User Study Experiment Pitch:

Question:
How does the explicit memory management in C (e.g., using `malloc` and `free`) versus Python's automatic memory management affect beginner programmers' understanding of program resource usage?

Why I'm Interested:
When I started with Python, I loved how easy it was to write code without worrying about memory, but switching to C felt like kind of a shock—suddenly, I had to manually allocate and free memory, and I kept running into segmentation faults. It made me wonder how much this control helps or hinders beginners. Does managing memory yourself make you more aware of what's happening under the hood, or does it just confuse people who are still learning the basics? I'm curious because it's a big difference between the two languages I know, and I want to see how it shapes others' learning experiences.

Break-Down Using School-of-Thought Archetypes:
- Social Scientist: This will be my main role. I'll design a user study to test how programmers grasp resource usage in C versus Python, focusing on their ability to predict or explain memory-related behavior. My falsifiable hypothesis is: "Programmers will more accurately predict memory usage in a C program requiring explicit memory management than in a Python program with automatic management."

Falsifiable Hypothesis:
"Beginner programmers will more accurately predict memory usage in a C program requiring explicit memory management than in a Python program with automatic management." This can be tested by comparing how well participants explain or anticipate memory behavior in each language.

Testing the Hypothesis in a Classroom Setting:

I'll create two short programs: one in C using `malloc` to allocate an array and `free` to release it, and one in Python doing the same task (e.g., creating a list) with no explicit memory management. Both will have a simple output (e.g., printing the array/list). During the one-hour class session, I'll use my 10-minute slot to test my table simultaneously. Here's the plan:

- Setup: I'll hand out printed code snippets (one C, one Python) and explain the task: "Read the code, then answer two questions: (1) How much memory does this program use? (2) When is that memory released?" I'll ensure everyone has a pencil and paper to write answers. No coding is needed—just analysis.

- Execution: Each student reads both snippets and writes their answers independently. I'll use a timer to keep it on track. The C code will explicitly show memory allocation (e.g., `malloc(10 * sizeof(int))`), while Python's list creation (e.g., `lst = [0] * 10`) hides it.

- Data Collection: I'll collect their answer sheets and ask a quick follow-up: "Which language made it easier to understand memory usage?" This fits the 10-minute window and lets everyone participate at once.
I'll measure accuracy by comparing their answers to the actual memory usage (e.g., 40 bytes for 10 integers in C, approximated in Python) and when it's freed (explicitly in C, garbage-collected in Python). Their feedback will add qualitative insight.

Preparation:
Since I've worked with Python and C, I'm comfortable writing these snippets. We've covered basic memory management in C (e.g., `malloc`/`free`) and Python's lists in class, so no new material is needed. The snippets will be short—under 10 lines each—to keep the task beginner-friendly and doable in 5 minutes. I'll test them myself beforehand to ensure they're clear and the memory usage is straightforward to calculate.