CMPT 120 LECTURE 5-1

Prof. Nick Vincent

Simon Fraser University

2023-10-04

QUICK NOTE FOR THOSE READING SLIDES

There’s only one set of slide for week 5 (Monday was a holiday).

5-1 HOUSEKEEPING

- Updates on marking
- Updates on additional study materials (and how to get feedback)
- Office hours today

5-1 AGENDA

- Quick set of review questions
- Quick review on accumulator pattern
- Completing our recommender system example
  - Putting together for loops, nested for loops, list slicing, string slicing

REVIEW QUESTIONS

QUESTION 1A

What does this snippet output?

```
response = "I LOVE RAIN!!"
words = response.lower().strip("!").split(" ")
if "rain" in words or "umbrella" in words:
    print("You're a fan of the rain?")
```

QUESTION 1A

You're a fan of the rain?

QUESTION 1B

```
response = "I LOVE RAIN!!"
try:
    words = response.lower().split(" ").strip("!")
    if "rain" in words or "umbrella" in words:
        print("You're a fan of the rain?")
except AttributeError:
    print("Something went wrong")
```

QUESTION 1B

```
Something went wrong
```

QUESTION 2A

What does this output?

```
1 fruits = "durian, rambutan, lychee"
2 fruit_list = fruits.split(",")
3 print("durian" in fruit_list)
4 print("rambutan" in fruit_list)
```

QUESTION 2A

True
False

QUESTION 2B

What does this output?

```
1  fruits = "durian,rambutan,lychee".split(",")
2  print(fruits[1][0].upper())
```

QUESTION 2B

R

QUESTION 3

Find three errors with this code:

```
1  z = int(input["Give me a number, any number: "])
2  if z > 5 and <= 10:
3      print(x)
```

QUESTION 3 – ANSWER

#| echo: true

```
z = int(input("Give me a number, any number: "))
if z > 5 and z <= 10:
    print(z)
```

QUESTION 4

What does this snippet output?

```
6
['b', 'c']
['a', 'b', 'c', 'd']
['d', 'e', 'f']
['a', 'b', 'c', 'd', 'e', 'f']
f
['d', 'e']
```

TRICK TO REMEMBER WITH SLICING

`letters[:4]` will give you 4 total items in your output!

ACCESSING CHARACTERS IN STRINGS

- Just like accessing items in a list
- If you think of strings as a list of characters, you're good to go
- (As long as you know how lists work)

OK, BACK TO CONTENT

- We want to extend our recommender systems example

LET'S RECAP...

```
with open("example.csv", "r") as file:
    # Skip header
    header = file.readline()

    # Process and print each record
    for line in file:
        columns = line.strip().split(",")
        nice_output = ""
        for column in columns:
          nice_output += column + " | "
        print(nice_output)
```

GENERATING DATA (ADVANCED)

```python
import csv
import random

# Lists of names and diets
names = ["Michael Jordan", "Tom Hanks", "Arya Stark", "Goku", "Scarlett Joh
         "Mickey Mouse", "Hermione Granger", "Naruto", "Serena Williams", "
         "Leonardo DiCaprio", "Mario", "Taylor Swift", "Darth Vader", "Jame
         "Usain Bolt", "Tony Stark", "Bugs Bunny", "The Rock", "Walter Whit
         "Sheldon Cooper", "Mulan", "Elsa", "Sherlock Holmes", "Jack Sparro
         "Luffy", "Luke Skywalker", "Marilyn Monroe", "Freddie Mercury", "L
         "Superman", "Wonder Woman", "Neo", "Beyonce", "Ash Ketchum",
         "Lara Croft", "Elvis Presley", "Kermit the Frog", "Spider-Man", "H
         "Indiana Jones", "Simba", "Rihanna", "Dexter Morgan", "Morpheus",
         "Wolverine", "Cinderella", "Captain America", "Thor", "Katniss Eve

movies_by_genre = {
    "Drama": ["The Shawshank Redemption", "Forrest Gump", "The Godfather",
    "Action": ["The Dark Knight", "Avengers: Endgame", "Pulp Fiction", "Jur
```

```
name, favorite_movie, second_favorite_movie, preferred_political_party,
ideal_diet
Sherlock Holmes, Finding Nemo, Finding Nemo, right wing, balanced
Naruto, Forrest Gump, Forrest Gump, very right wing, plant-heavy
```

GET YOUR OWN FAKE_DATA.CSV

- Download csv file from Canvas -> GitHub
- Download python file from Canvas -> GitHub and run it
- Bonus question: what will be different?

CHECK THAT YOU CAN READ IT

```
1 with open("fake_data.csv", "r") as file:
2     header = file.readline()
3     print(header)
4     line = file.readline()
5     print(line)
```

```
name,favorite_movie,second_favorite_movie,preferred_political_party,ideal_diet

Sherlock Holmes,Finding Nemo,Finding Nemo,right wing,balanced
```

SIMILARITY

Our initial definition definition: “common interests counter”

- If we have the same favorite movie, that’s one common interest
- If we have the same preferred political party, that’s one common interest.
- If we also have the same preferred diet, that’s one common interest.
- Let’s see who is the most similar!

