Module 1: The Anaconda Ecosystem & AI Setup

Jupyter Notebook Interface Configuration • Instructor Prompt Engineering Lab

Participant Name: _____ **Date:** _____

Section 1: The Visual Development Environment

Anaconda Navigator provides a fully integrated visual launchpad designed to manage local programming environments without interacting with command-line interface terminal utilities. By abstracting the terminal framework, you can deploy notebooks, verify runtime installations, and switch code sandboxes with point-and-click operations.

Section 2: Mastering the Jupyter Notebook Interface

Jupyter Notebook structures are built around independent functional elements called cells. Cells typically run in one of two distinct modes:

- **Code Cells:** Executable programming environments where Python statements can be compiled, isolated, and executed sequentially.
- **Markdown Cells:** Formatting note layouts designed for documentation, structural notes, titles, and explanations.

⌨️ **Critical Keyboard Shortcuts Reference:**

- `Shift + Enter` → Execute the currently selected cell block and jump to the next layer.
- `A` (in Command Mode) → Insert a completely fresh blank cell block **Above**.
- `B` (in Command Mode) → Insert a completely fresh blank cell block **Below**.
- `M` / `Y` → Toggle the selected cell mode between **Markdown documentation** and **Python Code**.

Section 3: Customizing Your Generative AI Persona

To turn generative language models (like ChatGPT or Claude) into optimized learning tools, you must explicitly declare their functional boundaries. If unguided, AI models often return advanced libraries, overly complex optimizations, or unannotated code.

By prescribing a strict role blueprint—defining it as a **Patient Python Tutor**—you force the engine to favor readable code structures and deep contextual explanations over high-level professional abstractions.

Section 4: Active Lab Verification & Analysis

Review the training architecture discussed in Module 1 to answer the optimization questions below.

Question 1: UI Modal Identification

In a Jupyter Notebook environment, if a cell is currently displaying plain text notes, equations, or headers rather than executable blocks of code, what mode is this cell running in, and what shortcut key instantly switches it back to a standard Python environment?

Question 2: Terminal Command Abstraction

Explain the specific operational and onboarding benefits that **Anaconda Navigator** brings to beginning data professionals compared to launching workspaces manually through terminal command consoles:

Question 3: The AI Persona Strategy

Why does configuring a customized system role profile (e.g., instructing an AI engine to act as a patient teacher rather than a general programming assistant) drastically improve learning efficiency for individuals running code for the first time?

Section 5: Practical Prompt Construction Sandbox

🚀 **"The Golden Prompt" Assignment:** Using the instructional guidelines from Module 1, write a complete, raw system-level persona setup prompt inside the box below.
Your structured prompt blueprint must explicitly mandate three strict criteria:

1. The AI must adopt the role of a patient, absolute-beginner Python tutor.
2. It must restrict code outputs to fundamental data types, avoiding complex, advanced libraries.
3. It must enforce line-by-line internal commentary for every code block generated.

Write Your Golden Persona Setup Prompt Below:

```
# Write out your custom, copy-pasteable tutor configuration script layout below:
```