## **Python Mini-Project: Barebones "Read-Only" HTTP Server**

**Objective:** This project assesses your foundational understanding of network programming in Python and basic HTTP protocol concepts. You'll build a minimalistic HTTP server that can serve static JSON data for a simple "items" resource.

**Tools:** Python's socket, re, typing, and json modules. **No built-in HTTP server libraries (e.g., http.server, Flask, Django) are allowed.**

**Scenario:** You need to create a server that provides a read-only list of inventory items.

### **Task Description:**

Your task is to create a Python script that implements a basic HTTP server.

**Core Requirements:**

1. **Server Setup:**
   * Create a **TCP server** using the socket module.
   * Bind the server to localhost and a chosen port (e.g., 8000).
   * Listen for incoming client connections.
   * Handle **one client connection at a time** (no need for multi-threading or asyncio for this condensed version). After serving a request, the server can optionally close the connection or keep it open for a single subsequent request, then close.
2. **Request Parsing:**
   * Upon receiving data from a client, **manually parse the incoming HTTP request**.
   * Extract the **HTTP method** (only GET is required) and the **request path**.
   * Use the re module for basic parsing of the request line (e.g., GET /items HTTP/1.1).
   * Ignore any request body or headers beyond the request line for this project.
3. **Data Storage (File-Based - Read-Only):**
   * Create a static JSON file named data.json in the same directory as your server script.
   * This file will contain a hardcoded list of items.

**Example data.json content:** JSON  
[

{"id": 1, "name": "Laptop"},

{"id": 2, "name": "Mouse"},

{"id": 3, "name": "Keyboard"}

]

* + Your server should **load this data once when it starts up** and use it for responses. No writing back to the file is required.

1. **Response Generation:**
   * Construct valid **HTTP responses**, including:
     + **Status line** (e.g., HTTP/1.1 200 OK, HTTP/1.1 404 Not Found).
     + Content-Type header: application/json.
     + Content-Length header.
     + Response body (JSON encoded).
   * Send the complete HTTP response back to the client.
2. **API Endpoints (Read-Only):**
   * **GET /items:**
     + **Description:** Retrieve all items.
     + **Response:** A JSON array of all items loaded from data.json.
     + **Status Codes:** 200 OK.
   * **GET /items/{id}:**
     + **Description:** Retrieve a single item by its ID.
     + **Response:** A JSON object representing the item if found, otherwise an error message.
     + **Status Codes:** 200 OK (item found), 404 Not Found (item not found).
   * **Handling Other Paths/Methods:** For any other requested path or HTTP method, respond with 404 Not Found or 405 Method Not Allowed (if a valid path but wrong method).

### **Deliverables:**

1. **server.py:** Your main Python script containing the HTTP server implementation.
2. **data.json:** The static JSON file containing your item data.
3. **README.md:** A brief document explaining:
   * How to run your server.
   * The API endpoints it exposes with example curl commands for GET /items and GET /items/{id}.