 0.0
913 -0.238144836104
914 0
915 0
916 -0.256137363622
917 1.0
918 0
919 0.0725853073789
920 0.342997170285
921 -0.0625
922 0.0975609756098
923 0.241746889208
924 -0.0725954088641
925 0.0184868466662
926 0.632455532034
927 0.454545454545
928 0.333333333333
929 0.216930457819
930 -0.666666666667
931 0.636363636364
932 -0.438141986419
933 -0.211829636434
934 -0.5356556823
935 0.866025403784
936 0.121566134771
937 0.324442842262
938 -0.0440225453163
939 0
940 -0.0842151921067
941 -1.0
942 0.585540043769
943 -0.220234468049

recommendations.py
1 from math import sqrt
2
3 def loadMovieLens(path='data'):
4     #Get movie titles
5     movies={}
6     for line in open('u.item'):
7         (id,title)=line.split('|')[0:2]
8         movies[id]=title
9     #Load data
10    prefs={}
11    for line in open('u.data'):
12        (user,movieid,rating,ts)=line.split('\t')
13        prefs.setdefault(user, {})
14        prefs[user][movies[movieid]]=float(rating)
15    return prefs
16
17 def sim_pearson(prefs,p1,p2):
18     #Get the list of mutually rated items
19     si={}
20     for item in prefs[p1]:
21         if item in prefs[p2]:
22             si[item]=1
23     #Find the number of elements
24     n=len(si)
25     #If there are no ratings in common, return 0
26     if n==0: return 0
27     #Add up all the preferences
28     sum1=sum([prefs[p1][it] for it in si])
29     sum2=sum([prefs[p2][it] for it in si])
30     #Sum up the squares
31     sum1Sq=sum([pow(prefs[p1][it],2) for it in si])
32     sum2Sq=sum([pow(prefs[p2][it],2) for it in si])
33     #Sum up the products
34     pSum=sum([prefs[p1][it]*prefs[p2][it] for it in si])
35     #Calculate Pearson score
36     num=pSum-(sum1*sum2/n)
37     den=sqrt((sum1Sq-pow(sum1,2)/n)*(sum2Sq-pow(sum2,2)/n))
38     if den==0:
39         return 0
40     r=num/den
41     return r
42
43 if __name__ == '__main__':
44     prefs=loadMovieLens()
45     # print (prefs['372'])
46     # print sim_pearson(prefs, '372', '1')
47     for i in range(1,944):
48         print str(i), sim_pearson(prefs, '372', str(i))
49

```

This next part uses code from page 14 of Programming Collective Intelligence to find the 5 most correlated users to #372: **904, 888, 857, 837, 779**.

```
recommendations.py x
43 def topMatches(prefs, person, n=5, similarity=sim_pearson):
44     scores=[(similarity(prefs, person, other), other)
45             for other in prefs if other!=person]
46     #Sort the list so the highest scores appear at the top
47     scores.sort()
48     # scores.sort(reverse=True)
49     scores.reverse()
50     return scores[0:n]
51
52 if __name__ == '__main__':
53     prefs=loadMovieLens()
54     # print (prefs['372'])
55     # print sim_pearson(prefs, '372', '1')
56     # for i in range (1,944):
57     #     print str(i), sim_pearson(prefs, '372', str(i))
58     print topMatches(prefs, '372')
59     # print getRecommendations(prefs, '372')
60
```

```
[(1.0, '904'), (1.0, '888'), (1.0, '857'), (1.0, '837'), (1.0, '779')]
[Finished in 0.2s]
```

Least correlated users: **139, 419, 594, 89, 156**.

