WEEK 1 – Core Python Foundations (Programming Basics)

- Day 1: Install Python, setup environment, run .py files, understand how interpreter works.
- Day 2: Variables, data types (int, float, str, bool), type casting, input/output.
- Day 3: Operators (arithmetic, comparison, logical), if/elif/else conditions.
- Day 4: Loops – for, while, range(), break/continue.
- Day 5: Lists – indexing, slicing, methods, list comprehension.
- Day 6: Tuples, Sets, Dictionaries – key/value access, iteration, nested structures.
- Day 7: Functions – def, return, parameters, *args, **kwargs, basic recursion.

Minimum Knowledge After Week 1

- Write basic programs confidently.
- Solve logic problems using loops and conditions.
- Comfortably use lists and dictionaries.
- Create reusable functions.

WEEK 2 – Intermediate Python & Code Structuring

- Day 8: String methods and string manipulation problems.
- Day 9: File handling – read/write text files, JSON handling.
- Day 10: Exception handling – try/except/finally, raise, custom exceptions.
- Day 11: Modules and packages – import, create modules, __name__ == '__main__'.
- Day 12: OOP basics – class, object, __init__, instance variables.
- Day 13: OOP advanced – inheritance, polymorphism, encapsulation.
- Day 14: Magic methods (__str__, __repr__), small OOP mini-project.

Minimum Knowledge After Week 2

- Write structured OOP-based programs.
- Handle files and JSON confidently.
- Organize code into modules.
- Use exception handling properly.

WEEK 3 – Practical Python (Development + Automation + System Level)

- Day 15: Virtual environments, pip, dependency management.
- Day 16: Logging module – log levels, file logging, formatting logs.
- Day 17: requests library – API calls, JSON parsing.
- Day 18: Intro to pytest – writing and running tests.
- Day 19: Basic Linux interaction using Python (os, subprocess modules).
- Day 20: Intro to multithreading and multiprocessing.
- Day 21: Small project – CLI tool or automation script.

Minimum Knowledge After Week 3

- Write scripts for automation tasks.
- Call APIs and validate responses.
- Use logging properly.
- Understand concurrency basics.
- Manage dependencies using virtual environments.

WEEK 4 – Advanced Concepts & Career Readiness

- Day 22: Decorators – concept and use cases.
- Day 23: Generators and iterators – yield keyword.
- Day 24: Data structures complexity (Big-O basics).
- Day 25: Intro to databases using Python (sqlite3 or basic DB connection).
- Day 26: Solve 10 coding problems (loops/strings/dictionaries).
- Day 27: Solve 10 more problems (logic + OOP).
- Day 28: Build a complete mini-project (API tool, automation script, or small app).
- Day 29: Full revision – OOP, exceptions, modules, concurrency.
- Day 30: Mock interview practice + refine weak areas.

Final Outcome After 1 Month

- Write clean and structured Python code.
- Build automation scripts and small tools.
- Understand OOP and core internals.
- Be ready for Junior Python Developer, SDET, or SRE entry-level roles.