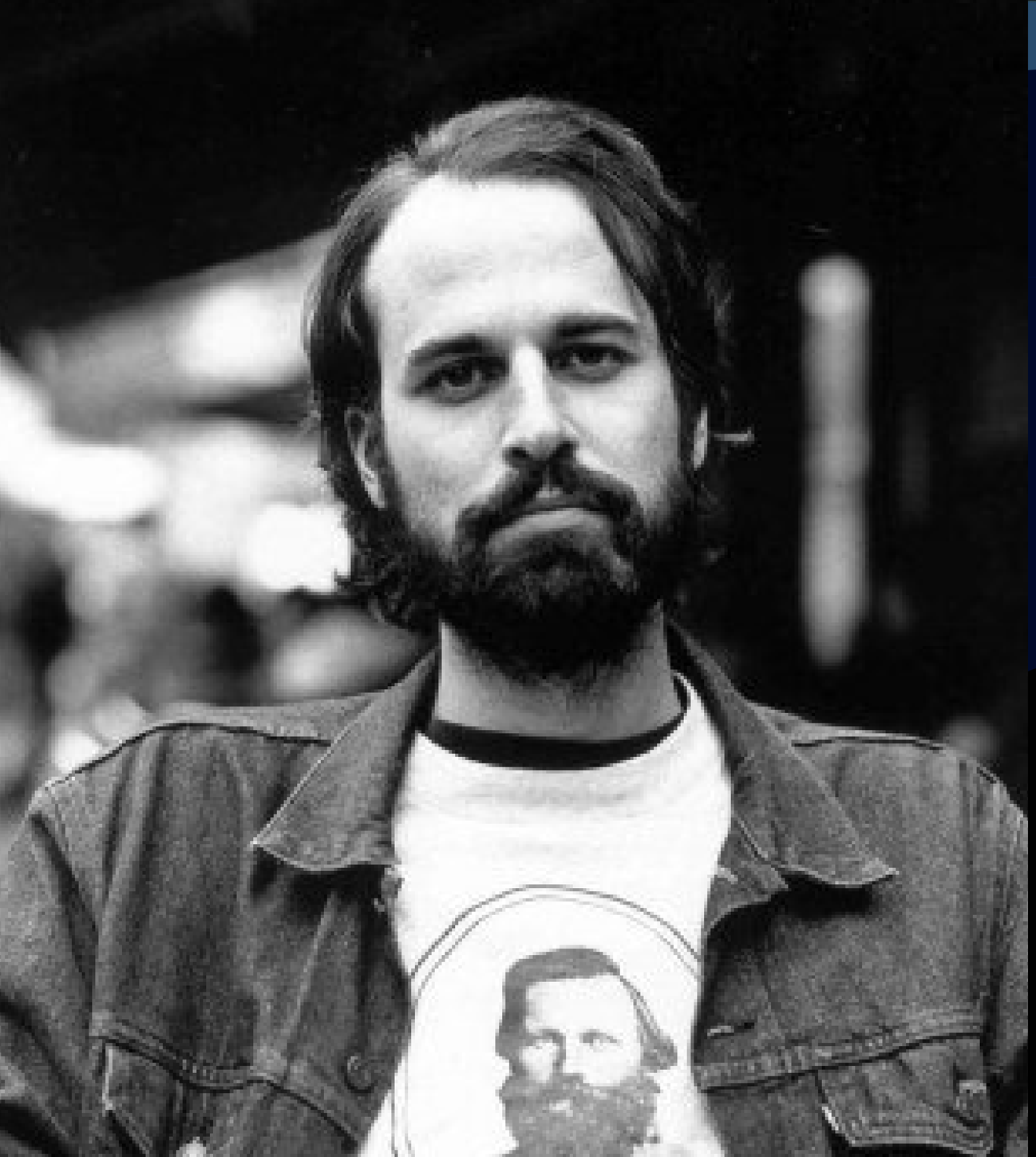
CIS 1100

Recommending

Python
Spring 2025
University of Pennsylvania



It’s too nice a day to read
a novel set in England.
We’re within inches of the perfect
distance from the sun,
the sky is blueberries and cream,
and the wind is as warm as air from a tire.

