CMPT 120 LECTURE 6-1

Prof. Nick Vincent

Simon Fraser University

2024-10-07

6-1 AGENDA

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

REVIEW QUESTIONS

QUESTION 6-1

What does this snippet output?

```python
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 6-1

```
You're a fan of the rain?
```

QUESTION 6-2

```python
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 6-2

```
Something went wrong
```

QUESTION 6-3

What does this output?

```
fruits = "durian, rambutan, lychee"
fruit_list = fruits.split(",")
print("durian" in fruit_list)
print("rambutan" in fruit_list)
```

QUESTION 6-3

```
True
False
```

QUESTION 6-4

What does this output?

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

QUESTION 6-4

```
R
```

QUESTION 6-5

Find three errors with this code:

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

QUESTION 6-5 ANSWER

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

QUESTION 6-6

What does this snippet output?

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

QUESTION 6-6

What does this snippet output?

```python
#                 0    1    2    3    4    5
letters = ['a', 'b', 'c', 'd', 'e', 'f']
print(len(letters))
print(letters[1:3])
print(letters[:4])
print(letters[3:])
print(letters[:])
print(letters[-1])
print(letters[3:-1])
print(letters[1::2])
print(letters[-1:-6:-2])
```

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

QUESTION 6-7

SPLIT A STRING INTO A LIST

Your coworker has created a very annoyingly “delimited” string. How can we split with a single call to .split()

We need to handle four delimiters: `, ; !`

We will use a new function: `.replace()`

`.replace()` takes two arguments and outputs a new string. In the new string, the first argument will be be replaced with the second, e.g.

“123”.replace(“3”, “4”) -> “124”

```
text = "csv,files,are;fun!aren't they"
```

```python
text = "csv,files,are;fun!aren't they"
text.replace(
    ";", ",").replace("!", ",").replace("!", ","
).replace(" ", ",").split(",")
```

```
['csv', 'files', 'are', 'fun', "aren't", 'they']
```

ACCUMULATOR WITHOUT SUM

“Go through all the items in a pile and pick out the red ones”

```python
items = ["red", "blue", "red", "green"]
items_we_grab = []
for item in items:
    if item == "red":
        items_we_grab += [item]
print(items_we_grab)
```

```
['red', 'red']
```

ACCESS SPECIFIC ELEMENTS OF A LIST USING INDEXING/SLICING

```python
numbers = [1, 2, 3, 4, 5]
first_element = numbers[0]    # 1
sliced_elements = numbers[1:4]  # [2, 3, 4]
```

ACCESS SPECIFIC CHARACTER IN A STRING USING INDEXING/SLICING

```python
text = "Hello World"
first_char = text[0]  # 'H'
sliced_chars = text[6:11]  # 'World'
```

PERFORM COMPARISONS BETWEEN NUMBERS, TAKING INTO ACCOUNT ORDER OF OPERATORS (OPERATOR PRECEDENCE)

```python
result = 2 + 3 * 4 > 10  # Evaluates to 2 + 12 > 10 => 14 > 10 => T
print(result)
```

```
True
```

PERFORM COMPARISONS (E.G. !=, <,>) WITH STRINGS

```python
string1 = "apple"
string2 = "banana"
print(string1 != string2)  # True
print(string1 < string2)   # True, because 'a' comes before 'b' alp
```

```
True
True
```

TRICK TO REMEMBER COMPARISON WITH STRINGS:

- Compare the first letter… if same, compare the second
- When comparing, “less than” / < means before
- “a” < “b” -> True (a is before b, so it is “less than b”)
- “A” < “a” -> True (capital letters are before lowercase)

INTERPRET CODE WITH NESTED CONDITIONALS + COMPARISON OPERATORS (E.G. !=, <,>=)

```python
x = 5
y = 10
if x != y:
    if x < y:
        print("x is less than y")
    else:
        print("x is greater than y")
else:
    print("x is equal to y")
```

```
x is less than y
```

FIND THE COMMON ELEMENTS BETWEEN 2 LISTS

- Any ideas?
- What concepts will we use? What Python keywords?

FIND THE COMMON ELEMENTS BETWEEN 2 LISTS

```python
list1 = [1, 2, 3, 4]
list2 = [3, 4, 5, 6]
common_elements = []
for x in list1:
    if x in list2:
        common_elements.append(x)
print(common_elements)  # [3, 4]
```

```
[3, 4]
```

UNDERSTAND + USE NESTED FOR LOOPS

```python
for i in range(3):
    for j in range(2):
        print(i, j)
```

```
0 0
0 1
1 0
1 1
2 0
2 1
```

APPLY OPERATOR ORDER TO EVALUATE EXPRESSIONS

```python
result = 3 + 4 * 2 / (1 - 5)  # Evaluates using operator precedence
print(result)
```

```
1.0
```

