## ****Day 1-30: Language Basics****

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| 1 | Print, Variables, Data Types | Print, Variables, Data Types | Print, Variables, Data Types |
| 2 | Operators (Arithmetic, Logical) | Operators (Arithmetic, Logical) | Operators, Type Inference (let, mut) |
| 3 | Strings, String Methods | Strings, String Operations | Strings (String vs &str) |
| 4 | Lists & Tuples | Arrays, Vectors | Arrays, Vec<T> |
| 5 | Dictionaries & Sets | Maps, Sets (STL) | HashMap, BTreeMap |
| 6 | Conditional Statements (if, elif, else) | If-else statements | Pattern Matching (match) |
| 7 | Loops (for, while) | Loops (for, while) | Loops (for, while, loop) |
| 8 | Functions & Scope | Functions, Function Overloading | Functions, Ownership Basics |
| 9 | Lambda Functions | Function Pointers & Lambdas | Closures |
| 10 | Exception Handling (try-except) | Exception Handling (try-catch) | Error Handling (Result, Option) |
| 11 | Lists & Comprehensions | Vector Manipulation | Iterators, Map & Filter |
| 12 | File Handling (open, read, write) | File I/O (fstream, ifstream) | File I/O (std::fs) |
| 13 | Modules & Imports | Header Files, Namespaces | Modules (mod, use) |
| 14 | Classes & Objects (OOP) | Classes & Objects | Structs & Implementations |
| 15 | Inheritance & Polymorphism | Inheritance, Virtual Functions | Traits |
| 16 | Encapsulation & Abstraction | Encapsulation | Privacy & Modules |
| 17 | Recursion | Recursion | Recursion |
| 18 | Decorators | Templates | Generics |
| 19 | Generators (yield) | Iterators | Iterators & Lazy Evaluation |
| 20 | Threading Basics (threading module) | Multithreading (std::thread) | Async & Await (tokio) |
| 21 | NumPy Basics | STL Basics | Rust Collections |
| 22 | List vs Set vs Dict Performance | Unordered Map vs Map vs Set | HashMap Performance |
| 23 | Memory Management & Garbage Collection | Pointers & References | Ownership & Borrowing |
| 24 | Deep Dive: Mutable & Immutable Variables | Deep Dive: Pointers | Deep Dive: Lifetimes |
| 25 | Built-in Data Types | C++ Standard Data Types | Rust Primitive Data Types |
| 26 | Comprehensions & Functional Programming | Functional C++ (std::function) | Functional Rust (Iterators, Closures) |
| 27 | Debugging & Profiling | Debugging with GDB | Debugging with rust-gdb |
| 28 | File Handling Advanced (CSV, JSON) | File Parsing | CSV & JSON Parsing |
| 29 | Review & Hands-on Exercises | Review & Hands-on Exercises | Review & Hands-on Exercises |
| 30 | Mini Project: CLI To-Do App in all three languages | Mini Project | Mini Project |

## ****Day 31-60: Data Structures & Algorithms (Improved)****

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| 31 | Introduction to Data Structures | Introduction to Data Structures | Introduction to Data Structures |
| 32 | Arrays: Basics & Operations | Arrays & Vectors | Arrays & Vectors |
| 33 | Dynamic Arrays (list) | Dynamic Arrays (vector) | Dynamic Arrays (Vec<T>) |
| 34 | 2D Arrays & Matrix Operations | 2D Arrays & Matrix Operations | 2D Arrays & Matrix Operations |
| 35 | Linked Lists (Singly) | Linked Lists (Singly) | Linked Lists (Singly) |
| 36 | Linked Lists (Doubly) | Linked Lists (Doubly) | Linked Lists (Doubly) |
| 37 | Circular Linked Lists | Circular Linked Lists | Circular Linked Lists |
| 38 | Stack (List, deque) | Stack (stack, deque) | Stack (Vec<T>) |
| 39 | Queue (queue, deque) | Queue (queue, deque) | Queue (VecDeque) |
| 40 | Priority Queue (heapq) | Priority Queue (priority\_queue) | Priority Queue (BinaryHeap<T>) |
| 41 | Hash Tables (dict) | Hash Tables (unordered\_map) | Hash Tables (HashMap) |
| 42 | Sets (set) | Ordered & Unordered Sets | HashSet, BTreeSet |
| 43 | String Matching Algorithms | KMP, Rabin-Karp | Pattern Matching in Rust |
| 44 | Sorting: Bubble, Selection, Insertion | Bubble, Selection, Insertion Sort | Sorting Algorithms |
| 45 | Merge Sort | Merge Sort | Merge Sort |
| 46 | Quick Sort | Quick Sort | Quick Sort |
| 47 | Counting Sort, Radix Sort | Counting Sort, Radix Sort | Counting Sort, Radix Sort |
| 48 | Binary Search | Binary Search | Binary Search |
| 49 | Two-Pointer & Sliding Window | Two-Pointer & Sliding Window | Two-Pointer & Sliding Window |
| 50 | Recursion Basics | Recursion Basics | Recursion Basics |
| 51 | Backtracking (Permutations) | Backtracking (Permutations) | Backtracking (Permutations) |
| 52 | Breadth-First Search (BFS) | BFS | BFS |
| 53 | Depth-First Search (DFS) | DFS | DFS |
| 54 | Graph Representation | Graph Representation | Graph Representation |
| 55 | Dijkstra’s Algorithm | Dijkstra’s Algorithm | Dijkstra’s Algorithm |
| 56 | Floyd-Warshall Algorithm | Floyd-Warshall | Floyd-Warshall |
| 57 | Union-Find (Disjoint Sets) | Union-Find | Union-Find |
| 58 | Trie Data Structure | Trie Implementation | Trie in Rust |
| 59 | Bit Manipulation | Bitwise Operators | Bitwise Operators |
| 60 | Dynamic Programming (Intro) | DP Basics | DP Basics |

