■ Comprehensive Programming Study Syllabus ■

A 48-Week Journey to Master Computer Science Fundamentals

"The only way to learn a new programming language is by writing programs in it." — Dennis Ritchie

■ Goal

To master Python programming in the context of data structures and algorithms, with emphasis on computer science fundamentals, systems programming (C), and mathematical foundations for computational problem-solving.

■ Methodology

• **Build a Portfolio:** Construct a comprehensive GitHub repository filled with high-value projects demonstrating mastery across all three tracks. • **Train on LeetCode:** Systematic practice on algorithmic problems to sharpen problem-solving skills. • **Read the Classics:** Engage deeply with the canonical texts of computer science.

■ How to Make the Most of This Syllabus

Equipment: A reliable laptop (Unix-like OS preferred), fast internet, your favorite IDE, and a sturdy wall for debugging frustrations. ■ **Time Commitment:** 3 days/week, 3-4 hours per session. Consistency beats intensity. **Three-Track System:** Each week alternates between Python (algorithms), C/Systems (low-level), and Design/Math (theory). **Project-First:** Every day includes a hands-on project—build your GitHub portfolio!

■ Words of Wisdom

"Programs must be written for people to read, and only incidentally for machines to execute." — *Abelson & Sussman* | "Talk is cheap. Show me the code." — *Linus Torvalds* | "Premature optimization is the root of all evil." — *Donald Knuth*

■ Required Books for This Syllabus

The following books are referenced throughout the 48-week curriculum. The abbreviations shown are used in the daily reading assignments.

CLRS — *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein

K&R; — *The C Programming Language* by Kernighan and Ritchie

Automate — *Automate the Boring Stuff with Python* by Al Sweigart

Beej — *Beej's Guide to Network Programming* by Brian Hall (free online)

CSAPP — *Computer Systems: A Programmer's Perspective* by Bryant and O'Hallaron

OSTEP — *Operating Systems: Three Easy Pieces* by Arpaci-Dusseau (free online)

Sipser — *Introduction to the Theory of Computation* by Michael Sipser

Dragon Book — *Compilers: Principles, Techniques, and Tools* by Aho et al.

Gang of Four — *Design Patterns* by Gamma, Helm, Johnson, and Vlissides

CP Handbook — *Competitive Programmer's Handbook* by Antti Laaksonen (free PDF)

Plus specific chapters from: Elements of Programming Interviews, Python Official Tutorial, Valgrind Manual, GNU Make Manual, Python for Data Analysis, High Performance Python, Hands-On ML, CUDA by Example, Building Microservices, Designing Data-Intensive Applications, Linux Kernel Development, Linux Device Drivers, Art of Multiprocessor Programming, Making Embedded Systems, Real-Time Systems, Distributed Systems (Tanenbaum), Algorithmic Game Theory, and Elements of Information Theory.

Now, let's begin. Your future self will thank you. ■

■ Table of Contents: 48-Week Curriculum Overview

PHASE 1: FOUNDATIONS (Weeks 1-12)

• Weeks 1-4: Python Basics, C Introduction, Algorithm Analysis, Recursion, Sorting

• Weeks 5-8: Data Structures Basics (Lists, Dicts, Heaps, Hash Tables, Graphs), Memory Management

• Weeks 9-12: OOP Fundamentals, File I/O, Binary Search, Greedy Algorithms, Dynamic Programming Intro

PHASE 2: INTERMEDIATE (Weeks 13-24)

• Weeks 13-16: Linked Lists, Binary Trees, BSTs, System Calls (fork/exec), Red-Black Trees

• Weeks 17-20: Graph Algorithms (BFS/DFS), Shortest Paths (Dijkstra, Bellman-Ford), Sockets, Dynamic Programming

• Weeks 21-24: 2D DP, Backtracking, Tries, Union-Find, MST, Operating Systems Concepts

PHASE 3: ADVANCED (Weeks 25-36)

• Weeks 25-28: Advanced DP (Knapsack), Segment Trees, Concurrency, Network Flow, Virtual Memory

• Weeks 29-32: String Algorithms (KMP), File Systems, Topological Sort, Branch & Bound, Profiling

• Weeks 33-36: Sliding Window, Assembly, Computational Geometry, Bit Manipulation, Automata Theory

PHASE 4: MASTERY (Weeks 37-48)

• Weeks 37-40: Design Patterns, Kernel Modules, Turing Machines, asyncio, Decidability

• Weeks 41-44: Microservices, HPC, Complexity Classes, ML Fundamentals, GPU Programming

• Weeks 45-48: Data Engineering, Distributed Systems, System Design, Capstone Project

Daily Structure: Each day includes a specific Topic, Reading Assignment, LeetCode Problem (where applicable), and a hands-on Project. Print this syllabus and track your progress week by week.