The Charm of 5:30
by David Berman

Announcements

1. TA Applications due tonight at 11:59pm
    i. no late days :)
2. HW09: **What to Watch?** Part 1 Released now, due April 23rd
    i. last opportunity to use late tokens if you want; NO LATE TOKENS ON PART 2
    ii. Part 2 comes out April 24th, due April 30th
    iii. These assignments cannot be dropped
3. Monday's check-in involves a *reading*, not a video
    i. Should still be short, ~10 minutes
    ii. Related to W2W?P2

TA Apps

- If you're interested, apply!
  - Application is short
  - Don't prematurely disqualify yourself

HW9: What to Watch

Part 1: Scraping ✅

Part 2: Recommending

Last sem, we gave two weeks to do all of it. Now, one week for each part.

- Previous findings: overall challenge "not that bad," but doing it all at once was miserable.
- Uses nearly everything from the course:
  - Loops & conditionals & complex control structures
  - Calling functions
  - Sets & lists & dictionaries
  - Objects
  - Scraping

HW09: The Big Idea

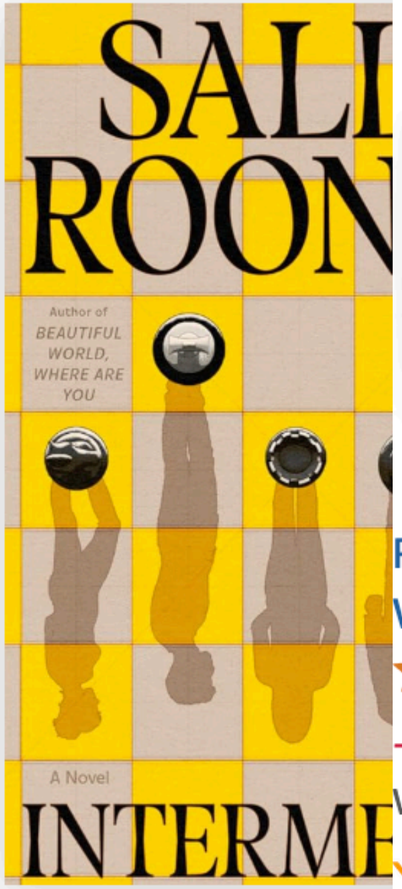
In traditional Computer Science, we talk about *algorithms...*

- finding the shortest path from A to B
- calculating the edit distance between two strings
- scheduling a shared resource to have minimal downtime

