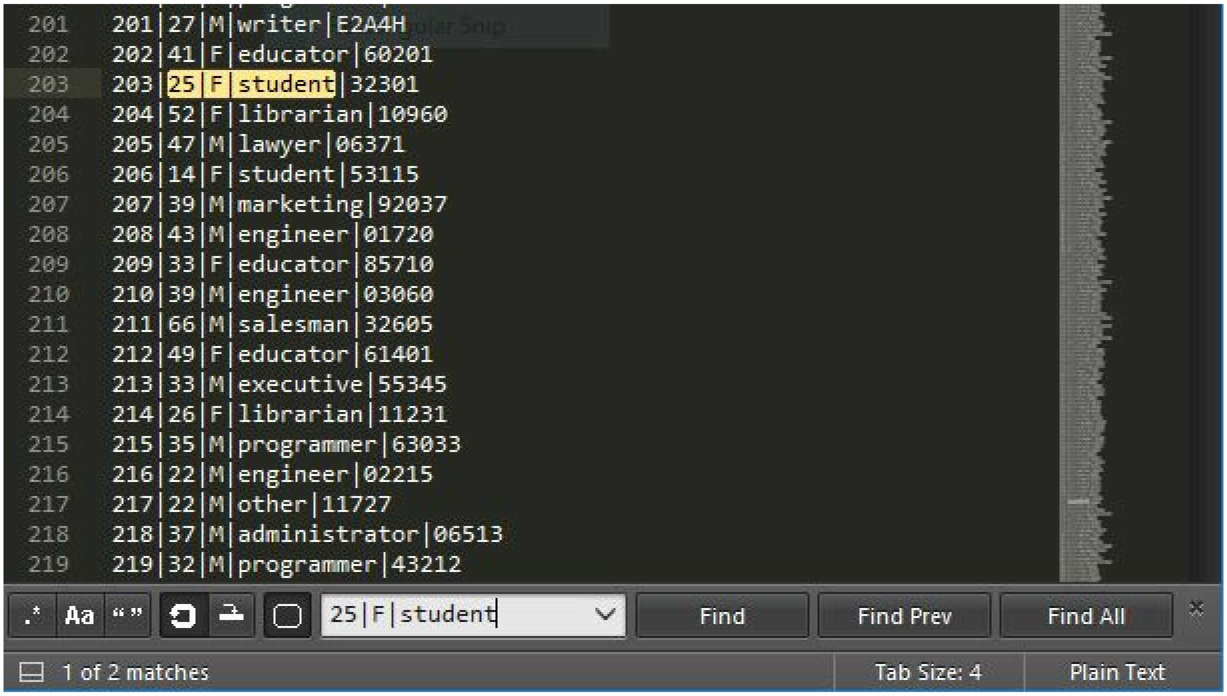
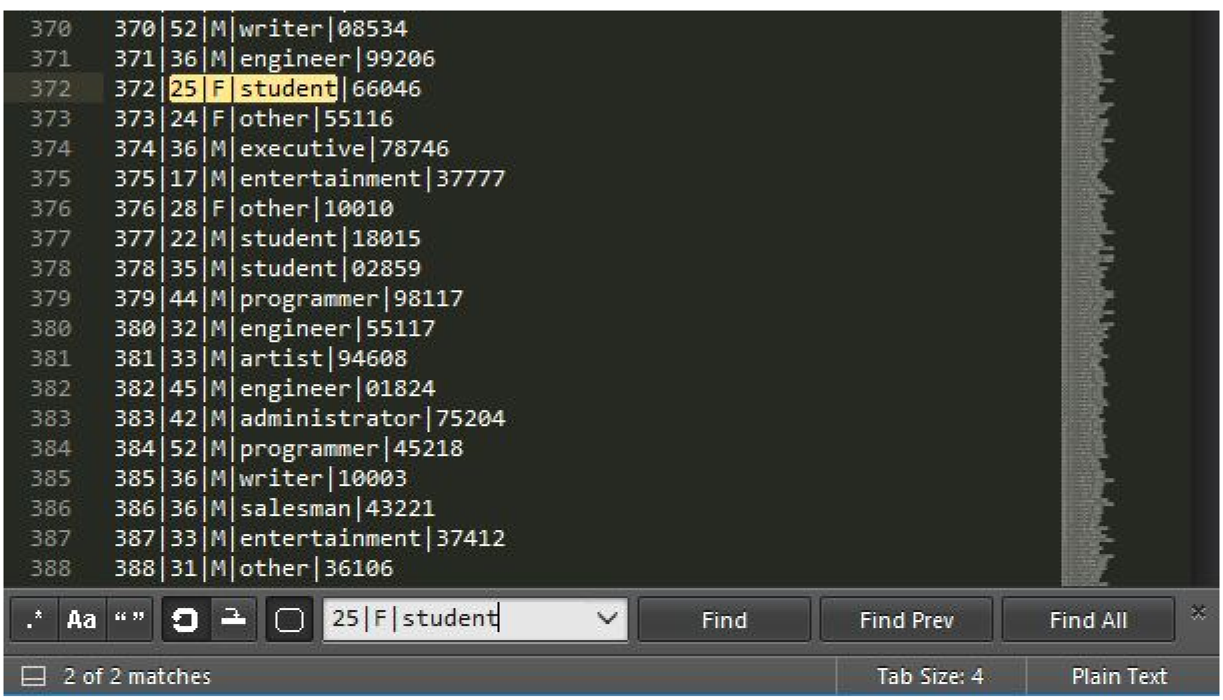
RACHEL MCCREARY
CS 432 ASSIGNMENT 7 REPORT