■■■ Week 1 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
1	**Python**	Python Basics: Variables, Types, Operators	Automate Ch.1-2	Two Sum, Add Two Numbers	Build a temperature converter (C/F/K) with input validation
2	**C/System**	C Introduction: Compilation, Basic I/O	K&R; Ch.1, Beej §1-2	Plus One (in C)	Create 'Hello World' with command-line arguments parser
3	**Design/Math**	Algorithm Analysis: Big-O Notation	CLRS §3.1-3.2	Valid Palindrome, Is Subsequence	Benchmark sorting algorithms and plot runtimes vs input size

■■■ Week 2 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
4	**Python**	Control Flow: if/elif/else, while loops	Automate Ch.2	Reverse Integer, Palindrome Number	Number guessing game with difficulty levels and hints
5	**C/System**	Data Types & Operators in C	K&R; Ch.2	Factorial (in C), Power of Two	Build sizeof() demonstration showing all C data type sizes
6	**Design/Math**	Proof Techniques: Induction	Sipser §0.1-0.2	Roman to Integer, Integer to Roman	Prove loop invariants for 3 simple algorithms (max, sum, reverse)

■■■ Week 3 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
7	**Python**	Lists & List Operations	Automate Ch.4, CLRS §10.1	Remove Duplicates, Remove Element	Implement todo list manager with add/remove/search/filter
8	**C/System**	Arrays & Strings in C	K&R; Ch.5.1-5.5	Reverse String (in C), Valid Anagram	Create string library: strlen, strcpy, strcat, strcmp from scratch
9	**Design/Math**	Recursion Basics	CLRS §4.1-4.3	Fibonacci, Climbing Stairs, Pow(x,n)	Visualize recursion tree for Fibonacci and Tower of Hanoi

■■■ Week 4 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
10	**Python**	Functions & Scope	Automate Ch.3	Power of Three, Happy Number	Build math library with functions for primes, factorials, GCD/LCM
11	**C/System**	Functions & Call Stack	K&R; Ch.4, Beej §3	Factorial Trailing Zeroes, Count Primes	Recursive factorial with manual stack trace output at each level
12	**Design/Math**	Divide & Conquer	CLRS §4.4-4.5	Merge Two Sorted Lists, Sort List	Implement merge sort and quicksort with comparison counter

■■■ Week 5 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
13	**Python**	Dictionaries & Sets	Automate Ch.5	Contains Duplicate, Single Number	Word frequency analyzer for text files with top-N display
14	**C/System**	Pointers Fundamentals	K&R; Ch.5.6-5.10	Reverse Linked List (in C)	Swap function using pointers, pointer arithmetic exercises
15	**Design/Math**	Sorting Algorithms: Comparison	CLRS §6.1-6.4	Sort Colors, Sort Array	Implement bubble, insertion, selection sorts and benchmark

■■■ Week 6 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
16	**Python**	String Manipulation	Automate Ch.6	Valid Anagram, Group Anagrams	Build Caesar cipher encoder/decoder with frequency analysis
17	**C/System**	Dynamic Memory Allocation	K&R; Ch.7.8.5, Beej §4	Merge Sorted Array	Dynamic array implementation with resize/grow functionality
18	**Design/Math**	Heaps & Priority Queues	CLRS §6.5	Kth Largest Element, Last Stone Weight	Min-heap and max-heap from scratch with heapify operations

■■■ Week 7 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
19	**Python**	File I/O & Exception Handling	Automate Ch.8-9	Valid Parentheses, Min Stack	Log parser that extracts errors/warnings from Apache logs
20	**C/System**	File Operations in C	K&R; Ch.7.1-7.6	Read N Characters (simulated)	File copy utility with progress bar (like 'cp' command)
21	**Design/Math**	Hash Tables Theory	CLRS §11.1-11.3	Two Sum II, 3Sum, 4Sum	Implement hash table with chaining and open addressing