```
recommendations.py x
43 def topMatches(prefs, person, n=5, similarity=sim_pearson):
44     scores=[(similarity(prefs, person, other), other)
45             for other in prefs if other!=person]
46     #Sort the list so the highest scores appear at the top
47     # scores.sort()
48     scores.sort(reverse=True)
49     scores.reverse()
50     return scores[0:n]
51
52 if __name__ == '__main__':
53     prefs=loadMovieLens()
54     # print (prefs['372'])
55     # print sim_pearson(prefs, '372', '1')
56     # for i in range (1,944):
57     #     print str(i), sim_pearson(prefs, '372', str(i))
58     print topMatches(prefs, '372')
59     # print getRecommendations(prefs, '372')
60
```

```
[(-1.0000000000000004, '139'), (-1.0000000000000004, '419'), (-1.
0000000000000004, '594'), (-1.0000000000000004, '89'), (-1.0, '156')]
[Finished in 0.2s]
```


I used the code from page 16 of Programming Collective Intelligence to compute ratings for all the films user #372 hasn't seen. It loops through every other person, calculating how similar they are to user #372. It then loops through every item for which they've given a score. The score for each item is multiplied by the similarity and these products are all added together. At the end, the scores are normalized by dividing each of them by the similarity sum, and the sorted results are returned.

Demo of the code and the resulting list of movie recommendations (Just realized it looks like they all ranked as 1.0 in this picture, but really I had just scrolled to the bottom):

The image shows a dual-pane window from a terminal or IDE. The top pane displays a Python script named `movie_recommendations.txt` with the following content:

```
#!/usr/bin/env python3
# python recommendations.py > movie_recommendations.txt
# python recommendations.py > movie_recommendations.txt

# Gets recommendations for a person by using a weighted average
# of every other user's rankings
def getRecommendations(prefs, person, similarity=sim_pearson):
    totals={}
    simSums={}
    for other in prefs:
        # don't compare me to myself
        if other==person: continue
        sim=similarity(prefs, person, other)
        # ignore scores of zero or lower
        if sim==0: continue
        for item in prefs[other]:
            # only score movies I haven't seen yet
            if item not in prefs[person] or prefs[person][item]==0:
                # similarity * score
                totals.setdefault(item,0)
                totals[item]+=prefs[other][item]*sim
    # sum of similarities
    simSums.setdefault(item,0)
    simSums[item]+=sim
    # Create the normalized list
    rankings=[(total,simSums[item],item) for item,total in totals.items()]
    # Return the sorted list
    rankings.sort()
    rankings.reverse()
    print rankings
    return rankings

if __name__ == '__main__':
    prefs=loadMovieLens()
    # print(prefs['372'])
    # print sim_pearson(prefs, '372', '1')
    # for i in range(1,1944):
    #     print str(i), sim_pearson(prefs, '372', str(i))
    getRecommendations(prefs, '372')
```

The bottom pane shows the output of the script, which is a long list of movie titles and their similarity scores, sorted in descending order. The output starts with:

