Competitive Programming Roadmap (Python, 16 Weeks)

Language: Python
Daily Time Commitment: 30 minutes
Total Duration: ~16 weeks (4 months)
Outcome: Strong DSA foundation, improved problem-solving speed, contest readiness

Daily 30–Minute Practice Structure (Mandatory)

Every day follows the same discipline-focused routine:

Time	Activity
5 min	Review concept / pattern
20 min	Solve 1–2 problems
5 min	Optimize + note mistakes

Discipline matters more than the number of problems solved.

Phase 0 – Setup & Mindset (Day 1–2)

Goals:

- Set up repository for tracking progress
- Learn effective practice techniques

Topics:

- Python fast I/O
- Time complexity basics
- Reading constraints

Deliverables:

- Repo structure ready
- First problems solved

Phase 1 – Core Programming Basics (Week 1–2)

Focus:
Build fluency in Python fundamentals and simple problem solving.

Topics:

- Input / Output
- Conditions
- Loops
- Functions
- Basic math
- String operations

Problem Types:

- Number manipulation
- Conditional logic
- Pattern printing
- Simple loops

Target:

- 20–25 problems
- No hesitation in writing Python code

Phase 2 – Arrays & Strings (Week 3–4)

Focus:
Develop understanding of sequence data processing.

Topics:

- Arrays (lists)
- Prefix sums
- Two pointers
- Frequency counting
- String traversal
- Palindromes

Problem Types:

- Subarray problems
- Frequency maps
- Simple optimizations

Target:

- Prefer O(n) approaches over brute force
- Recognize common patterns quickly

Phase 3 – Searching & Sorting (Week 5)

Focus:
Understand algorithmic efficiency and optimal retrieval.

Topics:

- Sorting algorithms (conceptual)
- Binary search
- Custom sorting
- Greedy thinking

Problem Types:

- Min/Max queries
- Binary search on answer
- Greedy decisions

Phase 4 – Hashing & Sets (Week 6)

Focus:
Use extra space to optimize time.

Topics:

- Hash maps (dict)
- Sets
- Frequency tables
- Collision ideas (conceptual)

Problem Types:

- Pair sums
- Unique elements
- Counting problems

Phase 5 – Recursion & Backtracking (Week 7)

Focus:
Think in terms of states, choices, and branching.

Topics:

- Recursion basics
- Call stack behavior
- Backtracking
- Subsets / permutations

Problem Types:

- Generate combinations
- Decision trees

Phase 6 – Stacks & Queues (Week 8)

Focus:
Master monotonic structures and simulation.

Topics:

- Stack operations
- Queue & deque
- Parentheses validation
- Next Greater Element

Phase 7 – Linked Lists (Week 9)

Focus:
Understand pointer-like manipulation in Python.

Topics:

- Singly linked list concepts
- Reversal techniques
- Cycle detection

Phase 8 – Trees & Binary Trees (Week 10–11)

Focus:
Work with hierarchical data.

Topics:

- Tree traversals
- DFS / BFS
- Height & diameter calculation
- Binary Search Tree (BST) basics

Phase 9 – Heaps & Priority Queues (Week 12)

Focus:
Efficient min/max retrieval using priority queues.

Topics:

- Heap basics
- `heapq` in Python
- Scheduling and priority problems

Phase 10 – Graph Basics (Week 13–14)

Focus:
Connectivity, traversal, and graph representation.

Topics:

- Graph representation
- BFS / DFS
- Connected components
- Intro to shortest paths

Phase 11 – Dynamic Programming (Week 15)

Focus:
Identify optimal substructure and overlapping subproblems.

Topics:

- 1D DP
- 2D DP
- Memoization vs Tabulation

Phase 12 – Contest Practice & Speed (Week 16)

Focus:
Apply knowledge under timed constraints.

Activities:

- Virtual contests
- Mixed topic problems
- Editorial study and analysis

REPO STRUCTURE (RECOMMENDED)

```
competitive-programming/
│
├── README.md
├── basics/
├── arrays_strings/
├── searching_sorting/
├── hashing/
├── recursion/
├── stacks_queues/
├── linked_list/
├── trees/
├── heaps/
├── graphs/
├── dp/
└── contests/
```