■■■ Week 8 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
22	**Python**	Regular Expressions	Automate Ch.7	Implement strStr(), Repeated Substring	Email validator and phone number formatter using regex
23	**C/System**	Structs & Typedef	K&R; Ch.6.1-6.5	Intersection of Two Arrays	Student database with search/sort by name/GPA/ID
24	**Design/Math**	Graph Representations	CLRS §22.1	Find Center, Find Judge	Implement adjacency matrix and adjacency list representations

■■■ Week 9 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
25	**Python**	OOP Basics: Classes & Objects	Python Official Tutorial §9.1-9.5	Design HashSet, Design HashMap	Create library management system with Book/Member/Transaction classes
26	**C/System**	Preprocessor & Macros	K&R; Ch.4.11	Majority Element	DEBUG macro system with file/line/function tracking
27	**Design/Math**	Stack & Queue ADTs	CLRS §10.1	Queue using Stacks, Stack using Queues	Balanced parentheses checker and expression evaluator

■■■ Week 10 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
28	**Python**	Inheritance & Polymorphism	Python Official Tutorial §9.5-9.8	Min Stack, Max Stack	Shape hierarchy (Circle/Rectangle/Triangle) with area/perimeter
29	**C/System**	Memory Debugging: Valgrind	Valgrind Manual Ch.2-3	Linked List Cycle	Fix provided buggy C code with memory leaks using Valgrind
30	**Design/Math**	Binary Search Variants	CLRS §2.3.3	Search Insert Position, First Bad Version	Implement binary search, lower_bound, upper_bound variants

■■■ Week 11 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
31	**Python**	Iterators & Generators	Python Official Tutorial §9.9-9.10	Range Sum Query, Running Sum	Fibonacci generator and prime number generator (infinite)
32	**C/System**	Bitwise Operations	K&R; Ch.2.9	Single Number, Number of 1 Bits	Bit manipulation toolkit: set/clear/toggle/check individual bits
33	**Design/Math**	Greedy Algorithms Intro	CLRS §16.1-16.2	Buy Sell Stock, Jump Game	Activity selection problem and coin change (greedy)

■■■ Week 12 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
34	**Python**	Decorators & Context Managers	Python Official Tutorial §9.11, PEP 343	LRU Cache, Design Browser History	Timing decorator and retry decorator with exponential backoff
35	**C/System**	Make & Build Systems	GNU Make Manual Ch.1-2	Pascal's Triangle	Multi-file C project with Makefile (separate compilation)
36	**Design/Math**	Dynamic Programming Intro	CLRS §15.1-15.2	Climbing Stairs, Min Cost Climbing	Solve Fibonacci with memoization and tabulation comparison

■■■ Week 13 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
37	Python	Linked Lists Implementation	CLRS §10.2	Reverse Linked List, Remove Nth Node	Singly and doubly linked list with full CRUD operations
38	C/System	Linked Lists in C	K&R; Ch.6 (Custom Structures)	Middle of Linked List, Palindrome Linked List	Generic linked list in C using void pointers
39	Design/Math	Amortized Analysis	CLRS §17.1-17.3	Design Dynamic Array	Dynamic array with doubling strategy and amortized cost proof

■■■ Week 14 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
40	Python	Binary Trees Basics	CLRS §12.1-12.2	Max Depth, Min Depth, Invert Tree	Binary tree with preorder/inorder/postorder traversals
41	C/System	Tree Structures in C	K&R; Ch.6 (Structures)	Same Tree, Subtree of Another	Binary tree in C with malloc'd nodes and traversal functions
42	Design/Math	Tree Properties & Theorems	CLRS §12.3	Symmetric Tree, Balanced Binary Tree	Calculate height, count nodes, find diameter of binary tree

■■■ Week 15 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
43	Python	Binary Search Trees	CLRS §12.1-12.3	Validate BST, Kth Smallest in BST	BST with insert/delete/search and in-order successor
44	C/System	Advanced Pointers: Function Pointers	K&R; Ch.5.11	Find Peak Element	Generic sorting function using qsort() with custom comparators
45	Design/Math	Balanced Trees: AVL Intro	CLRS §13.1-13.2	Delete Node in BST	Implement AVL tree rotations (LL, RR, LR, RL)

■■■ Week 16 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
46	Python	Tree Traversals: Iterative	CLRS §12.1	Inorder Traversal, Preorder Traversal	Morris traversal (O(1) space) for inorder traversal
47	C/System	System Calls: fork, exec	Beej §5-6	Generate Parentheses	Mini shell that executes commands with fork/exec (like bash)
48	Design/Math	Red-Black Trees	CLRS §13.3-13.4	BST Iterator	Trace red-black tree insertions and prove black-height property

