BEGINNER-FRIENDLY PYTHON COURSE CURRICULUM

Objective:

By the end of the course, students will create a Quiz Game that works in the terminal. The game will allow users to answer questions from a predefined list loaded from a JSON file, and upon completion, it will display the total marks, the number of correct answers, and incorrect answers. In the second stage, students will create a basic UI using Python Django, JSON-based question loading, HTML, CSS, and JavaScript.

Phase 1: Terminal-Based Quiz Game

Module 1: Introduction to Python

- **Lesson 1.1: Setting Up the Environment**
  - Downloading and installing Python on Windows.
  - Setting up Visual Studio Code or any Python IDE.
  - Installing necessary extensions/plugins.
  - Running a "Hello World" program in Python.
- **Lesson 1.2: Python Basics**
  - Variables and Data Types.
  - Input and Output.
  - Basic Operators (Arithmetic, Relational, Logical).
  - Comments and code readability.

Module 2: Control Flow and Data Structures

- **Lesson 2.1: Conditional Statements**
  - if, elif, else statements.
  - Examples like checking even/odd numbers.
- **Lesson 2.2: Loops**
  - for and while loops.
  - Loop control statements (break, continue).
- **Lesson 2.3: Lists and Tuples**
  - Defining and accessing elements.

    - Iterating through lists.
    - Basic operations on lists (adding, removing elements).
- **Lesson 2.4: Dictionaries**
    - Key-value pairs.
    - Accessing, updating, and deleting values.
    - Iterating through dictionary items.

Module 3: Functions

- **Lesson 3.1: Defining and Using Functions**
    - Function syntax.
    - Parameters and return values.
    - Examples (e.g., a function to add two numbers).
- **Lesson 3.2: Understanding Scope**
    - Local vs global variables.

Module 4: Building the Quiz Game

- **Lesson 4.1: Designing the Quiz Logic**
    - Loading questions and answers from a JSON file.
    - Asking the user for input and comparing it with the correct answer.
- **Lesson 4.2: Implementing Score Tracking**
    - Keeping track of correct and incorrect answers.
    - Calculating total marks.
- **Lesson 4.3: Displaying Results**
    - Printing the total score, number of correct answers, and incorrect answers.

Mini-Project (Phase 1)

- **Objective**: Create a terminal-based quiz game using Python.
    - Use a JSON file to define questions and answers.
    - Ask the user a series of questions.

    - Track and display scores and answers at the end.
    - Ensure results are shown after the quiz without storing them.

Phase 2: Adding a Basic UI with Django

Module 5: Introduction to Django

- **Lesson 5.1: Setting Up Django**
    - Installing Django.
    - Creating a Django project.
    - Overview of the Django folder structure.
- **Lesson 5.2: Django Basics**
    - Creating a basic Django app.
    - Understanding views, templates, and URLs.
    - Setting up the development server.

Module 6: Creating the Quiz Game Backend

- **Lesson 6.1: Loading Questions from a JSON File**
    - Reading questions and answers dynamically from a JSON file.
    - Structuring the JSON file for scalability.
- **Lesson 6.2: Connecting Backend and Frontend**
    - Writing views to fetch and display questions.
    - Handling user input and calculating scores.

Module 7: Frontend Basics (HTML, CSS, JS)

- **Lesson 7.1: Introduction to HTML and CSS**
    - Building a simple webpage.
    - Adding styles with CSS.
- **Lesson 7.2: Introduction to JavaScript**
    - Writing simple scripts for interactivity.

- **Lesson 7.3: Integrating Frontend with Django**
  - Using Django templates.
  - Sending data between backend and frontend.

Module 8: Completing the UI

- **Lesson 8.1: Adding Forms and Handling Input**
  - Creating forms for the quiz game.
  - Handling form submissions in Django.
- **Lesson 8.2: Displaying Results**
  - Showing the user's total marks, correct answers, and incorrect answers on the webpage.

Final Project (Phase 2)

- **Objective**: Create a web-based quiz game.
  - **Backend**: Python + Django (using JSON for questions).
  - **Frontend**: HTML + CSS + JavaScript.
  - **Features**:
    - UI for answering questions.
    - Dynamically loaded questions from a JSON file.
    - Results displayed after quiz completion.

Additional Notes

- **Beginner-Friendly Focus**: Lessons are simple, focusing on the essentials.
- **Hands-On Approach**: Practical exercises and examples throughout.
- **Debugging**: Encourage students to debug and test code regularly.
- **Supplementary Resources**: Provide external resources like Python tutorials and Django documentation for deeper learning.
- **Schedule Overview:** 24 days (4 weeks with 6 sessions/week).