PYTHON LEARNING SCHEDULE

TABLE OF CONTENTS

I. Timeline

Week	Topics
1st week	Topics 1 + 2
2nd week	Topic 3
3rd week	Topic 4
4th week	Topic 4 (continued)
5th week	Topics 5 + 6 + 7
6th week	Topics 8 + 9
7th week	Topic 10
8th week	Topic 10 (continued)
9th week	Topic 14
10th week	Topic 14 (continued)
11th week	Topic 14 (continued)
12th week	Topic 11
13th week	Topic 12

14th - 18th weeks	Topics 13 + 15 + 16

II. Topics

1. Python Basics

- Introduction to Python: History, features, and setup
- Python syntax, variables, and data types (int, float, string, boolean)
- Basic input/output and comments
- Arithmetic, comparison, and logical operators

2. Control Flow

- Conditional statements (if, elif, else)
- Loops: for loops, while loops
- Loop control statements: break, continue, pass

3. Functions

- Defining and calling functions
- Function arguments (positional, keyword, default, arbitrary arguments)
- Return values and scope of variables
- Lambda functions and higher-order functions
- Recursion basics

4. Data Structures+

- Lists: Creating, accessing, modifying, slicing, and list comprehensions
- Tuples: Immutability, usage, packing/unpacking
- Sets: Unique values, set operations (union, intersection, difference)
- Dictionaries: Key-value pairs, dictionary methods, and comprehension
- Arrays vs. Lists

5. Strings

- String methods and operations
- Formatting strings (f-strings, format method)
- String slicing and indexing

- Regular expressions (basic pattern matching)

6. Exception Handling

- try, except, else, finally
- Custom exceptions and raising exceptions

7. File Handling

- Opening, reading, writing, and closing files
- Working with files in different modes (r, w, a, etc.)
- Using with for file operations

8. Object-Oriented Programming (OOP)

- Classes and objects
- Attributes and methods
- Constructor (__init__) and destructors
- Inheritance, polymorphism, and encapsulation
- super() and method overriding
- Dunder methods (__str__, __repr__, etc.)

9. Modules and Packages

- Importing modules and packages
- Using pip to install external packages
- Creating custom modules and packages

10. Advanced Data Structures

- Stacks, Queues, and Linked Lists (introductory concepts for DSA)
- Trees and Graphs (basic implementations)
- Heaps, Hash tables, and priority queues

11. Functional Programming

- map(), filter(), and reduce()
- zip() and list comprehension for functional programming
- Decorators and closures
- Generator functions and yield

12. Working with Libraries for DSA

- collections (Counter, defaultdict, deque)

- itertools for combinatorial operations
- functools for caching and higher-order functions

13. Data Science Basics (Optional)

- Introduction to numpy and pandas for data manipulation
- Basics of data analysis using libraries like matplotlib and seaborn

14. Algorithms and Problem Solving Techniques

- Basic algorithms(prime number etc)
- Searching algorithms (linear, binary search)
- Sorting algorithms (bubble, selection, merge, quicksort)
- Understanding recursion for problem-solving
- Dynamic programming and memoization basics
- Backtracking and brute-force techniques

15. Advanced Topics (Beyond Core Python)

- **Concurrency and Parallelism**: threading, multiprocessing, async programming with asyncio
- **Web Development Basics**: Flask or Django for understanding web frameworks
- **Database Interaction**: sqlite3, SQLAlchemy for ORM, basics of interacting with SQL/NoSQL databases
- **Network Programming**: socket programming, HTTP requests with requests library
- **Data Serialization**: JSON, XML, CSV handling, pickle for object serialization
- **Testing and Debugging**: Unit testing with unittest, pytest, debugging techniques
- **Advanced Libraries**: Popular libraries for data analysis (pandas, numpy), visualization (matplotlib, seaborn), and machine learning (scikit-learn, tensorflow)

16. Best Practices

- Writing clean and Pythonic code (PEP 8)
- Using virtual environments for project management
- Writing documentation and type hints (typing module)
- Version control basics with Git