# ****Day 61-100: Object-Oriented & Functional Programming (Revised)****

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| 61 | OOP Principles: Encapsulation, Abstraction | OOP Principles | OOP in Rust |
| 62 | Classes & Objects | Classes & Objects | Structs & Implementations |
| 63 | Instance vs Static Methods | Static & Non-static Members | Associated Functions (impl) |
| 64 | Inheritance Basics | Single & Multiple Inheritance | Traits & Trait Bounds |
| 65 | Method Overriding | Virtual Functions | Trait Implementations & Overriding |
| 66 | Polymorphism & Dynamic Dispatch | Function Overloading, Operator Overloading | Dynamic Dispatch (dyn Trait) |
| 67 | Constructors & Destructors | Constructor & Destructor Overloading | Drop Trait for Cleanup |
| 68 | Abstract Classes & Interfaces | Pure Virtual Functions | Traits as Interfaces |
| 69 | Operator Overloading | Operator Overloading | Overloading (Add, Mul, etc.) |
| 70 | Friend Functions & Classes | friend keyword | Module Privacy in Rust |
| 71 | Exception Handling (try-except) | Exception Handling (try-catch) | Error Handling (Result, Option) |
| 72 | Deep Copy vs Shallow Copy | Copy & Move Semantics | Rust Ownership & Move Semantics |
| 73 | Mixins & Multiple Inheritance | CRTP (Curiously Recurring Template Pattern) | Rust Trait Compositions |
| 74 | Template Classes & Generics | Function & Class Templates | Rust Generics |
| 75 | Metaprogramming Basics | Macros & #define | Rust Macros (macro\_rules!) |
| 76 | Functional Programming Basics | Lambda Functions, Map/Filter | Functional C++ (std::function) |
| 77 | First-Class Functions | Function Pointers | Function Traits (Fn, FnMut, FnOnce) |
| 78 | Higher-Order Functions | Function Decorators | Functional Composition |
| 79 | Partial Functions | Partial Application | impl for Partial Functions |
| 80 | Functional Paradigm: Pipelines | Functional STL Algorithms | Rust Iterator Chains (map, filter, fold) |
| 81 | Generators & Coroutines | Iterators & Generators (yield) | Iterators & Lazy Evaluation |
| 82 | Lazy Evaluation | Lazy Evaluation in C++ | Lazy Evaluation in Rust |
| 83 | Referential Transparency | Pure Functions | Pure Functions in Rust |
| 84 | Anonymous Functions & Lambdas | Anonymous & Lambda Functions | Closures |
| 85 | Immutability & Pure Functions | Immutable Data Structures | Rust Ownership & Immutability |
| 86 | Function Composition | Function Wrapping | Chaining Function Calls |
| 87 | Currying | Implementing Currying | Currying in Rust |
| 88 | Monads & Functors | Python Generators as Monads | Functors in C++ |
| 89 | Functional State Management | State Monad in Python | Functional Design Patterns |
| 90 | Functional vs OOP | Comparing OOP & FP | OOP vs FP in C++ |
| 91 | Purely Functional Data Structures | Persistent Data Structures | Rust Functional Data Structures |
| 92 | Immutable Collections | Read-Only Collections | Rust's Rc<T> and Arc<T> |
| 93 | Higher-Kinded Types (HKT) | Simulating HKT in Python | HKT in C++ |
| 94 | Functor & Monad Laws | Functional Theory | Functor & Monad Implementations |
| 95 | Category Theory Basics | Intro to Category Theory | Rust & Category Theory |
| 96 | Functional Error Handling | Functional Exception Handling | Rust ? Operator |
| 97 | Pattern Matching in Functional Programming | Pattern Matching in Python | std::variant |
| 98 | Review of OOP Concepts | Review of C++ OOP Concepts | Review of Rust OOP Concepts |
| 99 | Review of Functional Concepts | Review of C++ Functional Concepts | Review of Rust Functional Concepts |
| 100 | Hands-on Exercises & Practice | Hands-on Exercises & Practice | Hands-on Exercises & Practice |

