CompSci 101
Fall 2021

Lecture 19

Reminders

- Assignments
  - Assign 5 due
  - Assign 6 live

- Spring 2022 UTA applications open
  - See Ed announcement

Key instructions

- Input ✔
- Output ✔
- Assignments* ✔
- Math/Logic ✔
- Conditionals ✔
- Repetition ✔

**not listed in book*

Python Data Types

- int, float, bool ✔
- Collections
  - Strings ✔
  - Lists ✔
  - Tuples ✔
  - Sets ✔
  - Dictionaries ✔

PFTD

- **Exceptions**
- **Recommender**
  - Recommendations big picture
  - Assignment big picture
  - Simple recommendation example
  - Actual recommendation assignment

KISS Principle

- Think of the non-computing context for any word/terms
- KISS model
  - Work smarter, not harder!!
- “Good programmers are simply good designers.”
  - *-Dr. Washington*
- Design first and always!
- Importance of reusability
- *USE PyCharm/PythonTutor IF YOU HAVE QUESTIONS!*

People to Know: Frieda McAlear

- BS (Vesalius College-Brussels)
- Master of Research in Geograph (Queen Mary University of London)
- Senior Research Associate
  - Kapor Center
- Examines: 1) the barriers facing youth of color in STEM, (2) their coping strategies, and (3) programmatic interventions and resources to reduce barriers to STEM attainment.
- Co-founder-M4SJ (Mapping for Social Justice)
- Native Alaskan (Inupiaq)

Python exceptions

- What should you do if you prompt user for a number and they enter "one"
  - Test to see if it has digits?

- Exceptions make your program *robust*.
- Use exceptions with **`try:`** and **`except:`**

General syntax

```
try:
    # code block that may cause the error
except errorName:
    # code that should happen if error occurs
```

Handling Exceptions

- What happens: **`x = int("123abc")`**

```python
d=["This is a test", 12, "string", "Blue Devils."]

st=input("Choose 1:")
val=int(st)
if 0<=val and val<len(d):
    print(d[val])
```

- PyCharm example

Recommendation Systems: Yelp

- Are all users created equal?
- Weighting reviews

- What is a recommendation?

Recommender Systems: Amazon

- How does Amazon create recommendations?

Recommendation Systems: Netflix

- **Netflix offered a prize in 2009**
  - Beat their system? Win $1M
  - http://nyti.ms/sPvR

NETFLIX

Compsci 101 Recommender

- Doesn't work at the scale of these systems, uses publicly accessible data, but ...
  - Movie data, food data, book data

- Make recommendations
  - Based on ratings, how many stars there are
  - Based on weighting ratings by users like you!

- Collaborative Filtering: math, stats, compsci

Machine learning!

Simple Example

Tandoor	Fornio	McDon	Loop	Panda	Twin
0	3	5	0	-3	5
1	1	0	3	0	-3
-3	3	3	5	1	-1

- Rate restaurants on a scale of -5 to 5
- What restaurant should I choose to go to?
  - How do I decide?
- What do the ratings say? Let's take the average!

Calculating Averages

Tandoor	Fornio	McDon	Loop	Panda	Twin
0	3	5	0	-3	5
1	1	0	3	0	-3
-3	3	3	5	1	-1

- What is the average rating of each restaurant?
- Tandoor: (1 + -3)/2 = -1.00
  - Don't count rating if not rated
- Il Fornio: (3 + 1 + 3)/3 = 2.33
- Where to eat? Who has the highest average?
  - McDonalds and The Loop (8/2=4.00)

Python Specification

- Items: list of strings (header in table shown)

```
items = ["DivinityCafe", "FarmStead", "IlForno", "LoopPizzaGrill",
         "McDonalds", "PandaExpress", "Tandoor", "TheCommons",
         "TheSkillet"]
```

- Dictionaries: Key-value pairs are name: ratings (string: int list)
- len(ratings[i]) == len(items)

```
ratings = \
        {"Sarah Lee"  : [3, 3, 3, 3,  0,-3, 5, 0,-3],
         "Melanie"    : [5, 0, 3, 0,  1, 3, 3, 3, 1],
          "J J"       : [0, 1, 0,-1,  1, 1, 3, 0, 1],
          "Sly one"   : [5, 0, 1, 3,  0, 0, 3, 3, 3],
          "Sung-Hoon" : [0,-1,-1, 5,  1, 3,-3, 1,-3],
          "Nana Grace": [5, 0, 3,-5, -1, 0, 1, 3, 0],
          "Harry"     : [5, 3, 0,-1, -3,-5, 0, 5, 1],
          "Wei"       : [1, 1, 0, 3, -1, 0, 5, 3, 0]
        }
```