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

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

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

```
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
```

```
.* Aa “” 26|F|student  Find  Find Prev  Find All
1 of 2 matches    Tab Size: 4    Plain Text
```

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:

```
recommendations.py ×
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'])

```

```
{'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 and 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
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['372'])
```

```
{'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
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['241'])
```

```
{'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:

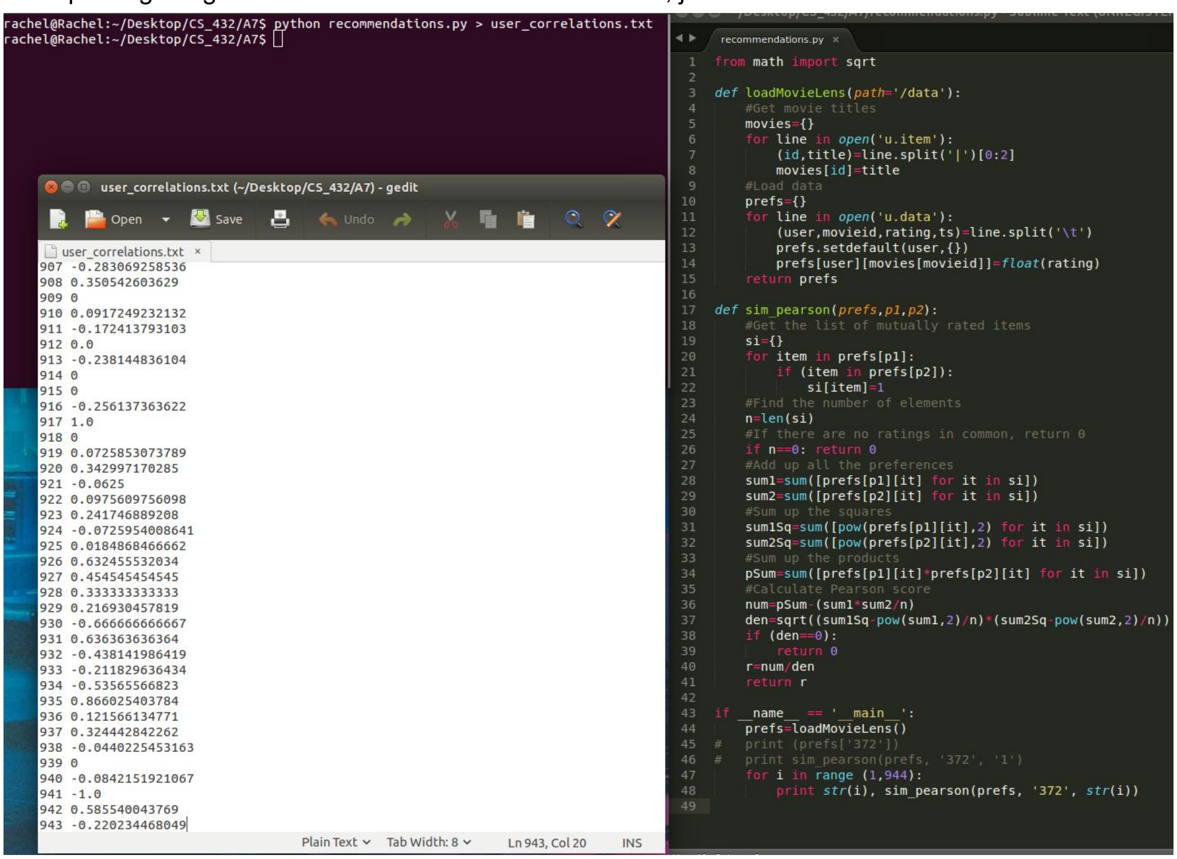
USER ID	TOP 3 FILMS	BOTTOM 3 FILMS
203	Grosse Pointe Blank (5) Scream (5) Star Wars (5)	A Time to Kill (1) Fargo (1) Mortal Kombat: Annihilation (2)
372	Psycho (5) The Shining (5) The Silence of the Lambs (5)	Evil Dead II (2) Aliens (3) Fargo (3)
241	Amistad (5) Scream (5) Soul Food (5)	Alien: Resurrection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



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 ×

```
def topMatches(prefs,person,n=5,similarity=sim_pearson):
    scores=[(similarity(prefs,person,other),other)
                    for other in prefs if other!=person]
    #Sort the list so the highest scores appear at the top
    scores.sort()
#   scores.sort(reverse=True)
    scores.reverse()
    return scores[0:n]

if __name__ == '__main__':
    prefs=loadMovieLens()
#   print (prefs['372'])
#   print sim_pearson(prefs, '372', '1')
#   for i in range (1,944):
#       print str(i), sim_pearson(prefs, '372', str(i))
    print topMatches(prefs,'372')
#   print getRecommendations(prefs,'372')

```