But in "real life," we talk about **"The Algorithm."**

rs *also* enjoyed

©
ová
6



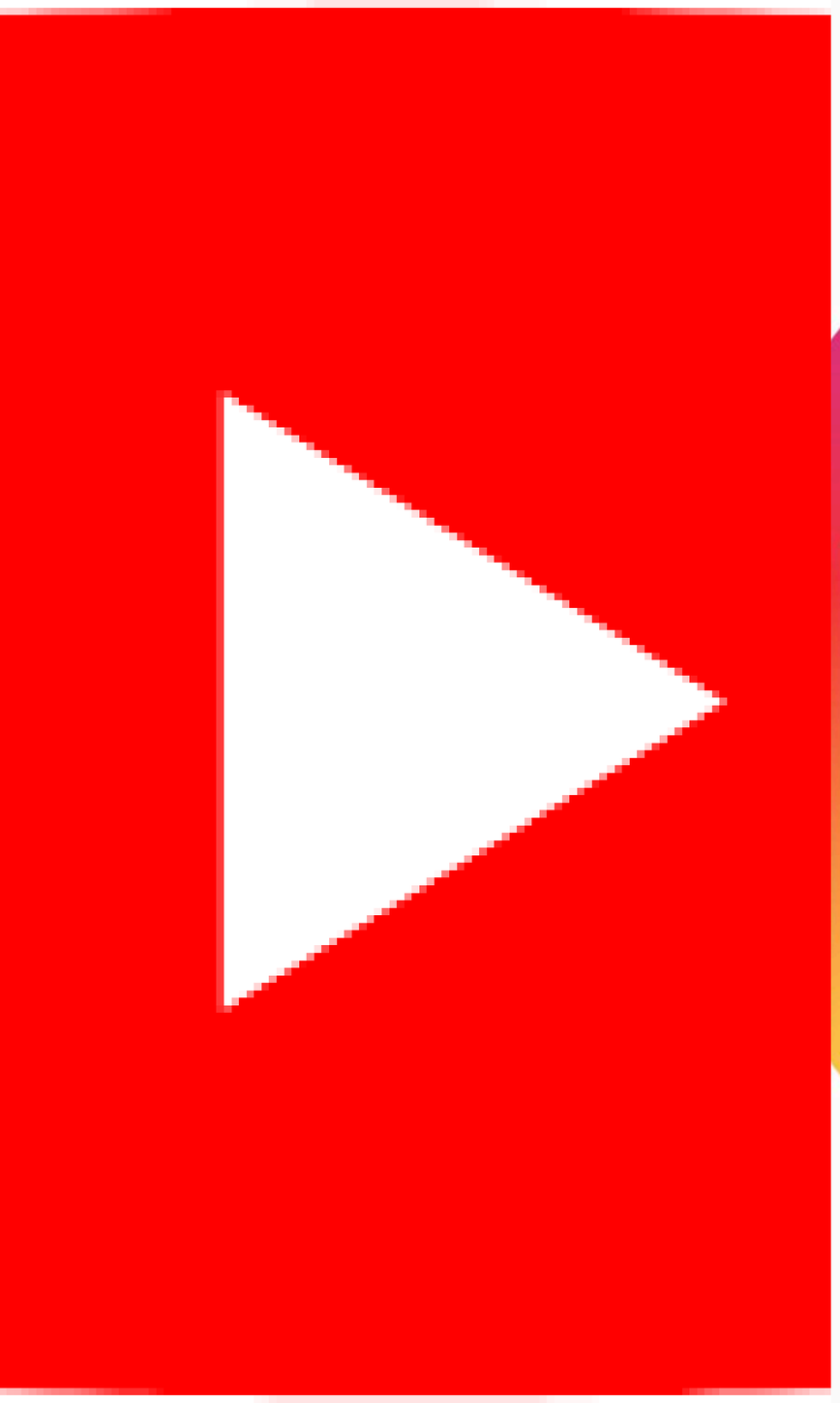
Intermezzo
Sally Rooney
★ 3.89 · 209k

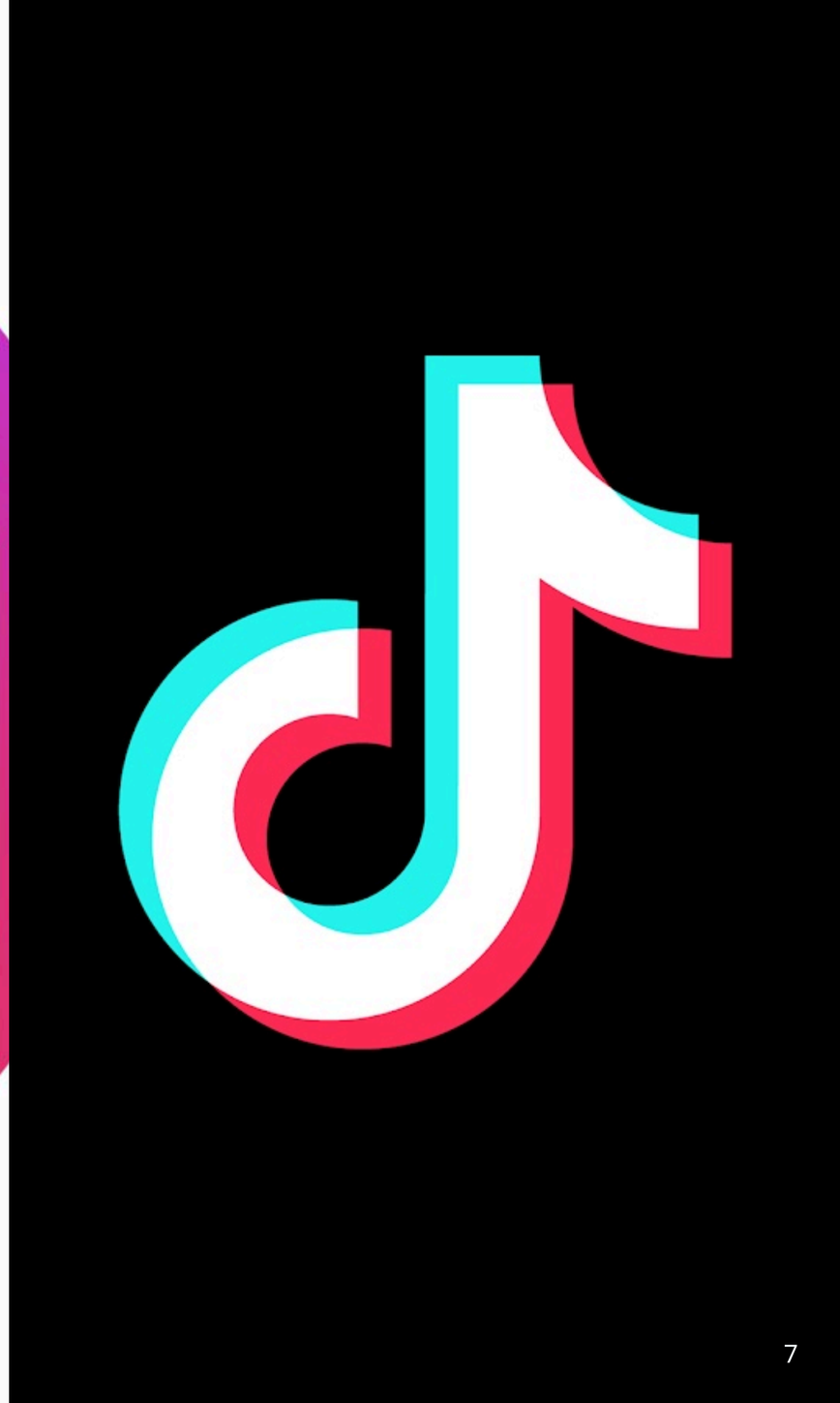
n Devices for you

Ring Video Doorbell
Wired with Ring Chime
4.5 out of 5 stars 1,032
-18% $69.99
Was: $84.98
prime FREE Delivery

Ring Battery Do
Now with 66% r
coverage, Head-
Video, Live View
4.5 out of 5 stars 10,8
#1 Best Seller in
Home Improvemen
-35% $109.98
List: $169.99
prime FREE Deliv
1 sustainability

cies of War: Vietnam





Assignment Goals

Write a program that allows you to find people's movie reviews on the internet. Then, after collecting them, use that information to make informed recommendations about what other people might like to watch!

1. Gives you something to do that feels substantial
   & uses a wide range of material from the course
2. Help you become better digital citizens, especially if this is your only CIS course!

The Scraping

Content from last few lectures tells you how.

Quick walkthrough of the sites you'll be scraping. **ATTENZIONE !**

- Information spread across multiple pages—figure out the URLs!
- Don't know ahead of time how many pages there are.
- The information you need is in tables, but needs some cleaning.