MANY DEFINITIONS OF SIMILAR

In CS and math, there are many ways we can see how "similar" or "close" two items are

- (cosine distance, manhattan distance, etc.)
- So we need to specify a definition and algorithm

OUR MOVIE/POLITICAL PARTY/DIET RECOMMENDER SYSTEM

To recommend a movie to someone, we take the following approach:

- We'll look at favorite movie, preferred political party, and ideal diet
- Each match counts as one similarity score. So score between any two people ranges from 0 to 3.
- When we find the "highest" score, suggest our user watch that person's second favorite movie!

OUR ALGORITHM

INITALIZE

- Set variables to hold all the things we need (i.e., get our buckets ready)

LOOP

- Run through every record and process it

INITIALIZE

- Pick the user we'll give a recommendation to
- Define a variable to hold similarity score (starts at zero)
- Define a variable to keep track of which person is most similar
- Define a variable to keep track of what what we'll recommend

LOOP

FOR EACH PERSON OTHER THAN OUR CHOSEN ONE

OUR PSEUDO CODE

```
# select a user (who will receive recommendation)
# init variables to hold top similarity score, top person, recommendation

# load data
    # use a loop to go through all other record lines (one record = one use
    # make sure to skip chosen user
    # split each record line into a list of items
        # use a nested loop to go through each item
        # make sure we compare columns correctly
    # check if similarity score is higher
        # update if so

# print top score, record, and recommendation
```

INITIALIZE

```
# select a user + init variables to keep track of scores
index_of_user = 17
top_score = 0
top_record = ""
recommendation = ""

# load data
    # use a loop to go through all other record lines (one record = one use
    # make sure to skip chosen user
    # split each record line into a list of items
        # use a nested loop to go through each item
        # make sure we compare columns correctly
    # check if similarity score is higher
        # update if so

# print top score, record, and recommendation
```

LOAD DATA

```
# select a user + init variables to keep track of scores
index_of_user = 17
top_score = 0
top_record = ""
recommendation = ""

# load data


# load data
with open("fake_data.csv", "r") as file:
    # Skip header
    header = file.readline()
    print(header)
    # use a loop to go through all other record lines (one record = one use
    # make sure to skip chosen user
    # split each record line into a list of items
        # use a nested loop to go through each item
```

```
name,favorite_movie,second_favorite_movie,preferred_political_party,ideal_diet
```

FIRST FOR LOOP

```
# select a user + init variables to keep track of scores
index_of_user = 17
top_score = 0
top_record = ""
recommendation = ""

# load data
with open("fake_data.csv", "r") as file:
    # Skip header
    all_lines = file.readlines()
    header = all_lines[0]
    records = all_lines[1:]

    user_record = records[index_of_user]

    # use a loop to go through all other record lines (one record = one use
    for record in records:
        if line == user_record:
```

```
['Sherlock Holmes', 'Finding Nemo', 'Finding Nemo', 'right wing', 'balanced']
['Naruto', 'Forrest Gump', 'Forrest Gump', 'very right wing', 'plant-heavy']
['Beyonce', 'Lord of the Rings: The Fellowship of the Ring', 'Jurassic Park',
'very right wing', 'balanced']
```

```
['Luke Skywalker', 'Jurassic Park', 'The Dark Knight', 'very right wing',
'meat-heavy']
['Scarlett Johansson', 'Toy Story', 'Frozen', 'right wing', 'meat-heavy']
['Captain America', 'Titanic', 'The Godfather', 'left wing', 'fruit-heavy']
['Thor', 'The Matrix', 'Inception', 'very left wing', 'dairy-heavy']
['The Rock', 'Pulp Fiction', 'Avengers: Endgame', 'left wing', 'dairy-heavy']
['Beyonce', 'The Lion King', 'Shrek', 'very right wing', 'meat-heavy']
['Captain America', 'Jaws', 'Casablanca', 'very right wing', 'plant-heavy']
['Elsa', 'Blade Runner', 'Back to the Future', 'very left wing', 'fruit-
heavy']
['Cinderella', 'Jaws', 'Gone with the Wind', 'right wing', 'meat-heavy']
```

NOTE ON PRINTING AS WE GO

As we write this code, at each step let's make our code print something out so we know we're making progress!

When you're feeling stuck, trying printing something for every new line of code you write.

NESTED LOOP

```
# select a user + init variables to keep track of scores
index_of_user = 17
top_score = 0
top_record = ""
recommendation = ""

# load data
with open("fake_data.csv", "r") as file:
    all_lines = file.readlines()
    header = all_lines[0]
    records = all_lines[1:]

    user_record = records[index_of_user].strip().split(",")

    # use a loop to go through all other record lines (one record = one use
    for record in records:
        columns = record.strip().split(",")
```

```
Skipping  Luffy,Shrek,Finding Nemo,very right wing,meat-heavy

You are ['Luffy', 'Shrek', 'Finding Nemo', 'very right wing', 'meat-heavy']
After careful consideration, we have found that the most similar user is
```

```
Luke Skywalker,Jurassic Park,The Dark Knight,very right wing,meat-heavy

You have a similarity score of 2
We recommend you watch The Dark Knight
```