Section 1: Foundations: Local LLM Operations and Prompt Engineering

This first section builds the basic skills needed for all other labs. Developing locally has major learning benefits. It makes the LLM easier to understand, changing it from a remote service to something real on the student's computer. This gives a direct, hands-on understanding of resource limits, like RAM and VRAM needs, which are often hidden when using APIs.[4] It also creates a free environment for unlimited testing—important for learning prompt engineering by trying things over and over, as the curriculum points out.[1] After this section, every student will be able to install, manage, and interact with local models through both the command line and Python.

Lab 1.1: Environment Setup and First Interactions

Learning Objectives: Install Ollama, download multiple LLMs from the library, and run a basic interactive chat session from the command line (CLI).

Technical Background: This lab covers the first steps for any hands-on AI development and directly applies the curriculum's focus on hands-on work.[1]

Procedure:

1. **Install Ollama:** Follow the OS-specific installation instructions. For Linux and macOS, run curl -fsSL https://ollama.com/install.sh | sh. For Windows, download and run the installer from the official website.[3]
2. **Download Models:** Open a terminal or command prompt and use the ollama pull command to download two models from the official library: Meta's Llama 3 (8 billion parameters) and Google's Gemma 3 (4 billion parameters).[2]

```
Bash
ollama list
ollama rm gemma3:4b
```

3. **Run an Interactive Session:** Use the ollama run command to start a chat session with one of the downloaded models.[3]

```
Bash
ollama run llama3:8b
```

4. **Interact and Exit:** Ask the model a few questions (e.g., "Why is the sky blue?"). Look at the response. To exit the session, type /bye and press Enter.[5]
5. **Manage Models:** Use the ollama list command to see all models on your machine. To remove a model, use the ollama rm command followed by the model name.[5]

```
Bash
ollama list
ollama rm gemma3:4b
```

Lab 1.2: Model Customization with Modelfile

Learning Objectives: Create a custom model with a permanent system prompt and different generation settings.

Technical Background: This lab introduces model customization, a way to create specialized AI assistants. It applies the idea of a "persona" by building it directly into a new, custom model to make its behavior consistent.[1]

Procedure:

1. **Understand Modelfile:** A Modelfile is a text file that defines a custom model. Key instructions are FROM (the base model), PARAMETER (sets generation settings like temperature), and SYSTEM (defines the system prompt).[4]
2. **Create the Modelfile:** Create a new file named Modelfile (no extension) and add the following to define a "Socratic Tutor" persona. A lower temperature value makes the model's output more predictable and focused.[4]

```
FROM llama3:8b
```

```
PARAMETER temperature 0.2

SYSTEM """You are a Socratic tutor. You never give direct answers. Instead,
you respond to every query by asking a thoughtful, guiding question to help
the user discover the answer for themselves."""
```

3. **Build the Custom Model:** Use the ollama create command to build the new model from your Modelfile. The -f flag points to the file path.[12]

```
Bash
ollama create socratic-tutor -f./Modelfile
```

4. **Test the Custom Model:** Run the new model. Ask it a direct question like, "What is the capital of France?" and see how its behavior has changed to match the system prompt.

```
Bash
ollama run socratic-tutor
```

Lab 1.3: Programmatic Control with the Ollama Python Library

Learning Objectives: Use the official ollama-python library to generate responses, handle streaming output, and manage API errors in a Python script.

Technical Background: This lab moves from CLI interaction to controlling with code, an important step for using LLMs in applications. The code from this lab will be a base for all future Python labs.

Procedure:

1. **Install the Library:** Install the official Python client for Ollama using pip.[3]

```
Bash
```

```
pip install ollama
```

2. **Basic Generation:** Create a Python script (app.py) and use the ollama.chat() function to send a message to a model and print its response.[12]

```
Python
import ollama

response = ollama.chat(
    model='llama3:8b',
    messages=[{'role': 'user', 'content': 'Explain the importance of bees
in 100 words.'}]
)

print(response['message']['content'])
```

3. **Streaming Responses:** Change the script to enable streaming output by setting stream=True. This shows the response token-by-token, which improves the user experience for long responses.[14]

```
Python
import ollama

stream = ollama.chat(
    model='llama3:8b',
    messages=,
    stream=True
)

for chunk in stream:
    print(chunk['message']['content'], end='', flush=True)
```

4. **Error Handling:** Good applications need to handle errors. Change the script to ask for a model that is not on your machine. Use a try...except block to catch the ollama.ResponseError, check for a 404 status code, and automatically download the

missing model before trying again.[14]

```
Python
import ollama

model_name = 'gemma3:1b' # A model we haven't pulled yet

try:
    ollama.chat(model=model_name, messages=[{'role': 'user', 'content':
'Hi!'}])
    print(f"Successfully connected to {model_name}")
except ollama.ResponseError as e:
    if e.status_code == 404:
        print(f"Model '{model_name}' not found. Pulling model...")
        ollama.pull(model_name)
        print("Model pulled successfully. Please run the script again.")
    else:
        print(f"An error occurred: {e.error}")
```

Lab 1.4: Practical Advanced Prompting with Llama 3 and Gemma 3

Learning Objectives: Use advanced prompting techniques (Chain-of-Thought) to solve a simple reasoning problem and compare the performance of Llama 3 and Gemma 3.

Technical Background: This lab directly uses the core concepts of Module 2 from the curriculum, moving from simple questions to structured prompting that shows how a model reasons.[1]

Procedure:

1. **The Logic Puzzle:** Give the following logic puzzle to the models: "I have three boxes, one labeled 'Apples', one labeled 'Oranges', and one labeled 'Apples and Oranges'. I know that every single box is mislabeled. I am allowed to pick only one fruit from only one box to determine the correct labels for all three. Which box should I pick from?"
2. **Attempt 1 (Zero-Shot Prompting):** Using the Python script from Lab 1.3, ask the question directly to both llama3:8b and gemma3:4b. Record their answers. They will likely be wrong because the problem needs careful logic.
3. **Attempt 2 (Chain-of-Thought Prompting):** Change the prompt to include the instruction: "Think step-by-step before providing the final answer." This technique, known as Chain-of-Thought (CoT), encourages the model to show its reasoning steps, which

often gives better results in multi-step problems.[1]

```
Python
# Example prompt content for Attempt 2
prompt = """
Solve the following logic puzzle: I have three boxes, one labeled 'Apples',
one labeled 'Oranges', and one labeled 'Apples and Oranges'. I know that
every single box is mislabeled. I am allowed to pick only one fruit from
only one box to determine the correct labels for all three. Which box
should I pick from?

Think step-by-step before providing the final answer.
"""
```

4. **Analysis:** In a separate text file, write a short comparison of the models' responses. Note which model solved the puzzle correctly with each prompt. Compare the clearness and logic of the "step-by-step" reasoning from Llama 3 and Gemma 3.

The following table is a quick reference for the main Ollama CLI commands used in this section.

Table 1: Core Ollama CLI Command Reference

Command	Example Usage	Description
ollama run	ollama run llama3:8b	Starts an interactive chat session with a model.[4]
ollama pull	ollama pull gemma3:4b	Downloads a model from the Ollama library to your computer.[2]
ollama list	ollama list	Shows all models that have been downloaded locally.[2]
ollama rm	ollama rm llama3:8b	Removes a model from your computer.[13]

ollama create	ollama create my-model -f./Modelfile	Creates a new custom model from a Modelfile.[4]
ollama cp	ollama cp llama3:8b my-llama-copy	Creates a copy of a model with a new name.[4]