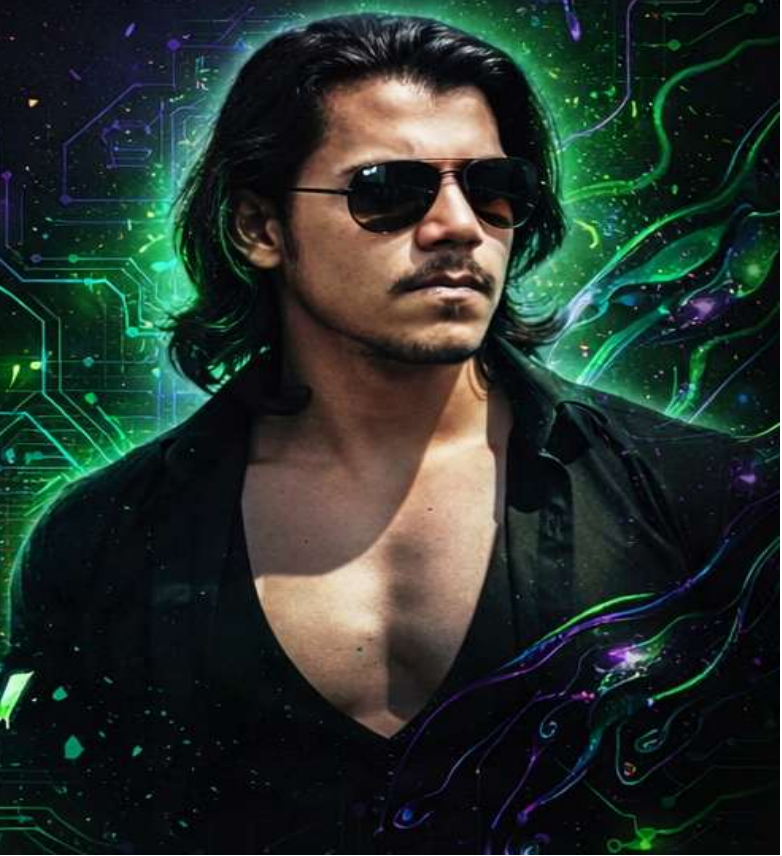
PYTHON PRACTICE SET

— BASIC TO ADVANCED PROBLEMS —

CRACK INTERVIEWS, MASTER PYTHON, AND MORE



AUTHOR:

Shivansh Yadav

(VENOM)

COMPREHENSIVE PYTHON PRACTICE WORKBOOK

About This Book

Python Practice Set – Venom Edition is a professionally designed workbook focused on building practical Python proficiency from basic to advanced levels.

This workbook emphasizes real-world problem solving, covering concepts commonly tested in technical interviews, backend development, data processing, and automation scripting.

Each problem is structured with a clear difficulty level, real-world context tags, and solution guidelines to help learners develop logical thinking, clean coding practices, and long-term problem-solving confidence.

</> How to Use This Workbook

1. Start with Section 1 to build strong Python fundamentals.
2. Progress to Section 2 to understand object-oriented design principles.
3. Use Section 3 to master advanced Python constructs and frameworks.
4. Solve Section 4 for interview-grade and real-world professional challenges.

Attempt problems independently before reviewing solution guidelines. Use the skill-mapping table to align your practice with your career goals.

Skill-Mapping Overview

The table below maps each section of this workbook to core skill areas commonly evaluated in interviews and required in real-world Python roles. This helps learners understand how each problem contributes to professional competency.

Section	Python Core	OOP	Backend	Data
Section 1: Python Basics	✓ Strong	◐ Introductory	◐ Introductory	◐ Introductory
Section 2: Object-Oriented Programming	✓ Moderate	✓ Strong	✓ Moderate	◐ Introductory
Section 3: Advanced Python	✓ Strong	✓ Moderate	✓ Strong	✓ Strong
Section 4: Professional Challenges	✓ Expert	✓ Strong	✓ Expert	✓ Strong

© Copyright & Usage

Appendix

A. Learning Path Recommendations

This workbook supports multiple learning paths depending on career goals:

- **Beginner Path** – Complete Section 1 sequentially.
- **Interview Preparation Path** – Focus on Sections 1, 2, and 3, then selected challenges in Section 4.
- **Backend Development Path** – Prioritize Sections 2, 3, and 4.
- **Data & Automation Path** – Emphasize Sections 1 and 3 with selected problems from Section 4.