Python Specification

- Items: list of strings (header in table shown)

```
items = ["DivinityCafe", "FarmStead", "IlForno", "LoopPizzaGrill",
         "McDonalds", "PandaExpress", "Tandoor", "TheCommons",
         "TheSkillet"]
```

- Values in dictionary are ratings: int list
  - **`len(ratings[i]) == len(items)`**

index = 4

index = 4

```
ratings = \
        {"Sarah Lee"  : [3, 3, 3, 3,  0,-3, 5, 0,-3],
         "Melanie"    : [5, 0, 3, 0,  1, 3, 3, 3, 1],
          "J J"       : [0, 1, 0,-1,  1, 1, 3, 0, 1],
          "Sly one"   : [5, 0, 1, 3,  0, 0, 3, 3, 3],
          "Sung-Hoon" : [0,-1,-1, 5,  1, 3,-3, 1,-3],
          "Nana Grace": [5, 0, 3,-5, -1, 0, 1, 3, 0],
          "Harry"     : [5, 3, 0,-1, -3,-5, 0, 5, 1],
          "Wei"       : [1, 1, 0, 3, -1, 0, 5, 3, 0]
        }
```

Recommender averages

- **`def averages(items,ratings):`**

- *Input: items -- list of restaurants/strings*
- *Input: ratings -- dictionary of name to ratings*
  - key: string, “Melanie”
  - value: list of ints, **`[1,0,-1, ... 1`**]
    - parallel list to list of restaurants (`items`)
      - k^{th} rating maps to k^{th} restaurant
- *Output: recommendations*
  - List of tuples (name, avg rating) or (str, float)
    - Sort by rating from high to low

Activity 1:
https://bit.ly/101f21-11-16-1

Drawbacks of Averaging

- Are all user’s ratings the same to me?
  - Weight/value ratings of people most similar to me

- Collaborative Filtering
  - https://en.wikipedia.org/wiki/Collaborative_filtering
  - How do we determine who is similar to/"near” me?

- Mathematically: treat ratings as vectors in an N-dimensional space, N = # of items that are rated
  - a.k.a. weight has higher value → closer to me

Determining "closeness"

- Calculate a number measuring closeness to me (higher number→closer)
  - I'm also a rater, "me" is parameter to function

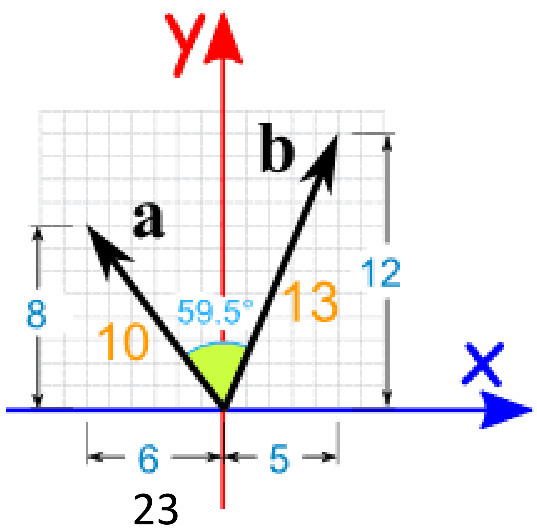
Same as before, dictionary of rater to ratings

- Function:
  - **`similarities("rodger", ratings)`**

- Return **`[("rater1", #), ("rater2", #), ...]`**
  - List of tuples based on closeness to me
  - sorted high-to-low by similarity

What's close? Dot Product

- https://en.wikipedia.org/wiki/Dot_product
  - For [3,4,2] and [2,1,7]
    - **3*2 + 4*1 + 2*7 = 6+4+14 = 24**
- How close am I to each rater?
- What happens if the ratings are
  - Same sign? Me: 3, -2 Other: 2, -5
  - Different signs? Me: -4 Other: 5
  - One is zero? Me: 0 Other: 4
- What does it mean when # is…
  - Big? Small? Negative?



Writing similarities

- Given dictionary, return list of tuples

```
def similarities(name, ratings):
    return [('name0', #), ...('nameN', #)]
```

- What is the # here?
  - Dot product of two lists
  - One list is fixed (name)
  - Other list varies (loop)

- Think: How many tuples are returned?

food.json

```
{"Sarah Lee":
    [3, 3, 3, 3, 0, -3, 5, 0, -3],
  "Melanie":
    [5, 0, 3, 0, 1, 3, 3, 3, 1],
   "J J":
    [0, 1, 0, -1, 1, 1, 3, 0, 1],
   "Sly one":
    [5, 0, 1, 3, 0, 0, 3, 3, 3],
   "Sung-Hoon":
    [0, -1, -1, 5, 1, 3, -3, 1, -3],
   "Nana Grace":
    [5, 0, 3, -5, -1, 0, 1, 3, 0],
   "Harry":
    [5, 3, 0, -1, -3, -5, 0, 5, 1],
   "Wei":
    [1, 1, 0, 3, -1, 0, 5, 3, 0]
}
```

Collaborative Filtering

- Once we know raters "near" me? Weight them!
  - How many raters to consider? 1? 10?
  - Suppose Fran is **`[2, 4, 0, 1, 3, 2]`**
- What is Sam’s similarity to Fran?