# ****Day 101-140: Concurrency & Multi-threading****

| Introduction to Concurrency |
| --- |
| Creating & Joining Threads |
| Shared Data & Race Conditions |
| Deadlocks & Avoidance |
| Semaphores |
| Atomic Operations |
| Producer-Consumer Pattern |
| Message Passing Concurrency |
| Async Programming |
| Parallel Processing |

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| **Basic Threading & Parallelism** |  |  |  |
| 101 | Intro to Concurrency & Multithreading | threading module basics | std::thread basics |
| 102 | Creating & Joining Threads | threading.Thread() | std::thread with join() & detach() |
| 103 | Thread Safety & Race Conditions | Race conditions & GIL | Data races in C++ |
| 104 | Synchronization: Locks & Mutex | threading.Lock() | std::mutex |
| 105 | Synchronization: Reentrant Locks | threading.RLock() | std::recursive\_mutex |
| 106 | Synchronization: Condition Variables | threading.Condition() | std::condition\_variable |
| 107 | Reader-Writer Locks | threading.RLock() | std::shared\_mutex |
| 108 | Thread Pools & Work Queues | concurrent.futures.ThreadPoolExecutor | std::thread::hardware\_concurrency(), Thread pools |
| 109 | Producer-Consumer Model | queue.Queue(), Queue.put(), Queue.get() | std::queue & condition variables |
| 110 | Deadlocks: Causes & Prevention | Thread contention & order | Deadlocks & std::lock\_guard |
| **Asynchronous Programming** |  |  |  |
| 111 | Basics of Async Programming | asyncio basics | std::future, std::promise, std::async |
| 112 | Creating Async Functions | async def & await | std::async & std::future |
| 113 | Event Loop Execution | asyncio.run() | Custom event loops |
| 114 | Async Tasks & Futures | asyncio.create\_task() | std::future & std::promise |
| 115 | Handling Multiple Async Tasks | asyncio.gather() | std::when\_all() |
| 116 | Async File Handling | aiofiles module | std::async with I/O |
| 117 | Async Networking | asyncio.open\_connection() | boost::asio |
| 118 | Async Message Passing | asyncio.Queue() | std::future message passing |
| 119 | Cancellation of Async Tasks | asyncio.CancelledError | std::future::cancel() |
| **Parallel Computing & Performance Optimization** |  |  |  |
| 120 | Multiprocessing Basics | multiprocessing module | std::thread::hardware\_concurrency() |
| 121 | Shared Memory & IPC | multiprocessing.shared\_memory | shm\_open(), shm\_unlink() |
| 122 | Fork vs Threads | os.fork() | pthread\_create() |
| 123 | Parallel Execution with MapReduce | multiprocessing.Pool.map() | OpenMP & TBB |
| 124 | GPU Acceleration | numba.cuda | CUDA, OpenCL |
| **Advanced Concurrency Models** |  |  |  |
| 125 | Actor Model | pykka | CAF (C++ Actor Framework) |
| 126 | Message-Passing Concurrency | multiprocessing.Pipe() | std::queue with std::condition\_variable |
| 127 | Lock-Free Programming | queue.Queue() (Python GIL) | std::atomic & lock-free |
| 128 | Software Transactional Memory (STM) | pystm (experimental) | STM Libraries |
| **Debugging & Profiling Multithreaded Code** |  |  |  |
| 129 | Debugging Threaded Programs | pdb for debugging | gdb thread debugging |
| 130 | Race Condition Detection | threading debugger | ThreadSanitizer (TSAN) |
| 131 | Performance Profiling | cProfile | perf & gprof |
| **Concurrency in Distributed Systems** |  |  |  |
| 132 | Distributed Concurrency Basics | dask.distributed | MPI & C++ Threads |
| 133 | Message Queues & Brokers | RabbitMQ with pika | ZeroMQ |
| 134 | Event-Driven Programming | asyncio & Event Loops | Event-Driven STL |
| 135 | Kubernetes & Container Concurrency | Running async in Docker | Threading inside containers |
| 136 | Cloud-based Concurrency | celery for distributed tasks | AWS Lambda with C++ |
| **Real-World Use Cases** |  |  |  |
| 137 | File Processing with Multithreading | Multi-threaded file parsing | Fast file readers |
| 138 | Database Access & Concurrency | SQLite concurrent reads | Connection pools in C++ |
| 139 | Web Server Optimization | Multi-threaded Flask | High-performance cpp-httplib |
| 140 | Final Review & Hands-on Exercises | Hands-on concurrency problems | Hands-on concurrency problems |

