Here’s a simple, very human 12-step plan to build your Personal Firewall project using Python:

Step 1: Understand the Project and Set Clear Goals

Before writing any code, spend some time figuring out exactly what you want your firewall to do. Write down what types of traffic you want to block or allow (for example, specific IP addresses, ports, or protocols). Sketch a rough flow of how packets should be handled, and decide which features you’ll implement first.

Step 2: Set Up Your Development Environment

Create a new folder or GitHub repository for the project. Set up a Python virtual environment and install the necessary packages, especially Scapy for packet sniffing. List all your dependencies in a requirements file.

Step 3: Build a Basic Packet Sniffer Using Scapy

Write a simple Python script that uses Scapy to capture both incoming and outgoing network packets. For now, simply print out packet details (such as source/destination IP, port, and protocol) to verify that your script is working as expected.

Step 4: Define Your Rule Sets

Decide on the rules you want your firewall to enforce. Create a simple data structure (like a Python dictionary or lists) to store allowed and blocked IP addresses, ports, and protocols. Make sure it’s easy to update and modify these rules later on.

Step 5: Integrate Rule Checking into Your Sniffer

Enhance your packet sniffer by adding logic that checks each captured packet against your predefined rule sets. If a packet violates the rules (for example, coming from a blocked IP or targeting a forbidden port), mark it so you can later log or take action against it.

Step 6: Implement Logging for Suspicious Packets

Add a logging mechanism using Python’s built-in logging module or by writing directly to a log file. Record key details like timestamps, source/destination IPs, ports, and the reason why a packet was flagged. This log will be vital for later audits and for checking that your firewall is working correctly.

Step 7: Develop a Command-Line Interface (CLI)

Create a simple CLI that lets you view, add, or remove firewall rules while your script is running. This interface should also allow you to check log entries on the fly, making it easier to manage your firewall without stopping the program.

Step 8: Optionally Integrate iptables Commands on Linux

For a more robust solution, integrate system-level enforcement by using Python’s subprocess module to run iptables commands. This step lets your script actually block unwanted traffic at the operating system level, but be careful and test in a controlled environment.

Step 9: Build a Simple Graphical User Interface (GUI) with Tkinter

If you'd like a visual way to monitor your firewall, build a basic Tkinter GUI. Create a window that displays live traffic logs and current firewall rules, and include buttons to add or remove rules. This step can be added later if you want to start with CLI first.

Step 10: Test Your Firewall Thoroughly

Now that you have all the pieces working, test your firewall under various scenarios. Simulate network traffic (or use a controlled test network) to make sure packets are captured, rules are enforced, and logs capture all the necessary details accurately.

Step 11: Clean Your Code and Write Documentation

Once everything works together, clean up your code. Add comments to explain your logic and how the different pieces work. At the same time, prepare your project report (1–2 pages max) that covers:

- Introduction: What the project is about and its importance.

- Abstract: A short summary of the project and its functionalities.

- Tools Used: List Python, Scapy, iptables, Tkinter, etc.

- Steps Involved: An overview of your 12-step process.

- Conclusion: What you learned from the project and potential future improvements.

Step 12: Finalize and Prepare for Submission

Do one final round of testing and code review. Push all your changes to your GitHub repository and double-check that your report meets the guidelines without exceeding 2 pages. Once you're satisfied, prepare your submission package with the GitHub link and your final report PDF.

This roadmap breaks your work into manageable, human-friendly steps, ensuring that each part of the project is clearly understood before moving on to the next. Developing your own custom firewall this way makes it personal and greatly improves your learning experience. Enjoy the process, and feel free to explore and tweak each step as you become more comfortable with the tools. Happy coding!