■■■ Week 17 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
49	**Python**	Heaps & heapq Module	Python Library Ref: heapq module	Top K Frequent, Kth Largest in Stream	Task scheduler using priority queue with deadlines
50	**C/System**	Inter-Process Communication	Beej §7	Task Scheduler	Producer-consumer using pipes between parent/child processes
51	**Design/Math**	Graph Traversal: BFS	CLRS §22.2	Level Order, Zigzag Level Order	Shortest path in unweighted graph using BFS

■■■ Week 18 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
52	**Python**	Graph Algorithms: BFS/DFS	CLRS §22.2-22.3	Number of Islands, Clone Graph	Graph class with BFS/DFS and cycle detection
53	**C/System**	Sockets Basics	Beej §8-10	Word Ladder (BFS)	Echo server and client using TCP sockets
54	**Design/Math**	Graph Traversal: DFS	CLRS §22.3	Course Schedule, Course Schedule II	Topological sort and strongly connected components (Tarjan)

■■■ Week 19 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
55	**Python**	Shortest Paths: Dijkstra	CLRS §24.3	Network Delay Time, Path with Max Probability	Dijkstra's algorithm with priority queue implementation
56	**C/System**	Network Programming Patterns	Beej §11-12	Cheapest Flights K Stops	Multi-threaded chat server handling multiple clients
57	**Design/Math**	Single-Source Shortest Paths	CLRS §24.1-24.2	Cheapest Flights, Min Cost to Hire K	Bellman-Ford with negative cycle detection

■■■ Week 20 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
58	**Python**	Dynamic Programming: 1D	CLRS §15.3	House Robber, Decode Ways, Word Break	Solve: coin change, decode ways, word break (1D DP)
59	**C/System**	Memory Management Strategies	CSAPP §9.9-9.11	Coin Change, Perfect Squares	Memory pool allocator (like a simple malloc)
60	**Design/Math**	DP: Optimal Substructure	CLRS §15.3	Longest Increasing Subsequence, Max Subarray	Proof of optimal substructure for LIS and knapsack

■■■ Week 21 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
61	Python	Dynamic Programming: 2D	CLRS §15.4	Unique Paths, Min Path Sum, LCS	Grid DP: unique paths, min path sum, longest common subsequence
62	C/System	Cache Optimization	CSAPP §6	Triangle (min path sum)	Matrix multiplication with cache-friendly access patterns
63	Design/Math	DP: Overlapping Subproblems	CLRS §15.3	Edit Distance, Interleaving String	Compare recursive, memoized, and tabulated DP solutions

■■■ Week 22 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
64	Python	Backtracking Algorithms	CLRS Ch.8 Supplement	Permutations, Combinations, Subsets	Generate all permutations, combinations, and subsets
65	C/System	Compiler Basics: Lexing	Dragon Book §1-3	Letter Combinations Phone, Palindrome Partition	Tokenizer for simple arithmetic expressions
66	Design/Math	Search Space Pruning	CLRS Ch.8 Supplement	N-Queens, N-Queens II	N-Queens with backtracking and constraint checking

■■■ Week 23 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
67	Python	Trie Data Structure	CLRS §12 Extension	Implement Trie, Word Search II	Autocomplete system using trie (like search suggestions)
68	C/System	Compiler: Parsing	Dragon Book §4	Add and Search Word, Replace Words	Recursive descent parser for arithmetic expressions
69	Design/Math	String Algorithms	CLRS §32.1-32.3	Longest Common Prefix, Repeated DNA	Rabin-Karp and KMP string matching algorithms

■■■ Week 24 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
70	Python	Union-Find (Disjoint Set)	CLRS §21.1-21.3	Number Connected Components, Graph Valid Tree	Union-Find with path compression and union by rank
71	C/System	Operating System Concepts	OSTEP Ch.1-5	Accounts Merge, Redundant Connection	Round-robin process scheduler simulation
72	Design/Math	Minimum Spanning Trees	CLRS §23.1-23.2	Min Cost Connect Points, Water/Land	Kruskal's and Prim's MST implementations

■■■ Week 25 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
73	Python	Advanced DP: Knapsack	CLRS §15.3-15.4	Partition Equal Subset, Target Sum	0/1 knapsack, unbounded knapsack, and partition problems
74	C/System	Concurrency: Threads	OSTEP Ch.26-27	Coin Change, Coin Change 2	Multi-threaded web server with thread pool
75	Design/Math	NP-Completeness Intro	CLRS §34.1-34.2	Partition to K Equal Sum	Reduce 3-SAT to Clique problem