B. Interview Preparation Focus Areas

Key interview-focused topics covered in this workbook include:

- Core Python syntax and data structures
- Functions, recursion, and error handling
- Object-oriented programming concepts
- Functional programming techniques
- File handling and memory-efficient design
- Concurrency and asynchronous concepts

C. Backend and Data Skill Alignment

This workbook aligns Python concepts with real-world backend and data use cases.

Backend skills include web frameworks, databases, APIs, logging, testing, and concurrency.

Data skills include transformation, filtering, aggregation, serialization, and efficient data processing using Python's standard library.

D. Best Practices for Effective Practice

- Write clean and readable code.
- Avoid copy-pasting solutions.
- Refactor code after solving each problem.
- Revisit difficult problems periodically.
- Use version control to track learning progress.

E. Final Note to the Learner

Python mastery comes from consistent practice and thoughtful problem-solving.

This workbook is designed to challenge your thinking, strengthen your fundamentals, and prepare you for real-world development and interviews.

Use it deliberately. Solve deeply. Improve continuously.

Section 1: Python Basics

This section builds a strong foundation in core Python concepts required for everyday programming.
It focuses on syntax, data types, control flow, functions, and basic file handling.
The problems emphasize correctness, clarity, and understanding how Python behaves at a fundamental level.
Ideal for beginners and as a refresher for interview preparation.

Q1. Display Python Version

Difficulty Level: ⭐ Very Easy

Context Tags: Interview, Scripting

Problem Statement:
Develop a Python program that identifies and displays the version of Python installed on the system. The output should be clean, readable, and rely only on the standard library.

Solution Guideline:
Use the sys module to access version information.

Q2. Type Conversion and Addition

Difficulty Level: ⭐ Very Easy

Context Tags: Interview, Scripting

Problem Statement:
Create one integer and one numeric string, convert the string into an integer, and perform addition. Clearly display the result.

Solution Guideline:
Apply explicit type casting before arithmetic operations.

Q3. String Slicing

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Scripting

Problem Statement:
Accept a string input and extract the last three characters safely. The program should handle short strings without errors.

Solution Guideline:
Validate input length and apply slicing.

Q4. List Operations

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Backend

Problem Statement:
Create a list of items, append a new element, sort it alphabetically, and display the final list. Ensure clarity in output.

Solution Guideline:
Use append() and sort() methods.

Q5. Mutable vs Immutable

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview, Backend

Problem Statement:
Demonstrate the difference between mutable and immutable data types by attempting controlled modifications. Observe and explain the behavior.

Solution Guideline:
Modify a list and attempt modification on a tuple.

Q6. Dictionary Traversal

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Backend

Problem Statement:
Create a dictionary storing structured data and display its keys and values separately. Maintain clear distinction in output.

Solution Guideline:
Use keys() and values() methods.

Q7. Set Operations

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Data

Problem Statement:
Compute and display the union and intersection of two predefined sets. Ensure correctness and readability.

Solution Guideline:
Use set operators or built-in methods.

Q8. Even or Odd Check

Difficulty Level: ⭐ Very Easy

Context Tags: Interview, Scripting

Problem Statement:
Determine whether a given integer is even or odd and display an appropriate message.

Solution Guideline:
Use conditional logic with modulus operator.

Q9. Multiplication Table

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Scripting

Problem Statement:
Generate a multiplication table for a given number from 1 to 10 using loops.

Solution Guideline:
Use for loop with formatted output.

Q10. Circle Area Function

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Backend

Problem Statement:
Define a function that calculates the area of a circle using a default radius value. The function should return the computed result.

Solution Guideline:
Use default parameters and math constants.

Q11. Recursive Factorial

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Implement a recursive function to compute factorial of a number with proper base conditions.

Solution Guideline:
Define base and recursive cases correctly.

Q12. Basic File Handling

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend, Scripting

Problem Statement:
Write data to a text file and read it back while ensuring proper resource management.

Solution Guideline:
Use correct file modes and safe handling.

Section 2: Object-Oriented Programming

This section introduces object-oriented design principles using Python.
It covers class creation, inheritance, encapsulation, polymorphism, and special methods.
Problems are structured to reflect real-world modeling scenarios and interview-style OOP questions.
Essential for understanding scalable and maintainable Python applications.

Q1. Employee Class

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Backend

Problem Statement:
Design a class that represents an employee entity in an organization. The class should store essential employee attributes and support object instantiation. Demonstrate object creation and display employee details clearly.

Solution Guideline:
Use __init__ for initialization and instance attributes.