```
[(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 ×

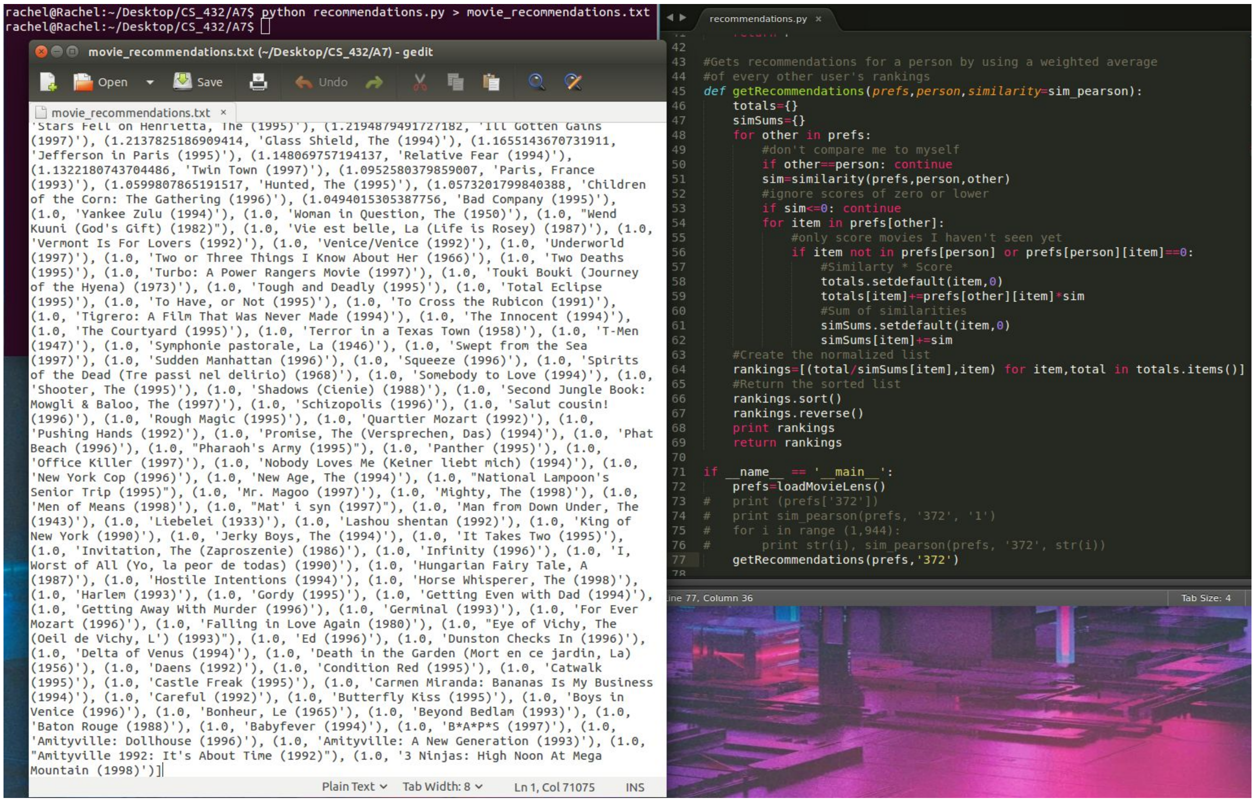
```
def topMatches(prefs,person,n=5,similarity=sim_pearson):
    scores=[(similarity(prefs,person,other),other)
                    for other in prefs if other!=person]
    #Sort the list so the highest scores appear at the top
#   scores.sort()
    scores.sort(reverse=True)
    scores.reverse()
    return scores[0:n]

if __name__ == '__main__':
    prefs=loadMovieLens()
#   print (prefs['372'])
#   print sim_pearson(prefs, '372', '1')
#   for i in range (1,944):
#       print str(i), sim_pearson(prefs, '372', str(i))
    print topMatches(prefs,'372')
#   print getRecommendations(prefs,'372')

```

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

PART 3

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):



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:

recommendations.py

```
def transformPrefs(prefs):
    result={}
    for person in prefs:
        for item in prefs[person]:
            result.setdefault(item, {})
            #Flip item and person
            result[item][person]=prefs[person][item]
    return result

if __name__ == '__main__':
    prefs=loadMovieLens()
#   print (prefs['372'])
#   print sim_pearson(prefs, '372', '1')
#   for i in range (1,944):
#       print str(i), sim_pearson(prefs, '372', str(i))
#   print topMatches(prefs,'372')
#   print getRecommendations(prefs,'372')
    movies=transformPrefs(prefs)
    print topMatches(movies, 'Crow, The (1994)')
```

```
[(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:

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

```
[(-1.000000000000004, '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:

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

```
[(1.000000000000004, '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:

```
    movies=transformPrefs(prefs)
#   print topMatches(movies, 'Crow, The (1994)')
    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)