■■■ Week 26 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
76	Python	Segment Trees	CP Handbook Ch.9.3	Range Sum Query Mutable, Count Smaller	Segment tree for range queries (sum, min, max)
77	C/System	Thread Synchronization	OSTEP Ch.28-30	Design Bounded Blocking Queue	Dining philosophers problem with mutexes
78	Design/Math	Approximation Algorithms	CLRS §35.1-35.2	Set Cover (greedy)	2-approximation for vertex cover problem

■■■ Week 27 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
79	Python	Binary Indexed Trees (Fenwick)	CP Handbook Ch.9.4	Count Smaller After Self, Reverse Pairs	Fenwick tree for prefix sums and range updates
80	C/System	Lock-Free Programming Basics	Art of Multiprocessor Ch.10	Binary Search Tree Iterator	Lock-free queue using compare-and-swap (CAS)
81	Design/Math	Randomized Algorithms	CLRS §5.1-5.2	Random Pick with Weight, Shuffle Array	Randomized quicksort and Monte Carlo primality test

■■■ Week 28 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
82	Python	Suffix Arrays & LCP	CP Handbook Ch.26	Longest Duplicate Substring, Distinct Subseq	Build suffix array and LCP array for pattern matching
83	C/System	Virtual Memory Concepts	OSTEP Ch.13-16	LRU Cache, LFU Cache	Page replacement simulator (FIFO, LRU, Optimal)
84	Design/Math	Advanced Graph: Network Flow	CLRS §26.1-26.2	Maximum Flow, Min Cut	Ford-Fulkerson max flow with BFS (Edmonds-Karp)

■■■ Week 29 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
85	**Python**	KMP String Matching	CLRS §32.3-32.4	Implement strStr() KMP, Shortest Palindrome	KMP pattern matching with failure function construction
86	**C/System**	File Systems Basics	OSTEP Ch.39-40	Design File System, In-Memory FS	Simple in-memory file system with directories
87	**Design/Math**	Max Flow Min Cut Theorem	CLRS §26.2	Max Bipartite Matching	Bipartite matching using max flow

■■■ Week 30 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
88	**Python**	Topological Sorting	CLRS §22.4	Course Schedule II, Alien Dictionary	Topological sort using Kahn's algorithm and DFS
89	**C/System**	I/O Scheduling	OSTEP Ch.37	Task Scheduler, Reorganize String	Disk scheduling algorithms (FCFS, SSTF, SCAN)
90	**Design/Math**	Strongly Connected Components	CLRS §22.5	Strongly Connected Components, Critical Connections	Kosaraju's algorithm implementation

■■■ Week 31 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
91	**Python**	Advanced Backtracking	CLRS Ch.8 Supplement	Sudoku Solver, Word Search	Sudoku solver with constraint propagation optimization
92	**C/System**	Memory Hierarchies	CSAPP §6	Robot Room Cleaner, Expression Add Operators	Cache behavior simulator for different access patterns
93	**Design/Math**	Branch and Bound	CLRS Ch.35 Supplement	Traveling Salesman (small), Assignment Problem	Traveling salesman with branch and bound

■■■ Week 32 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
94	**Python**	Monotonic Stack/Queue	Elements of Programming Interviews Ch.8	Largest Rectangle, Maximal Rectangle	Next greater element using monotonic stack
95	**C/System**	System Performance Profiling	CSAPP §5.14, perf tutorial	Trapping Rain Water, Container with Water	Profile C program and optimize hot spots
96	**Design/Math**	Computational Geometry Basics	CLRS §33.1-33.2	Convex Hull, Max Points on Line	Graham scan and Jarvis march convex hull algorithms

■■■ Week 33 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
97	**Python**	Sliding Window Technique	Elements of Programming Interviews Ch.5	Min Window Substring, Longest Substring K Distinct	Max/min in sliding window and longest substring problems
98	**C/System**	Assembly Language Basics	CSAPP §3	Permutation in String, Find All Anagrams	Write simple functions in x86-64 assembly
99	**Design/Math**	Sweep Line Algorithms	CLRS §33.3	Meeting Rooms II, Merge Intervals	Interval scheduling and line segment intersection