```
(1.0, 'Stars Fell on Henrietta, the (1955)', (1.219487949127182, 'Ill Gotten Gains (1997)', (1.2137825186909414, 'Glass Shield, The (1994)', (1.1655143670731911, 'Jefferson in Paris (1995)', (1.148069757194137, 'Relative Fear (1994)', (1.1322180743704486, 'Twin Town (1997)', (1.0952580379859007, 'Paris, France (1993)', (1.0599807865191517, 'Hunted, The (1995)', (1.0573201799840388, 'Children of the Corn: The Gathering (1996)', (1.0494015305387756, 'Bad Company (1995)', (1.0, 'Vankee Zulu (1996)', (1.0, 'Woman in Question, The (1958)', (1.0, 'Wend Kuindt (and's eff) (1982)', (1.0, 'Wie est belle, La (Life is Rosey) (1987)', (1.0, 'Vermont Is For Lovers (1992)', (1.0, 'Venice/Venice (1992)', (1.0, 'Underworld (1997)', (1.0, 'Two or Three Things I Know About Her (1966)', (1.0, 'Two Deaths (1995)', (1.0, 'Turbo: A Power Rangers Movie (1997)', (1.0, 'Toukt Boukt (Journey of the Hyena) (1973)', (1.0, 'Tough and Deadly (1995)', (1.0, 'Total Eclipse (1995)', (1.0, 'To Have, or Not (1995)', (1.0, 'To Cross the Rubicon (1991)', (1.0, 'Tigero: A Film That Was Never Made (1994)', (1.0, 'The Innocent (1994)', (1.0, 'The Courtyard (1995)', (1.0, 'Terror in a Texas Town (1958)', (1.0, 'T-Men (1947)', (1.0, 'Sudenne pastorale, La (1946)', (1.0, 'Sweet from the Sea (1997)', (1.0, 'Sydney Manhattan (1996)', (1.0, 'Squeeze (1996)', (1.0, 'Spirits of the Dead (Tre passi nel delirio) (1968)', (1.0, 'Somebody to Love (1994)', (1.0, 'Shooter, The (1995)', (1.0, 'Shadows (Clenie) (1988)', (1.0, 'Second Jungle Book: Mowgli & Baloo, The (1997)', (1.0, 'Schizopolis (1996)', (1.0, 'Salut cousin! (1996)', (1.0, 'Rough Magic (1995)', (1.0, 'Quartier Mozart (1992)', (1.0, 'Pushing Hands (1992)', (1.0, 'Promtise, The (Versprechen, Das) (1994)', (1.0, 'Phat Beach (1996)', (1.0, 'Pharaoh's Army (1995)', (1.0, 'Panther (1995)', (1.0, 'Office Killer (1997)', (1.0, 'Nobody Loves Me (Kellner Liebt mich) (1994)', (1.0, 'Milk Cop (1997)', (1.0, 'New Age, The (1994)', (1.0, 'National Lampoon's Senior Trip (1995)', (1.0, 'Maggie (1997)', (1.0, 'Mighty, The (1998)', (1.0, 'Men of Means (1998)', (1.0, 'Mat i syn (1997)', (1.0, 'Man from Down Under, The (1943)', (1.0, 