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Collaborative Filtering

- Once we know raters "near" me? Weight them!
  - How many raters to consider? 1? 10?
  - Suppose Fran is `[2, 4, 0, 1, 3, 2]`
- What is Sam’s similarity to Fran?
  - 2*0 + 4*3 + 0*5 + 1*0 + 3*-3 + 2*5 = **13**
  - **Sam’s ratings** `[0, 3, 5, 0, -3, 5] * 13`
  - **Sam’s weighted:** `[0, 39, 65, 0, -39, 65]`

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Think: What is Chris’s similarity?

What is Chris's similarity and weights?

- Suppose Fran is `[2, 4, 0, 1, 3, 2]`
- Chris's similarity is:

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

What is Chris's similarity and weights?

- Suppose Fran is **`[2, 4, 0, 1, 3, 2]`**
- Chris's similarity is:
  - 2*1 + 4*1 + 0*0 + 1*3 + 3*0 + 2*(-3) = 3
- Chris' weighted ratings:
  - 3 * [1, 1, 0, 3, 0, -3]
  - **`[3, 3, 0, 9, 0, -9]`**

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1

Steps for Recommendations

- Start with you, a rater/user and all the ratings
  - Get similarity "weights" for users: dot product
- Calculate new weighted ratings for all users
  - `[weight * r for r in ratings]`
- Based on these new ratings, find average
  - Using weighted & original average function
  - Don't use zero-ratings
- Check recommendations by … (not required)
  - Things I like are recommended? If so, look at things I haven't tried!

Recommendations

- Get new weighted averages for each eatery
  - Then find the best eatery I've never been to

```
def recommendations(name,items,ratings,numUsers)
   return [('eatery0', #), ...('eateryN', #)]
```

Fran gets
a recommendation
(considering numUsers raters)

Chris:3
Nat:12
Sam:13

```
rc = recommendations("Fran",items,ratings,3)
#use this to provide evals to Fran
```

Similarities Summarized

- How do we get weighted ratings?

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1
Fran	2	4	0	1	3	2

```
def similarities(name,ratings):
   return [('name', #), ...('name', #)]

weights = similarities("Fran", ratings)
```

Making Recommendations

- How do we get weighted ratings? Call average?

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	3	5	0	-3	5
Chris	1	1	0	3	0	-3
Nat	-3	3	3	5	1	-1
Fran	2	4	0	1	3	2

```
weights = similarities("Fran", ratings)
weights = #slice based on numUsers
weightedRatings = {}. # new dictionary
for person, weight in weights:
    weightedRatings[?] = ?
```

Calculating Weighted Average

	Tandoor	IlForno	McDon	Loop	Panda	Twin
Sam	0	39	65	0	-39	65
~~Chris~~	~~3~~	~~3~~	~~0~~	~~9~~	~~0~~	~~-9~~
Nat	-36	36	36	60	12	-12
Total	-36	75	101	60	-27	53
Avg	-36	37.5	50.5	60	-13.5	26.5

```
recommendations("Fran",items,ratings,2)
```

- Make recommendation for Fran? Best? Worst?
- Fran should eat at Loop! Even though only using Nat's rating
  - No recommendation from Sam, so only 1 recommendation for Loop
- But? Fran has been to Loop! Gave it a 1, … McDonalds!!!! ??

Activity 2: https://bit.ly/101f21-11-16-2

Assignment Modules

Implement functions in this order

RecommenderEngine

1. averages(…)
2. similaries(…)
3. recommendations(…)

RecommenderMaker

1. makerecs(…)

TestRecommender

Can be implemented before Recommender stuff or after

MovieReader

1. getdata(…)

BookReader

1. getdata(…)

RecommenderEngine before RecommenderMaker and use TestRecommender

Function Call Ordering

- **Some_Reader_Module.getdata(…)**
- **RecommenderMaker.makerecs(…)**
  - RecommenderEngine.recommendations(…)
    - RecommenderEngine.similarities(…)
    - RecommenderEngine.averages(…)

Start with inner most call and work outwards

Test on your computer and on Gradescope as you go!

Reminders

- Work smarter, not harder
- Design first
- Get smaller parts working, then build on it
- Try to identify where you are stuck
  - Identify resources to help solve problem
- Leverage your design and PythonTutor to understand program flow of control
  - http://pythontutor.com