■■■ Week 34 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
100	**Python**	Two Pointers Technique	Elements of Programming Interviews Ch.5	3Sum, 4Sum, Container With Most Water	Two-pointer problems: 3Sum, container with most water
101	**C/System**	Linking & Loading	CSAPP §7	Trapping Rain Water, Remove Duplicates	Create and use static and shared libraries in C
102	**Design/Math**	Linear Programming Basics	CLRS §29.1-29.2	Max Profit Assignment	Solve LP problems using simplex method (by hand)

■■■ Week 35 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
103	**Python**	Bit Manipulation Mastery	Elements of Programming Interviews Ch.4	Single Number II, Bitwise AND of Range	Count set bits, power of 2, XOR tricks compilation
104	**C/System**	Security: Buffer Overflows	CSAPP §3.10	Sum of Two Integers (bitwise)	Demonstrate buffer overflow and implement protection
105	**Design/Math**	Cryptography Basics	CLRS §31.6-31.7	Power (modular exponentiation)	Implement RSA encryption/decryption

■■■ Week 36 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
106	**Python**	Python Internals: CPython	CPython Source	Custom Sort String, Sort Characters	Explore CPython source: object.c, dictobject.c
107	**C/System**	Advanced C: Undefined Behavior	C Standards Doc	Reverse Bits, Reverse Integer	Identify and fix undefined behavior in sample C code
108	**Design/Math**	Automata Theory	Sipser Ch.1	Regular Expression Matching, Wildcard Matching	Build DFA for regex and simulate on input strings

■■■ Week 37 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
109	Python	Design Patterns: Creational	Gang of Four Ch.3	Design Pattern Implementation, Clone Graph	Implement Singleton, Factory, Builder, Prototype patterns
110	C/System	Linux Kernel Modules	Linux Kernel Development Ch.16-17	Design Underground System	Simple 'Hello World' loadable kernel module
111	Design/Math	Context-Free Grammars	Sipser Ch.2	Valid Parenthesis String, Remove Invalid	Design CFG for balanced parentheses and palindromes

■■■ Week 38 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
112	Python	Design Patterns: Structural	Gang of Four Ch.4	Adapter Pattern, Composite Pattern	Implement Adapter, Proxy, Decorator, Facade patterns
113	C/System	Device Drivers Basics	Linux Device Drivers Ch.1-3	Design Hit Counter, Logger Rate Limiter	Character device driver with read/write operations
114	Design/Math	Pushdown Automata	Sipser Ch.2	Decode String, Basic Calculator II	Design PDA for {a^n b^n | n >= 0}

■■■ Week 39 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
115	Python	Design Patterns: Behavioral	Gang of Four Ch.5	Observer Pattern, Iterator Pattern	Implement Observer, Strategy, Command, State patterns
116	C/System	Real-Time Systems	Real-Time Systems Ch.4-5	Design Parking System, Seat Reservation	Rate monotonic scheduling simulation
117	Design/Math	Turing Machines	Sipser Ch.3	Add Strings, Multiply Strings	Design TM for palindrome recognition

■■■ Week 40 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
118	Python	Concurrency: asyncio	Python Library Ref: asyncio	Async Web Scraper, Parallel Courses	Async web scraper fetching multiple URLs concurrently
119	C/System	Embedded Programming	Making Embedded Systems Ch.1-3	Fizz Buzz Multithreaded, Print in Order	LED blink program for Arduino/Raspberry Pi
120	Design/Math	Decidability	Sipser Ch.4	Is Graph Bipartite, Possible Bipartition	Prove halting problem is undecidable

■■■ Week 41 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
121	Python	Microservices Architecture	Building Microservices Ch.1-4	Design API Gateway, Rate Limiter	Design REST API for e-commerce (products, orders, users)
122	C/System	High-Performance Computing	Parallel Programming Ch.1-2	Parallel Courses II, Build Array Concurrent	Parallel matrix multiplication using OpenMP
123	Design/Math	Reducibility	Sipser Ch.5	Word Ladder, Word Ladder II	Show HALT_TM reduces to EMPTY_TM

■■■ Week 42 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
124	Python	Testing: Unit & Integration	pytest Documentation: Getting Started	Test Suite Design, Valid Parentheses	Write pytest suite with fixtures, mocks, parametrize
125	C/System	Compiler Optimization	Dragon Book §8-9	Basic Calculator, Expression Tree	Implement constant folding optimization pass
126	Design/Math	Complexity Classes	Sipser Ch.7	Subset Sum, Hamiltonian Path	Explore P, NP, NP-Complete relationships

