Final Project Summary: Flashcard Study Application

1. IDEA (10 pts)

What I decided to build: I chose to build a flashcard study application to help me prepare for my upcoming Managerial Accounting exam. The program reads flashcards from a text file and quizzes me on the terms and definitions in an interactive study session.

Specific features the program does:

- Reads 49 flashcards from a text file (flashcards.txt) where each line contains a term and definition separated by a colon
- Randomizes the order of flashcards so I don't just memorize the sequence
- Studies in batches of 10 cards at a time to make studying more manageable
- Shows me each term, lets me try to recall the answer, then reveals the definition
- Tracks which cards I get right and wrong in each batch
- Immediately reviews the cards I missed after each batch
- Allows me to continue to the next batch or stop studying
- Shows overall statistics at the end (total correct/incorrect and percentage)

Why this scope was appropriate: This project was simple enough to complete but complex enough to require real problem-solving. It's not just a basic input/output program—it involves file handling, data structures, randomization, user interaction, and program flow control. The batch and review features add meaningful complexity without being overwhelming.

---

2. RESEARCH (35 pts)

How flashcards work in the real world: Flashcards are a proven study technique based on active recall. You see a term or question, try to remember the answer, then check if you were right. Reviewing wrong answers immediately helps reinforce what you missed. Studying in smaller batches is more effective than trying to memorize everything at once.

How a computer represents flashcards: I learned that flashcards can be represented as data structures in Python:

- Each individual flashcard is a **dictionary** with two keys: `"term"` and `"definition"`
- A collection of flashcards is a **list** that contains multiple dictionaries

- Example: `[{"term": "Controlling", "definition": "The process of..."}, {...}]`

Technical concepts I researched:

File I/O (Input/Output):

- How to open a file: `open("filename.txt", "r")` where "r" means read mode
- How to read all lines: `.readlines()` returns a list where each element is one line
- Why to close files: `file.close()` to free up system resources

String manipulation:

- `.split(":")` breaks a string into parts at the colon character, returning a list
- `.strip()` removes extra whitespace (spaces, newlines) from the beginning and end of strings
- This was crucial because my file format was "Term: Definition" and I needed to separate them

Data structures:

- Lists: ordered collections that can hold multiple items, accessed by index
- Dictionaries: key-value pairs that let me store related data together (term with its definition)
- Why I chose these: Lists are perfect for collections of flashcards, dictionaries are perfect for pairing terms with definitions

Randomization:

- `import random` gives access to randomization functions
- `random.shuffle(list)` randomizes the order of items in a list in-place
- This ensures I'm not just memorizing the order of flashcards

User input:

- `input("prompt")` displays a message and waits for the user to type something
- `input()` with no message just waits for Enter to be pressed
- `.lower()` converts text to lowercase so "Y", "y", "Yes" all work the same

Program flow:

- `for` loops to iterate through flashcards
- `if/else` statements to handle correct/incorrect answers
- Breaking lists into batches using slicing: `flashcards[start:end]`

- Calculating number of batches needed: `(total + batch_size - 1) // batch_size`

Resources I used:

- Python documentation on file handling
- Research on string methods like split() and strip()
- Understanding of list and dictionary data structures
- Information on the random module

Kept professor in the loop: Throughout the process, I communicated about my progress and asked for guidance when deciding on features like batch studying and immediate review.

---

3. MAKE THE COMPUTER DO ONE CONCRETE THING (15 pts)

The core technical task I identified: Before I could have a working flashcard app, I needed to make the computer **read my text file and parse it into usable data**. This was the foundational step—without it, nothing else would work.

Breaking it down (what the computer actually needs to do):

1. Open the file containing my flashcards
2. Read all the text content line by line
3. For each line, separate the term from the definition (split at the colon)
4. Store each term-definition pair in a way the program can use later
5. Put all the flashcards into a collection I can iterate through

My mental model: I thought of it like this: the computer needs to take raw text (just characters on a page) and turn it into structured data (organized information it can work with). The text file is just strings, but I need to transform those strings into a list of flashcard objects that the program can quiz me on.

The concrete first goal: Get Python to open my flashcards.txt file, read it, and print out the contents to prove it can access the file. Once that worked, I could move on to parsing and structuring the data.

---

4. TRANSLATE TO PYTHON (15 pts)

How I learned to express my ideas in code: I used a combination of researching Python syntax, understanding what each function does, and trial-and-error testing to figure out the right code to write.

The translation process:

Step 1 - Reading the file:

- Mental idea: "Open the file and get its contents"
- Python code: