In [ ]:

```
user_user_dict = {
    'Lisa Rose': {'Lady in the Water': 2.5,
                    'Snakes on a Plane': 3.5,
                    'Just my Luck': 3.0,
                    'Superman Returns': 3.5,
                    'You, Me and Dupree': 2.5,
                    'The Night Listener': 3.0},
    'Gene Seymour':{'Lady in the Water': 3.0,
                    'Snakes on a Plane': 3.5,
                    'Just my Luck': 1.5,
                    'Superman Returns': 5.0,
                    'The Night Listener': 3.0,
                    'You, Me and Dupree': 3.5}
,   'Michael Phillips':{'Lady in the Water': 2.5,
                        'Snakes on a Plane': 3.0,
                        'Superman Returns': 3.5,
                        'The Night Listener': 4.0},
    'Claudia Puig': {'Snakes on a Plane': 3.5,
                    'Just my Luck': 3.0,
                    'The Night Listener': 4.5,
                    'Superman Returns': 4.0,
                    'You, Me and Dupree': 2.5},
    'Mick LaSalle':{'Lady in the Water': 3.0,
                    'Snakes on a Plane': 4.0,
                    'Just my Luck': 2.0,
                    'Superman Returns': 3.0,
                    'The Night Listener': 3.0,
                    'You, Me and Dupree': 2.0},
    'Jack Matthews':{'Lady in the Water': 3.0,
                        'Snakes on a Plane': 4.0,
                        'Superman Returns': 5.0,
                        'The Night Listener': 3.0,
                        'You, Me and Dupree': 3.5},
    'Toby': {'Snakes on a Plane': 4.5,
            'Superman Returns': 4.0,
            'You, Me and Dupree': 1.0}
}
```

In [ ]:

```
#Suppose we build a recommender system following the user-user
# similarities approach with Pearson correlation as a similarity measure. What will
# be the rating prediction for user Michael Phillips, for movie “You, Me and
# Dupree”? Give the details of your computation.
# In computing the Pearson user-user similarities, restrict the user vectors to onl
# those components (movies) the two users have in common.

import math
def pearson_correlation(user1, user2):
    common_movies = {}
    for movie in user1:
        if movie in user2:
            common_movies[movie] = 1
    if len(common_movies) == 0:
        return 0
    n = len(common_movies)
    sum1 = sum([user1[movie] for movie in common_movies])
    sum2 = sum([user2[movie] for movie in common_movies])
    sum1Sq = sum([pow(user1[movie], 2) for movie in common_movies])
    sum2Sq = sum([pow(user2[movie], 2) for movie in common_movies])
    pSum = sum([user1[movie] * user2[movie] for movie in common_movies])
    num = pSum - (sum1 * sum2 / n)
    den = math.sqrt((sum1Sq - pow(sum1, 2) / n) * (sum2Sq - pow(sum2, 2) / n))
    if den == 0:
        return 0
    return num / den

def predict_rating(user, movie):
    total = 0
    simSums = 0
    for other in user_user_dict:
        if other == user:
            continue
        sim = pearson_correlation(user_user_dict[user], user_user_dict[other])
        if sim <= 0:
            continue
        if movie in user_user_dict[other]:
            total += user_user_dict[other][movie] * sim
            simSums += sim
    if simSums == 0:
        return 0
    return total / simSums

print(" Rating prediction = ", predict_rating('Michael Phillips', 'You, Me and Dupr
```

```
Rating prediction =  2.694636703980363
```

In [ ]:

```
#If we use the user-bias, item-bias approach to recommendation (Netflix
# competition), what will b_r (short for b_(lisa rose)) be after the first pass ove
# Set lambda_1=lambda_2=gamma=0.1, and start with zero bias values.

# mu = (Sum of all ratings) / (Total number of ratings)
def get_mu():
    total = 0
    count = 0
    for user in user_user_dict:
        for movie in user_user_dict[user]:
            total += user_user_dict[user][movie]
            count += 1
    return total / count
print("Mu = ", get_mu())

# br = br + gamma*(("lisa rose's movie rating"-(mu+br))- lambda*br)
# in the first pass, br = 0
def get_br(user):
    gamma = 0.1
    lambda_1 = 0.1
    mu = get_mu()
    br = 0
    for movie in user_user_dict[user]:
        br = br + gamma * ((user_user_dict[user][movie] - (mu + br)) - lambda_1 * b
    return br
```

```
print("br after first pass = ", get_br('Lisa Rose'))
```

```
Mu =  3.2285714285714286
br after first pass =  -0.10596754485699997
```