■■■ Week 43 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
127	Python	Performance Optimization	High Performance Python Ch.2-3	Profile & Optimize, LRU Cache	Profile Python code with cProfile and optimize bottlenecks
128	C/System	SIMD Programming	Intel Intrinsics Guide: Getting Started	Range Sum Query 2D, Matrix Block Sum	Vectorized dot product using SSE/AVX intrinsics
129	Design/Math	Advanced Complexity	Sipser Ch.8-9	Longest Palindromic Subsequence	Space complexity and PSPACE examples

■■■ Week 44 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
130	Python	Machine Learning Fundamentals	Hands-On ML Ch.1-2	Linear Regression, K-Means Clustering	Linear regression from scratch and with sklearn
131	C/System	GPU Programming Intro	CUDA by Example Ch.1-4	Matrix Multiplication, Vector Add	Vector addition kernel in CUDA
132	Design/Math	Probability in CS	CLRS Appendix C	Random Pick Index, Linked List Random	Expected runtime of randomized algorithms

■■■ Week 45 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
133	Python	Data Engineering Basics	Python for Data Analysis Ch.5-7	ETL Pipeline, Top K Frequent Words	CSV ETL pipeline: extract, transform, load with pandas
134	C/System	Distributed Systems Basics	Distributed Systems (Tanenbaum) Ch.5	Design Tic-Tac-Toe, Design Snake Game	Implement 2-phase commit protocol simulation
135	Design/Math	Game Theory Basics	Algorithmic Game Theory Ch.1-2	Nim Game, Predict Winner, Stone Game	Solve prisoner's dilemma and Nash equilibrium

■■■ Week 46 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
136	Python	System Design: Scalability	Designing Data-Intensive Apps Ch.1	Design Twitter, Design TinyURL	Design URL shortener with load estimation
137	C/System	Consensus Algorithms	Raft Paper (in Search of Understandable)	Design Leaderboard, Top K Frequent	Raft leader election simulation
138	Design/Math	Information Theory Basics	Elements of Information Theory Ch.2	Huffman Encoding, Data Compression	Calculate entropy and information content

■■■ Week 47 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
139	Python	Capstone Project Planning	N/A	Design comprehensive project combining patterns	Design full-stack app: web scraper + API + dashboard
140	C/System	Capstone Project: C Component	N/A	Optimize critical path with C	Build performance-critical data processor in C
141	Design/Math	Capstone: Algorithm Design	N/A	Analyze and optimize algorithms	Design custom algorithm with complexity analysis

■■■ Week 48 ■■■

Day	Focus	Topic	Reading	LeetCode	Project
142	Python	Capstone: Implementation	N/A	Complete implementation milestone	Complete implementation with documentation and tests
143	C/System	Capstone: Integration & Testing	N/A	Integration testing	Integrate C module with Python using ctypes/CFFI
144	Design/Math	Capstone: Analysis & Presentation	N/A	Performance benchmarks	Performance benchmarks and final presentation

■ Congratulations! You've Completed the Journey!

■ Your Achievement Checklist

If you've made it through all 48 weeks, you've accomplished something remarkable:

- ■ Completed 144 days of structured study across Python, C, and algorithms
- ■ Solved 300+ LeetCode problems spanning all difficulty levels
- ■ Built 144 hands-on projects demonstrating practical mastery
- ■ Read seminal texts: CLRS, K&R, Sipser, CSAPP, and more
- ■ Created a comprehensive GitHub portfolio showcasing your work
- ■ Mastered data structures: arrays, trees, graphs, heaps, tries
- ■ Implemented classic algorithms: sorting, searching, DP, graph traversal
- ■ Learned systems programming: memory management, concurrency, networking
- ■ Understood theoretical foundations: complexity theory, computability
- ■ Completed a capstone project integrating all three tracks

■ What's Next? Your Path Forward

1. Contribute to Open Source: Give back to the community. Find projects on GitHub, fix bugs, add features. **2. Specialize Deeper:** Choose a domain: ML, Distributed Systems, Compilers, Security, Graphics, Databases, or OS. **3. Teach Others:** Write blogs, create tutorials, mentor beginners. Teaching solidifies understanding. **4. Build Something Real:** Create a product people use. Real-world problems teach what textbooks can't. **5. Never Stop Learning:** Technology evolves. Stay curious, stay humble, keep building.