'Liebelet (1933)', (1.0, 'Lashou shentan (1992)', (1.0, 'King of New York (1990)', (1.0, 'Jerky Boys, The (1994)', (1.0, 'It Takes Two (1995)', (1.0, 'Invitation, The (Zaproszenie) (1986)', (1.0, 'Infinity (1996)', (1.0, 'I, Worst of All (Yo, la peor de todas) (1990)', (1.0, 'Hungarian Fairy Tale, A (1987)', (1.0, 'Hostile Intentions (1994)', (1.0, 'Horse Whisperer, The (1998)', (1.0, 'Harlem (1993)', (1.0, 'Gordy (1995)', (1.0, 'Getting Even with Dad (1994)', (1.0, 'Getting Away With Murder (1996)', (1.0, 'Germinal (1993)', (1.0, 'For Ever Mozart (1997)', (1.0, 'Falling in Love Again (1980)', (1.0, 'Eye of Vichy, The (Oeil de Vichy, L') (1993)', (1.0, 'Ed (1996)', (1.0, 'Dunston Checks In (1996)', (1.0, 'Delta of Venus (1994)', (1.0, 'Death in the Garden (Mort en ce jardin, La) (1956)', (1.0, 'Daens (1992)', (1.0, 'Condition Red (1995)', (1.0, 'Catwalk (1995)', (1.0, 'Castle Freak (1995)', (1.0, 'Carmen Miranda: Bananas Is My Business (1994)', (1.0, 'Careful (1992)', (1.0, 'Butterfly Kiss (1995)', (1.0, 'Boys in Venice (1996)', (1.0, 'Bonheur, Le (1965)', (1.0, 'Beyond Bedlam (1993)', (1.0, 'Baton Rouge (1988)', (1.0, 'Babyfever (1994)', (1.0, 'B*A*P*S (1997)', (1.0, 'Amityville: Dollhouse (1996)', (1.0, 'Amityville: A New Generation (1993)', (1.0, 'Amityville 1992: It's About Time (1992)', (1.0, '3 Ninjas: High Noon At Mega Mountain (1998)']]
```

The bottom pane also shows a status bar with the text "Tab Width: 36" and "Tab Size: 4".

Top 5 movie recommendations for user #372:

- Stripes (5.0)
Star Kid (5.0)
Someone Else's America (5.0)
Prefontaine (5.0)
Mondo (5.0)

Bottom 5 movie recommendations for user #372:

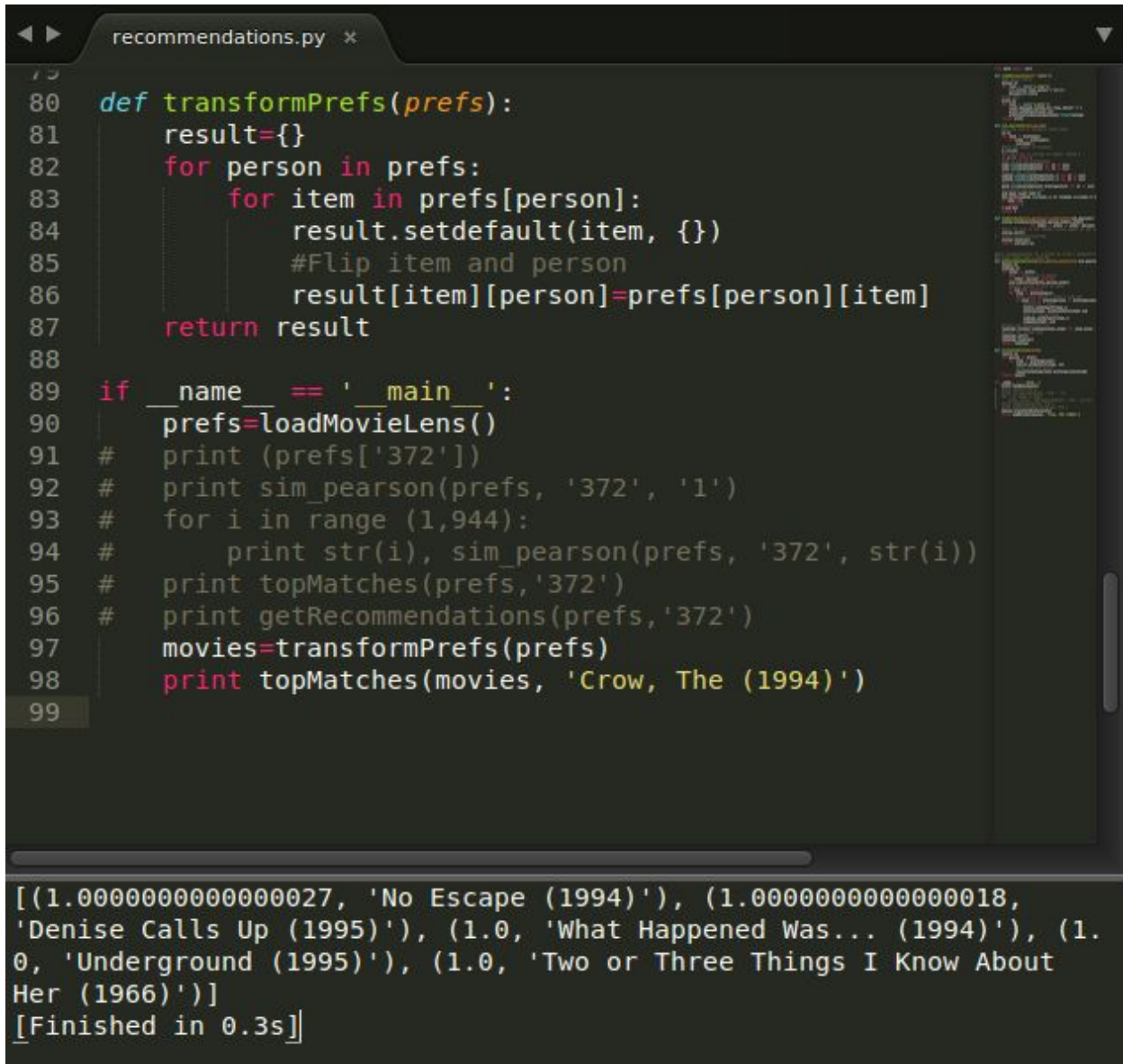
- 3 Ninjas: High Noon At Mega Mountain (1.0)
Amityville 1992: It's About Time (1.0)
Amityville: A New Generation (1.0)
Amityville: Dollhouse (1.0)
B*A*P*S (1.0)

PART 4

I'm going to go with The Crow (#68) as my favorite movie and Pocahontas (#542) as my least favorite.

Using the code on page 18 of Programming Collective Intelligence, I switched the user and item IDs so the topMatches function could be re-used for finding similar movies.

Top 5 correlated movies to The Crow:



