Level 1: Python for Backend Development(2 weeks)

Goal
This track introduces fundamental Python programming skills needed to build backend applications. Each section includes topics, practical exercises, projects, and expected outcomes. By completing this track, you'll be prepared to transition into using Flask for web development.

---

1. Introduction to Python

- **Objective**: Grasp the basics of Python programming.
- **Topics**:
    - What is Python? Installation and setup.
    - Basic syntax, variables, and data types.
- **Learning Outcome**: Create a basic Python script.
- **Project 1**: **Simple Calculator**
    - **Task**: Create a calculator for addition, subtraction, multiplication, and division.
    - **Instructions**:
        1. Define a function for each arithmetic operation.
        2. Use `input()` to get user numbers and chosen operation.
        3. Display the result.

Expected Output:
```
Enter first number: 10
Enter second number: 5
Choose operation (+, -, *, /): +
Result: 15
```

---

2. Control Structures

- **Objective**: Learn control flow in Python.
- **Topics**:
    - Conditional statements (`if`, `elif`, `else`)
    - Loops (`for`, `while`)
- **Learning Outcome**: Build programs that respond to user input and repeat tasks.
- **Project 2**: **Number Guessing Game**
    - **Task**: Create a game where the user guesses a randomly generated number within limited attempts.
    - **Instructions**:
        1. Use the `random` module to generate a number between 1 and 100.
        2. Set a limit on the number of attempts.
        3. Provide feedback on each guess.

Expected Output:
```
Guess a number between 1 and 100: 50
Too high! Try again.
Guess a number between 1 and 100: 25
Correct! You guessed the number in 2 attempts.
```

---

3. Functions and Modules

- **Objective**: Understand and use functions and modules.
- **Topics**:
  - Function definition, arguments, return values
  - Importing and using modules
- **Learning Outcome**: Create reusable functions and organize code into modules.
- **Project 3**: **To-Do List Manager**
  - **Task**: Develop a CLI to-do list app to add, view, and remove tasks.
  - **Instructions**:
    1. Define functions to add, view, and remove tasks.
    2. Use a list to store tasks.
    3. Enable user interaction via menu options.

Expected Output:
```
Menu:
1. Add Task
2. View Tasks
3. Remove Task
4. Exit
Choose an option: 1
Enter the task: Finish homework
Task added: Finish homework
```

---

4. Data Structures

- **Objective**: Learn about Python's built-in data structures.
- **Topics**:
  - Lists, tuples, sets, dictionaries
- **Learning Outcome**: Use suitable data structures for various tasks.
- **Project 4**: **Student Grades Tracker**
  - **Task**: Create a program to track student grades and calculate the average.
  - **Instructions**:
    1. Use a dictionary to store student names as keys and grades as values.
    2. Define functions to add grades, display all grades, and calculate the average.
    3. Enable user input for student names and grades.

Expected Output:

```
Enter student name: John Doe
Enter grade: 85
Enter student name: Jane Smith
Enter grade: 90
All Grades: {'John Doe': 85, 'Jane Smith': 90}
Average Grade: 87.5
```

---

5. Advanced Project: Expense Tracker

- **Objective**: Build a Python-based expense tracker for effective expense logging and management.
- **Project Steps**:
  - **Step 1**: Set up your project in `expense_tracker.py`.
    - **Goal**: Create a Python script to manage expenses.
  - **Step 2**: Implement `add_expense()` function
    - Allows users to add an expense.
    - **Requirements**: File handling for data storage and error handling.
  - **Step 3**: Implement `read_expenses()` function
    - Retrieves and displays recorded expenses.
    - **Requirements**: File handling and error handling for missing files.
  - **Step 4**: Implement `main()` function
    - Entry point of the program, gathers user choices, and repeats until exit.

Expected Outputs:
```
Choose an option:
1. Add Expense
2. View Expenses
3. Exit
Expense added successfully.
All Expenses:
2024-10-01 - $50 - Groceries
2024-10-02 - $20 - Transport
```

---

6. Introduction to Object-Oriented Programming (OOP)

- **Objective**: Understand OOP principles in Python.
- **Topics**:
    - Classes, objects, inheritance, encapsulation
- **Learning Outcome**: Model real-world entities with classes.
- **Project 6**: **Library Management System**
    - **Task**: Develop a simple library management system to manage books.
    - **Instructions**:
        1. Define a `Book` class with attributes (`title`, `author`, `status`).
        2. Add methods for adding, borrowing, and returning books.
        3. Use a list to store books.

Expected Output:
```
Available Books:
1. Title: Python Basics, Author: John Doe, Status: Available
Enter 1 to borrow "Python Basics".
Book borrowed: Python Basics
```

Each project and section are designed to build your Python skills step-by-step. Upon completion, you'll be ready to advance to using Python for backend development with Flask.