Q2. Calculator Class

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview

Problem Statement:
Create a calculator class that accepts a numeric value at initialization. The class should expose a method that computes and returns the square of the value. Ensure clean separation of responsibilities.

Solution Guideline:
Store value in constructor and implement computation method.

Q3. Method Overriding

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Backend

Problem Statement:
Implement inheritance where a child class overrides a method defined in its parent. This demonstrates runtime polymorphism and method resolution behavior.

Solution Guideline:
Define and override methods appropriately.

Q4. Multilevel Inheritance

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview, Backend

Problem Statement:
Create a multilevel inheritance hierarchy where each class adds functionality while reusing behavior from its parent classes.

Solution Guideline:
Use inheritance and super().

Q5. Static Method

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview

Problem Statement:
Design a class containing a utility method that does not depend on instance data. The method should be callable without creating an object.

Solution Guideline:
Use @staticmethod decorator.

Q6. Operator Overloading

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Enable addition for custom objects by defining how two instances should combine logically. This requires operator overloading using special methods.

Solution Guideline:
Implement __add__().

Q7. Property Decorator

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Restrict direct access to a class attribute and expose it through controlled getter and setter methods using property decorators.

Solution Guideline:
Use @property and setter decorators.

Q8. Class Method Counter

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Track the number of instances created from a class using a class-level counter. Update the counter automatically upon object creation.

Solution Guideline:
Use class variable and @classmethod.

Section 3: Advanced Python

This section focuses on advanced language features and functional programming concepts.
Topics include exception handling, comprehensions, lambda functions, iterators, generators, and virtual environments.
The problems encourage writing concise, efficient, and Pythonic code.
Suitable for learners transitioning from basic scripting to professional-level Python usage.

Q1. Exception Handling

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview, Backend

Problem Statement:
Write a Python program that performs arithmetic operations based on user input. The program must handle invalid input types and runtime errors gracefully. It should continue execution without crashing while providing meaningful error messages.

Solution Guideline:
Use structured try-except blocks with specific exception handling.

Q2. Enumerate Function

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview

Problem Statement:
Iterate over a sequence while accessing both index and value simultaneously. The index should begin from one instead of the default zero. Ensure the output is clearly formatted.

Solution Guideline:
Use enumerate() with start parameter.

Q3. List Comprehension

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview, Data

Problem Statement:
Generate a list of filtered values from a numeric range using list comprehension. This problem emphasizes concise syntax and efficient data transformation.

Solution Guideline:
Apply conditional logic inside the comprehension.

Q4. Lambda Function

Difficulty Level: ⭐ Very Easy

Context Tags: Interview

Problem Statement:
Define an anonymous function to perform a simple computation. Demonstrate how lambda expressions can replace small helper functions.

Solution Guideline:
Create and invoke a lambda expression.

Q5. Filter Function

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Data

Problem Statement:
Extract elements from a list that satisfy a given condition using functional programming tools. This approach avoids explicit loops and improves readability.

Solution Guideline:
Use filter() with a predicate function.

Q6. Map Function

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Data

Problem Statement:
Transform a list of values by applying a conversion formula to each element. This demonstrates functional-style data processing.

Solution Guideline:
Use map() with a conversion function.

Q7. Reduce Function

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview, Data

Problem Statement:
Aggregate multiple values in a list into a single result using a reduction operation. The task focuses on cumulative computation logic.

Solution Guideline:
Use functools.reduce() with comparison logic.

Q8. String Join

Difficulty Level: ⭐ Very Easy

Context Tags: Interview

Problem Statement:
Combine multiple string elements into a single formatted string using a delimiter. This problem emphasizes efficient string handling.

Solution Guideline:
Use the join() string method.

Q9. Virtual Environment

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend

Problem Statement:
Explain the purpose of Python virtual environments and how they isolate dependencies. Also describe how to generate a dependency file for reproducibility.

Solution Guideline:
Use venv creation steps and pip freeze.

Q10. Flask Application

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend

Problem Statement:
Develop a minimal Flask web application that responds on the home route. This introduces basic backend routing and application structure.

Solution Guideline:
Define routes and run the Flask development server.

Section 4: Professional Python Challenges

This section targets real-world and production-level Python skills.
It includes concurrency, asynchronous programming, database interaction, APIs, testing, logging, and design patterns.
Problems are designed to simulate industry scenarios and advanced interview challenges.
Best suited for experienced learners aiming for backend, data, or senior Python roles.