■ Final Reflections

This syllabus was never just about learning Python or C or algorithms. It was about transforming how you think—developing the mental models, problem-solving skills, and persistence that separate great programmers from the rest. You've learned to think recursively, reason about complexity, debug ruthlessly, and build systems that work.

That's what makes you a programmer now. Not the syntax you've memorized, but the way you approach problems. The confidence that no matter how complex the challenge, you can break it down, understand it, and solve it.

"The programmer, like the poet, works only slightly removed from pure thought-stuff. He builds castles in the air, from air, creating by exertion of the imagination." — *Frederick P. Brooks Jr., The Mythical Man-Month*

Now go build something amazing. The world needs what you can create. ■

■ The Essential Canon: Greatest Books in Computer Science

Algorithms & Data Structures

• *Introduction to Algorithms* by Cormen, Leiserson, Rivest, and Stein (CLRS) — The definitive algorithm reference
• *The Art of Computer Programming* by Donald Knuth — The magnum opus of computer science (Volumes 1-4A)
• *Algorithm Design* by Kleinberg and Tardos — Excellent for developing algorithmic intuition

Systems & Programming Languages

• *The C Programming Language* by Kernighan and Ritchie (K&R;) — The bible of C, written by its creators
• *Computer Systems: A Programmer's Perspective* by Bryant and O'Hallaron (CSAPP) — How computers really work
• *Operating Systems: Three Easy Pieces* by Arpaci-Dusseau (OSTEP) — Modern OS concepts, freely available online
• *Structure and Interpretation of Computer Programs* by Abelson and Sussman (SICP) — Mind-expanding programming paradigms
• *Compilers: Principles, Techniques, and Tools* by Aho, Lam, Sethi, and Ullman (Dragon Book) — The compiler bible
• *Computer Networking: A Top-Down Approach* by Kurose and Ross — Comprehensive networking guide

Theory & Mathematics

• *Introduction to the Theory of Computation* by Michael Sipser — Beautiful exploration of computability and complexity
• *Concrete Mathematics* by Graham, Knuth, and Patashnik — The mathematical foundations underlying computer science
• *Gödel, Escher, Bach: An Eternal Golden Braid* by Douglas Hofstadter — A Pulitzer Prize-winning journey through computation, consciousness, and creativity
• *Algorithmic Game Theory* edited by Nisan, Roughgarden, Tardos, and Vazirani — Game theory applications in CS

Software Engineering & Design

• *Design Patterns: Elements of Reusable Object-Oriented Software* by Gamma, Helm, Johnson, and Vlissides (Gang of Four) — The classic on design patterns
• *The Mythical Man-Month* by Frederick P. Brooks Jr. — Timeless wisdom on software project management
• *Code Complete* by Steve McConnell — Comprehensive guide to software construction
• *Clean Code* by Robert C. Martin — A handbook of agile software craftsmanship

Specialized Topics

• *Competitive Programmer's Handbook* by Antti Laaksonen — Advanced data structures and algorithms (free PDF)
• *Elements of Programming Interviews in Python* by Aziz, Lee, and Prakash — Interview preparation with practical problems

- *Python for Data Analysis* by Wes McKinney — Data manipulation with Pandas
- *Hands-On Machine Learning* by Aurélien Géron — Practical ML with Scikit-Learn and TensorFlow
- *CUDA by Example* by Sanders and Kandrot — Introduction to GPU programming
- *High Performance Python* by Gorelick and Ozsvald — Optimization techniques
- *Building Microservices* by Sam Newman — Modern distributed architecture
- *Designing Data-Intensive Applications* by Martin Kleppmann — The big ideas behind reliable, scalable systems
- *Distributed Systems* by Tanenbaum and Van Steen — Principles and paradigms
- *The Art of Multiprocessor Programming* by Herlihy and Shavit — Concurrent programming principles
- *Linux Device Drivers* by Corbet, Rubini, and Kroah-Hartman — Kernel programming
- *Linux Kernel Development* by Robert Love — Understanding the Linux kernel
- *Making Embedded Systems* by Elecia White — Design patterns for great embedded software
- *Real-Time Systems* by Jane W. S. Liu — Real-time and embedded systems
- *Elements of Information Theory* by Cover and Thomas — Information theory fundamentals
- *Parallel Programming* by Pacheco — Introduction to parallel programming with MPI and OpenMP

These books represent the collective wisdom of computer science's greatest minds. They are not just references—they are transformative works that will shape how you think about computing. Read them deeply, work through their exercises, and return to them throughout your career.

— End of Syllabus —