CONCATENATE LISTS

```python
list1 = [1, 2, 3]
list2 = [4, 5, 6]
concatenated_list = list1 + list2
print(concatenated_list)  # [1, 2, 3, 4, 5, 6]
```

```
[1, 2, 3, 4, 5, 6]
```

APPLY ACCUMULATION PATTERN FOR STRINGS AND LISTS (PREVIOUSLY WAS NUMBERS)

```python
# For strings
words = ["Hello", "World"]
sentence = ""
for word in words:
    sentence += word + " "
print(sentence.strip())  # Hello World

# For lists
matrix = [[1, 2], [3, 4]]
flat_list = []
for sublist in matrix:
    for item in sublist:
        flat_list.append(item)
print(flat_list)  # [1, 2, 3, 4]
```

```
Hello World
[1, 2, 3, 4]
```

CALCULATE THE MAXIMUM AMONG SEVERAL VALUES

```python
values = [3, 5, 2, 8, 6]
maximum = values[0]
for value in values:
    if value > maximum:
        maximum = value
print(maximum)  # 8
```

```
8
```

OK, BACK TO CONTENT

- We want to extend our recommender systems example

LET'S RECAP...

```python
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", "Scar
         "Mickey Mouse", "Hermione Granger", "Naruto", "Serena Will
         "Leonardo DiCaprio", "Mario", "Taylor Swift", "Darth Vader
         "Usain Bolt", "Tony Stark", "Bugs Bunny", "The Rock", "Wal
         "Sheldon Cooper", "Mulan", "Elsa", "Sherlock Holmes", "Jac
         "Luffy", "Luke Skywalker", "Marilyn Monroe", "Freddie Merc
         "Superman", "Wonder Woman", "Neo", "Beyonce", "Ash Ketchum
         "Lara Croft", "Elvis Presley", "Kermit the Frog", "Spider-
         "Indiana Jones", "Simba", "Rihanna", "Dexter Morgan", "Mor
         "Wolverine", "Cinderella", "Captain America", "Thor", "Kat

movies_by_genre = {
    "Drama": ["The Shawshank Redemption", "Forrest Gump", "The Godf
    "Action": ["The Dark Knight", "Avengers: Endgame", "Pulp Fictio
    "Sci-Fi": ["Inception", "The Matrix", "Back to the Future", "St
```

```
name, favorite_movie, second_favorite_movie, preferred_political_party,
ideal_diet
Marilyn Monroe, Gone with the Wind, Gone with the Wind, left wing, balanced
```

CHECK THAT YOU CAN READ IT

```python
with open("fake_data.csv", "r") as file:
    header = file.readline()
    print(header)
    line = file.readline()
    print(line)
```

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

Marilyn Monroe,Gone with the Wind,Gone with the Wind,left wing,balanced
```

SIMILARITY

Our initial 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, recommen

# load data
    # use a loop to go through all other record lines (one record =
    # 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

```python
# 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 =
    # 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

```python
# 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
    header = file.readline()
    print(header)
    # use a loop to go through all other record lines (one record =
    # 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
```

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

FIRST FOR LOOP

```python
# 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 =
    for record in records:
        if record == user_record:
```

```
['Marilyn Monroe', 'Gone with the Wind', 'Gone with the Wind', 'left wing',
'balanced']
['Goku', 'Lord of the Rings: The Fellowship of the Ring', 'Avengers: Endgame',
'very left wing', 'dairy-heavy']
```

```
['Sheldon Cooper', 'The Dark Knight', 'Jurassic Park', 'very right wing',
'meat-heavy']
['Thor', 'Pulp Fiction', 'The Dark Knight', 'very right wing', 'plant-heavy']
['Katniss Everdeen', 'Gone with the Wind', 'Gone with the Wind', 'right wing',
'meat-heavy']
['Tom Hanks', 'The Shawshank Redemption', 'The Shawshank Redemption', 'very
left wing', 'balanced']
['Marilyn Monroe', 'The Matrix', 'Back to the Future', 'very right wing',
'meat-heavy']
['Luke Skywalker', 'The Godfather', 'Titanic', 'very right wing', 'fruit-
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 =
    for record in records:
        columns = record.strip().split(",")

        if user_record[0] [illegible] columns[0]
```

NESTED LOOP

```
Skipping  Jack Sparrow,Avengers: Endgame,Lord of the Rings: The Fellowship of
the Ring,very right wing,plant-heavy

Skipping  Jack Sparrow,Star Wars: Episode IV,Inception,left wing,plant-heavy

You are ['Jack Sparrow', 'Star Wars: Episode IV', 'Inception', 'left wing',
'plant-heavy']
After careful consideration, we have found that the most similar user is
Scarlett Johansson,Shrek,Frozen,left wing,plant-heavy

You have a similarity score of 2
We recommend you watch Frozen
```