Q1. Execution Time Decorator

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Interview, Backend

Problem Statement:
Create a decorator that measures and logs the execution time of any function it wraps. The decorator should be reusable and transparent to the original function.

Solution Guideline:
Wrap the function and calculate execution duration.

Q2. Fibonacci Generator

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Generate Fibonacci numbers up to a given limit using a generator function. This ensures efficient memory usage compared to list-based solutions.

Solution Guideline:
Use yield inside an iterative structure.

Q3. Custom Context Manager

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Implement a custom context manager to safely manage resources such as files. Ensure resources are released properly even when exceptions occur.

Solution Guideline:
Define __enter__ and __exit__ methods.

Q4. Multithreading

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Demonstrate concurrent execution by running multiple threads performing independent tasks. This problem highlights thread-based parallelism and synchronization concepts.

Solution Guideline:
Use threading module.

Q5. Multiprocessing

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Execute CPU-bound tasks in parallel using multiple processes. This demonstrates performance improvement through process-based concurrency.

Solution Guideline:
Use the multiprocessing module.

Q6. Asynchronous Programming

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Write non-blocking asynchronous code to handle concurrent operations efficiently. This introduces async and await concepts in Python.

Solution Guideline:
Use asyncio with async/await syntax.

Q7. SQLite Database

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend, Data

Problem Statement:
Create and interact with a SQLite database using Python. Perform table creation, insertion, and querying operations.

Solution Guideline:
Use the sqlite3 module.

Q8. API Request

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend, Data

Problem Statement:
Fetch data from an external API endpoint and parse the JSON response. This demonstrates real-world API integration and data handling.

Solution Guideline:
Use the requests library.

Q9. Unit Testing

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Write automated unit tests to verify the correctness of a Python function. This encourages test-driven development practices.

Solution Guideline:
Use pytest and assertions.

Q10. Logging

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Implement logging to record application behavior, warnings, and runtime errors. This is critical for debugging and monitoring production systems.

Solution Guideline:
Configure logging module.

Q11. Regex Email Extraction

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Data

Problem Statement:
Extract email addresses from unstructured text using regular expressions. This problem focuses on pattern matching and text processing.

Solution Guideline:
Use re.findall() with a regex pattern.

Q12. Serialization

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend, Data

Problem Statement:
Serialize and deserialize Python objects for storage or transmission. Compare structured and binary serialization approaches.

Solution Guideline:
Use JSON and Pickle modules.

Q13. Flatten Nested List

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview, Data

Problem Statement:
Convert a nested list structure into a single flat list. This tests understanding of iteration and recursion.

Solution Guideline:
Use list comprehension or recursion.

Q14. Invert Dictionary

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview

Problem Statement:
Create a new dictionary by swapping keys and values from an existing dictionary. This demonstrates dictionary manipulation techniques.

Solution Guideline:
Use dictionary comprehension.

Q15. Combinations

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Interview

Problem Statement:
Generate all unique two-element combinations from a list without repetition. This problem focuses on combinatorial logic.

Solution Guideline:
Use itertools.combinations().

Q16. Type Hints

Difficulty Level: ⭐ ⭐ Easy

Context Tags: Interview

Problem Statement:
Add type annotations to function definitions to improve readability and tooling support. This enhances maintainability and static analysis.

Solution Guideline:
Apply Python type hint syntax.

Q17. Dataclass

Difficulty Level: ⭐ ⭐ ⭐ Medium

Context Tags: Backend

Problem Statement:
Define a data-centric class using dataclasses to minimize boilerplate code. This introduces modern Python design patterns.

Solution Guideline:
Use the @dataclass decorator.

Q18. Singleton Pattern

Difficulty Level: ⭐ ⭐ ⭐ ⭐ ⭐ Very Hard

Context Tags: Interview, Backend

Problem Statement:
Ensure that only one instance of a class exists throughout the program lifecycle. This problem focuses on correct implementation of design patterns.

Solution Guideline:
Use a metaclass-based Singleton approach.

Q19. Large File Processing

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Backend

Problem Statement:
Process very large files efficiently without loading them entirely into memory. This demonstrates memory-efficient programming practices.

Solution Guideline:
Use generators and lazy iteration.

Q20. Frequency Optimization

Difficulty Level: ⭐ ⭐ ⭐ ⭐ Hard

Context Tags: Interview, Data

Problem Statement:
Optimize a solution to identify the most frequently occurring element in a dataset. Focus on reducing time and space complexity.

Solution Guideline:
Use hashing or collections.Counter.