## ****Day 141-170: System Programming & Memory Management****

Garbage Collection

Dynamic Memory Allocation

Smart Pointers

File Descriptors & System Calls

Virtual Memory & Paging

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** | ****Node.js**** |
| --- | --- | --- | --- | --- |
| 141 | Memory Management Basics | Stack vs Heap | Stack vs Heap | V8 Engine Memory Model |
| 142 | Garbage Collection | Python GC (gc module) | Manual Memory Management (new, delete) | Rust Ownership & Borrow Checker |
| 143 | Dynamic Memory Allocation | malloc, ctypes | malloc, free, new, delete | Box<T>, Rc<T>, Arc<T> |
| 144 | Smart Pointers | Python References & sys.getrefcount | unique\_ptr, shared\_ptr | Rc<T>, Arc<T> |
| 145 | File Descriptors & System Calls | os.open(), os.read() | open(), read(), write() | std::fs::File |
| 146 | Virtual Memory & Paging | mmap | mmap System Calls | mmap in Rust |
| 147 | Process Management | psutil | fork(), exec() | std::process::Command |
| 148 | Signals & Interrupts | Signal Handling (signal module) | signal.h, raise(), kill() | std::os::unix::process::CommandExt |
| 149 | Threads vs Processes | threading vs multiprocessing | std::thread, pthread | std::thread::spawn |
| 150 | Mutex & Synchronization | threading.Lock | std::mutex, std::lock\_guard | Mutex<T> & RwLock<T> |
| 151 | Shared Memory & IPC | multiprocessing.shared\_memory | POSIX Shared Memory | shmem Crate |
| 152 | Pipes & Sockets | socket module | socket.h, send/recv | tokio::net::TcpStream |
| 153 | Epoll & Event-Driven I/O | selectors module | epoll, poll, select | mio for Async I/O |
| 154 | Asynchronous Programming | asyncio | std::future | async/await, tokio |
| 155 | Debugging & Profiling | cProfile, py-spy | GDB, Valgrind | cargo flamegraph |
| 156 | Stack Overflow & Buffer Overflow | Python Call Stack Analysis | Buffer Overflow Exploits | Stack Overflow Protection |
| 157 | Memory Leaks & Detection | objgraph, memory\_profiler | AddressSanitizer | cargo-tarpaulin |
| 158 | Compilers & Linkers | Python Bytecode (dis module) | Compilation & Linking | cargo check, rustc |
| 159 | ELF & Executables | pyinstaller | ELF File Structure (readelf) | objdump, cargo objdump |
| 160 | System Calls Deep Dive | os.syscalls() | syscall(), ioctl() | nix crate |
| 161 | Assembly Language Basics | Python’s ctypes & Inline Assembly | Inline Assembly (asm) | Inline Assembly in Rust |
| 162 | Performance Optimization | Numba, Cython | -O2, -O3, gprof | cargo build --release |
| 163 | Memory Fragmentation | Python Memory Pool (pymalloc) | Heap Fragmentation & Mitigation | Rust’s Stack vs Heap Management |
| 164 | Low-Level Networking | Raw Sockets | Raw Sockets & Packet Capture (pcap) | tokio::net |
| 165 | Kernel Interactions | os.sched\_yield() | Kernel Modules | Rust for Linux Kernel Development |
| 166 | Filesystem Internals | os.walk(), shutil | inode, statfs() | std::fs::metadata |
| 167 | Thread Pools | concurrent.futures | std::thread::thread\_pool | rayon |
| 168 | SIMD & Vectorization | numpy Vectorization | x86intrin.h | packed\_simd |
| 169 | Real-time System Programming | Python RT Scheduling | Real-time C++ (POSIX RT) | tokio::time |
|  |  |  |  |  |