```
79
80 def transformPrefs(prefs):
81     result={}
82     for person in prefs:
83         for item in prefs[person]:
84             result.setdefault(item, {})
85             #Flip item and person
86             result[item][person]=prefs[person][item]
87     return result
88
89 if __name__ == '__main__':
90     prefs=loadMovieLens()
91     # print (prefs['372'])
92     # print sim_pearson(prefs, '372', '1')
93     # for i in range (1,944):
94     #     print str(i), sim_pearson(prefs, '372', str(i))
95     # print topMatches(prefs,'372')
96     # print getRecommendations(prefs,'372')
97     movies=transformPrefs(prefs)
98     print topMatches(movies, 'Crow, The (1994)')
99
```

```
[(1.0000000000000027, 'No Escape (1994)'), (1.0000000000000018,
'Denise Calls Up (1995)'), (1.0, 'What Happened Was... (1994)'), (1.
0, 'Underground (1995)'), (1.0, 'Two or Three Things I Know About
Her (1966)')]
[Finished in 0.3s]
```

Bottom 5 correlated movies to The Crow:

```
97 movies=transformPrefs(prefs)
98 print topMatches(movies, 'Crow, The (1994)')
99 # print topMatches(movies, 'Pocahontas (1995)')

[(-1.0000000000000004, 'Hate (Haine, La) (1995)'), (-1.0, '8 Seconds (1994)'), (-1.0, 'Calendar Girl (1993)'), (-1.0, 'Carried Away (1996)'), (-1.0, 'Contempt (M\xe9pris, Le) (1963)')]
[Finished in 0.3s]
```

Top 5 correlated movies to Pocahontas:

```
97 movies=transformPrefs(prefs)
98 # print topMatches(movies, 'Crow, The (1994)')
99 print topMatches(movies, 'Pocahontas (1995)')

[(1.0000000000000004, 'Night Falls on Manhattan (1997)'), (1.0000000000000004, 'Once Upon a Time in the West (1969)'), (1.0, 'Wings of the Dove, The (1997)'), (1.0, 'Widows' Peak (1994)'), (1.0, 'Walking Dead, The (1995)')]
[Finished in 0.2s]
```

Bottom 5 correlated movies to Pocahontas:

```
97 movies=transformPrefs(prefs)
98 # print topMatches(movies, 'Crow, The (1994)')
99 print topMatches(movies, 'Pocahontas (1995)')

[(-1.0, 'Angels and Insects (1995)'), (-1.0, 'Antonia's Line (1995)'), (-1.0, 'Awfully Big Adventure, An (1995)'), (-1.0, 'Carried Away (1996)'), (-1.0, 'City of Angels (1998)')]
[Finished in 0.2s]
```


I made a table separating the movies I should and should not like, according to these results.

First, I want to point out that La Haine (Hate) is in the “not recommended” list, but it actually happens to be one of my favorite films.

I haven't seen any of the others, but I watched the trailers and put the ones I would watch up near the top in bold. Both sides were equal, so I don't think the results were all that accurate.

RECOMMENDED	NOT RECOMMENDED
No Escape (1994)	Hate (Haine, La) (1995)
An Awfully Big Adventure (1995)	Contempt (M\xe9pris, Le) (1963)
What Happened Was... (1994)	Once Upon a Time in the West (1969)
Underground (1995)	The Walking Dead (1995)
Two or Three Things I Know About Her (1966)	8 Seconds (1994)
Angels and Insects (1995)	Night Falls on Manhattan (1997)
Antonia's Line (1995)	Calendar Girl (1993)
Denise Calls Up (1995)	The Wings of the Dove (1997)
Carried Away (1996)	Widows' Peak (1994)
City of Angels (1998)	Carried Away (1996)