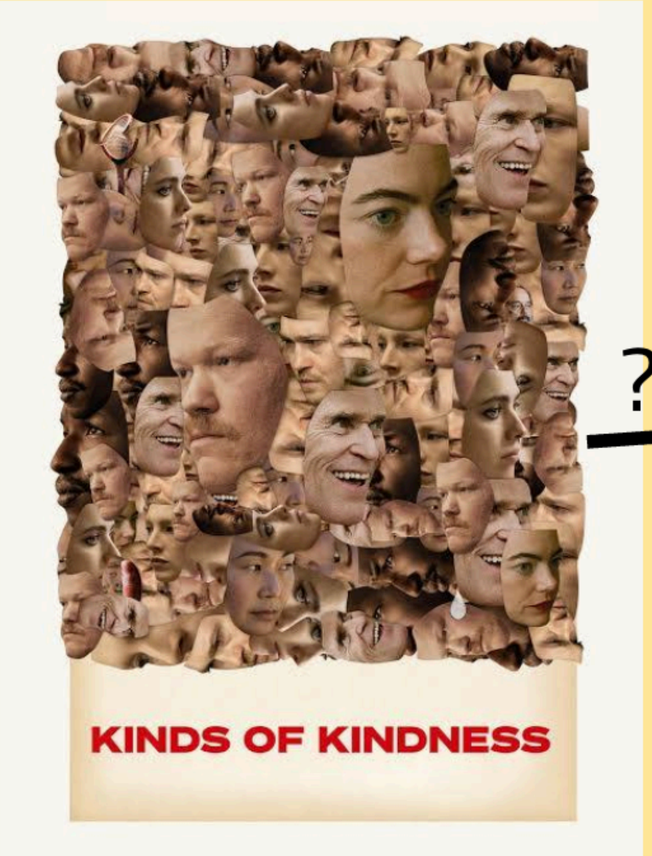
Recommending

After scraping, we have movie info and user ratings.

We want a way of making a recommendation for a user.

1. Turn each user's ratings into their **genre preferences** by taking the average score that they assign to movies of each genre.
2. Determine a way of comparing one user's preferences to another. (We'll use **cosine similarity.**)
3. Compare the preferences of the user seeking a recommendation to all other users' preferences in order to find the most similar other user.
4. Suggest movies that the most similar other user likes a lot.

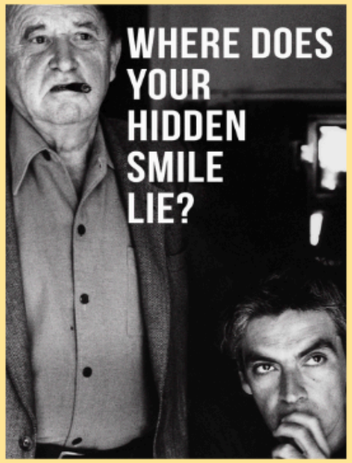
First Idea: Item-Based Filtering

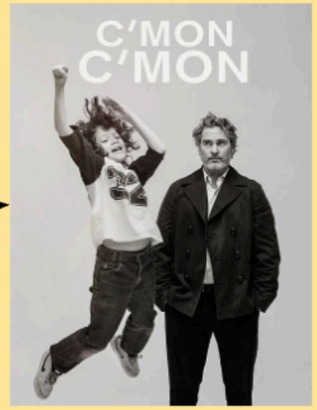


?

?

?





If I like movie **X**, can you find me some other movie **Y** that's similar?

Activity

In **(L11),** write down a favorite movie that you like. Then, try to come up with three essential aspects of what you like about it. *(2 minutes, silently)*

Activity

Turn to a partner and discuss for *3 minutes:*

- First, share the properties that you describe your movie **without sharing the name of the movie.**
- Decide if that sounds interesting to you or not.
- Reveal the name of the movie
  - If you've heard of it or seen it, did their description seem very useful?

Item-Based Filtering

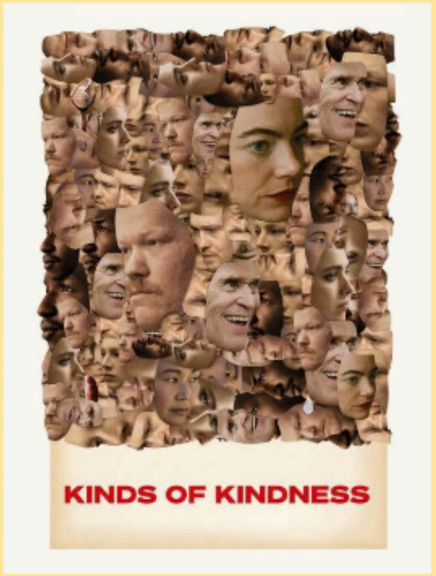
Item-Based Filtering requires that we have a way of modelling the "essential properties" of movies in order to find the most similar other movies.

