Week 1: Python Basics & Data Types

Focus Areas:

- Python syntax, variables, and data types
- Basic operators
- Strings, numbers, lists, tuples, dictionaries, sets

Learning:

- **Day 1-3**: Variables, Basic Data Types, and Operators
    - Understand how Python handles numbers, strings, booleans, etc.
    - Learn basic operators (`+`, `-`, `*`, `/`, `%`, `==`, `!=`, etc.)
- **Day 4-5**: Lists & Tuples
    - Learn list and tuple operations (indexing, slicing, adding/removing elements)
    - List comprehensions
- **Day 6-7**: Dictionaries & Sets
    - Learn dictionary key-value pairs and how to access/modify them
    - Sets and their properties (uniqueness, membership)

Exercise:

1. **Create a program that stores your favorite books in a dictionary**, where the key is the book title and the value is the author's name.
2. **Create a function to sum all elements in a list** and return the result.

Mini-Challenge:

- Write a Python function to check if a given string is a palindrome.

Week 2: Functions & Control Flow

Focus Areas:

- Functions (arguments, return values)
- Conditionals (`if`, `elif`, `else`)
- Loops (`for`, `while`)
- Recursion

Learning:

- **Day 1-2**: Functions
    - Define functions, return values, function arguments
    - Understand default parameters, `*args`, and `**kwargs`
- **Day 3-4**: Conditional Statements
    - Use `if`, `elif`, and `else` for decision-making
    - Learn logical operators (`and`, `or`, `not`)
- **Day 5-7**: Loops and Recursion
    - Looping through lists, dictionaries, and sets
    - Understand recursion and how it works

Exercise:

1. **Write a program to calculate factorial using recursion.**
2. **Create a function that accepts a list of numbers and returns a list with only the even numbers.**

Mini-Challenge:

- Write a Python function to calculate the Fibonacci series up to the nth term.

Week 3: Object-Oriented Programming (OOP)

Focus Areas:

- Classes and objects
- Attributes, methods, and constructors (`__init__`)
- Inheritance, polymorphism, encapsulation

Learning:

- **Day 1-2**: Classes and Objects
    - Learn how to define classes, instantiate objects
    - Understand the role of the `__init__` constructor
- **Day 3-4**: Inheritance and Polymorphism
    - Understand how inheritance allows code reuse
    - Use `super()` to call parent class methods
- **Day 5-7**: Encapsulation and Abstraction

    - Learn about private/public attributes and methods (_, __)
    - Introduction to abstract classes and interfaces (optional)

Exercise:

1. **Create a `Person` class with attributes like name, age, and method to greet.**
2. **Create a `Student` class that inherits from `Person` and adds attributes like grade and methods like study.**

Mini-Challenge:

- Write a class `Rectangle` that has methods to calculate the area and perimeter. Create another class `Square` that inherits from `Rectangle`.

Week 4: File Handling & Exceptions

Focus Areas:

- Reading and writing to files
- Exception handling (`try`, `except`, `finally`)
- Working with CSV and JSON

Learning:

- **Day 1-3**: File Handling
    - Learn to open files, read/write, and handle file paths
    - Work with file context managers (`with` statement)
- **Day 4-5**: Exception Handling
    - Handle errors using `try`, `except`
    - Understand custom exceptions and raising errors
- **Day 6-7**: CSV and JSON
    - Work with CSV and JSON data using Python's built-in modules

Exercise:

1. **Write a program to read and write to a text file.**
2. **Create a script to parse a CSV file containing names and emails and store the data in a dictionary.**

Mini-Challenge:

- Write a program that logs errors and prints them out using custom exceptions.

Week 5: Advanced Topics in Python

Focus Areas:

- Decorators
- Generators and Iterators
- Context Managers

Learning:

- **Day 1-2**: Decorators
    - Learn what decorators are and how to use them to modify functions.
- **Day 3-4**: Generators and Iterators
    - Understand how generators and iterators work in Python
- **Day 5-7**: Context Managers
    - Use Python's `with` statement and understand the context management protocol

Exercise:

1. **Write a simple decorator that logs the execution time of a function.**
2. **Write a generator to generate an infinite sequence of numbers (like a counter).**

Mini-Challenge:

- Implement a custom context manager using the `contextlib` module.

Week 6: Advanced Python and Clean Code Practices

Focus Areas:

- Typing and Type Hinting
- Writing Pythonic Code
- Using PEP8 and Code Style

Learning:

- **Day 1-2**: Typing & Annotations
    - Understand the role of type hints in Python functions and variables
- **Day 3-4**: Writing Pythonic Code
    - Learn Pythonic approaches to common problems (e.g., using `zip`, `map`, `filter`)

- **Day 5-7**: PEP8 and Code Style
    - Follow PEP8 guidelines and write clean, readable Python code

Exercise:

1. **Write a Python function that takes two lists and returns a list of their intersection.**
2. **Write a Python function to merge two dictionaries.**

Mini-Challenge:

- Refactor a poorly written Python function to make it more Pythonic.

Projects to Implement Along the Way:

1. **Simple Calculator**: Use OOP principles to create a calculator app that can add, subtract, multiply, and divide.
2. **To-Do List App**: Build a to-do list that saves tasks to a file (CSV or JSON).
3. **Weather App**: Build an app that uses an external API (e.g., OpenWeatherMap) to fetch weather data and display it.
4. **Bank Account Simulator**: Create a system where you can deposit, withdraw, and check the balance of a bank account.

Bonus: Daily Mini-Challenges

To keep practicing and reinforcing your learning:

- Solve 1 challenge daily from websites like:
    - LeetCode
    - HackerRank
    - Codewars