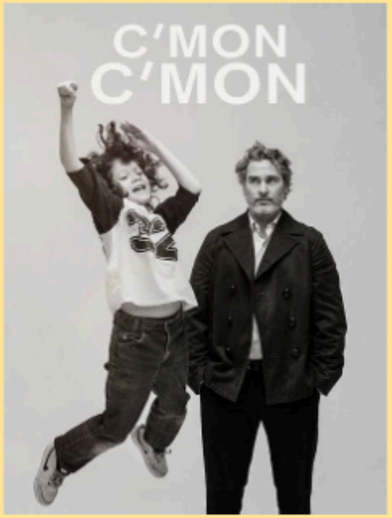
Not really easy to do in practice!

User-Based Recommendations

Easier to make recommendations based on *what other people like* rather than some *essential properties about what you like.*

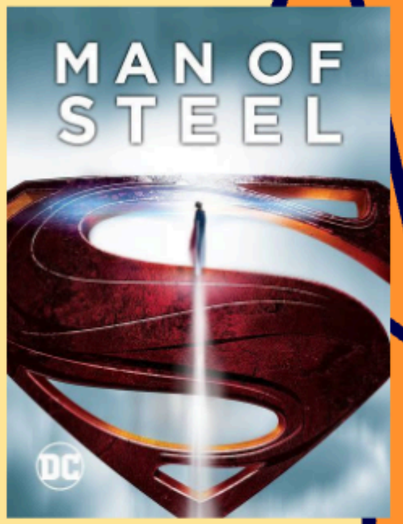
😍 😍 🤢 😍 🤢


KINDS OF KINDNESS


C'MON
C'MON


THE
BATMAN


DECISION TO LEAVE


MAN OF
STEEL

😍 😍 🤢 😍 ?

User-Based Recommendations

User-Based Filtering allows us to make recommendations based on what similar users liked.

Can be a bit easier since we don't have to worry as much about what makes each movie "itself".

Modeling User Reviews as Preferences

Genres
Action, Adventure, Comedy, Fantasy

Genre can be a useful proxy for more detailed properties of a movie.

We'll model a user's overall preferences by calculating the average scores they assign to movies tagged with a particular genre.

Activity: Ratings to Preferences

```
movie_info = {
    1: ("Harry's Adventure", ("Comedy", "Adventure")),
    2: ("Travis' Tragedy", ("Drama", "IMAX", "Comedy")),
}

# Maps movie IDs to Sadia's ratings of those movies.
sadias_ratings = {1: 3, 2: 4}
```

No need to write code to do these, just mental decoding and arithmetic.

(S7) What rating does Sadia give to *Travis' Tragedy*?

(S8) What is the average rating that Sadia awards to **thriller** movies?

(S9) What is the average rating that Sadia awards to **comedy** movies?

(S10) What is the average rating that Sadia awards to **drama** movies?

Movie Recommender

A `MovieRecommender` stores an attribute called `self.movie_info`. It will look something like this.

```
{1210: ('Star Wars: Episode VI - Return of the Jedi',
        ('Action', 'Adventure', 'Sci-Fi')),
 2028: ('Saving Private Ryan', ('Action', 'Drama', 'War')),
 1307: ('When Harry Met Sally...', ('Comedy', 'Romance')),
 5418: ('Bourne Identity, The', ('Action', 'Mystery', 'Thriller')),
 56367: ('Juno', ('Comedy', 'Drama', 'Romance')),
 3751: ('Chicken Run', ('Animation', 'Children', 'Comedy'))}
```

(L13) If `self.movie_info` stores a dictionary with this shape, write an expression that can look up the *title* of a movie with ID `3943`.

(C12) Finish this function, which prints out each genre associated with the input `movie_id`

```
def print_all_genres(self, movie_id: int):
```

Movie Recommender

A `MovieRecommender` stores an attribute called `self.all_user_ratings`.

It will look something like this. (Actually much longer.)

```
{514: {2716: 5.0, 780: 2.0},
 279: {780: 4.0, 300: 2.5, 1010: 0.5}}
```

(L13) What do the "outer" keys (514, 279) represent? What do the "inner" keys (2716 or 300) represent? What do the `float` values (5.0, 4.0) represent?