## ****Day 171-200: Web Development & APIs (Expanded Topics)****

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| 171 | Introduction to Web Servers | Flask Basics | Crow Basics |
| 172 | Request-Response Cycle | HTTP Handling in Flask | HTTP Handling in C++ |
| 173 | Routing & URL Parameters | Routing in Flask | Routing in Crow |
| 174 | Query Parameters & Path Variables | Query & Path Parameters | Handling Queries in C++ |
| 175 | Middleware in Web Apps | Flask Middleware | Middleware in C++ |
| 176 | JSON & Data Serialization | json module | nlohmann/json |
| 177 | REST API Design Principles | RESTful API Basics | REST API in C++ |
| 178 | API Rate Limiting & Throttling | Flask-Limiter | C++ Rate Limiting Strategies |
| 179 | Authentication: Basics | Token-Based Auth | Auth in C++ |
| 180 | JWT Authentication | Flask-JWT | JWT Implementation in C++ |
| 181 | OAuth & OpenID | OAuth in Flask | OAuth Flow in C++ |
| 182 | WebSockets | websockets module | uWebSockets |
| 183 | Real-Time Communication | Flask-SocketIO | WebSockets in C++ |
| 184 | CORS & Security | CORS in Flask | Handling CORS in C++ |
| 185 | API Documentation | Swagger/OpenAPI | OpenAPI for C++ APIs |
| 186 | Server-Sent Events (SSE) | SSE in Flask | SSE in C++ |
| 187 | Background Jobs & Scheduling | Celery & Task Queues | Background Threads in C++ |
| 188 | Web Scraping Basics | requests & BeautifulSoup | Web Scraping with C++ |
| 189 | Caching Strategies | Flask-Caching | Caching in C++ |
| 190 | API Testing | PyTest for APIs | Unit Testing in C++ APIs |
| 191 | CI/CD for APIs | Automating API Deployment | CI/CD Pipelines for C++ APIs |
| 192 | Load Balancing & Scaling | Load Balancing in Flask | Load Balancing Strategies |
| 193 | Reverse Proxying & Nginx | Nginx for Flask | Nginx as Reverse Proxy |
| 194 | Rate Limiting & DDOS Protection | Flask-Limiter Advanced | Rate Limiting in C++ |
| 195 | API Gateway Introduction | Using Kong API Gateway | API Gateway for C++ |
| 196 | GraphQL vs REST | GraphQL with Flask | GraphQL in C++ |
| 197 | Microservices Architecture | Flask with Microservices | C++ Microservices |
| 198 | Logging & Monitoring | Logging APIs in Flask | Logging APIs in C++ |
| 199 | Error Handling & Debugging | API Debugging in Flask | API Debugging in C++ |
| 200 | Summary & Review | Web API Best Practices | Web API Best Practices |

## ****Day 171-210: Web Development & APIs (Expanded)****

| ****Day**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- |
| **🌍 Web Fundamentals** |  |  |  |
| 171 | HTTP Basics (requests) | HTTP Requests (cURL & cpp-httplib) | HTTP Requests (reqwest) |
| 172 | HTTP Status Codes | HTTP Responses | HTTP Response Handling |
| 173 | URL Encoding & Query Params | URL Encoding | URL Parsing (url) |
| 174 | Headers & Cookies | Handling Headers & Cookies | Header Management |
| **🔗 REST API Development** |  |  |  |
| 175 | Creating REST APIs (Flask/FastAPI) | Creating REST APIs (Crow/Pistache) | Creating REST APIs (Axum/Actix) |
| 176 | Routing & Query Parameters | Routing & Query Params | Routing in Rust |
| 177 | Middleware Basics | Middleware in C++ | Middleware in Rust |
| 178 | File Upload & Streaming | File Handling in APIs | File Uploading in Rust |
| **🔒 Security & Authentication** |  |  |  |
| 179 | JWT Authentication | JWT in C++ | JWT in Rust |
| 180 | OAuth2 & API Keys | API Key Management | OAuth2 Implementation |
| 181 | Cross-Origin Resource Sharing (CORS) | CORS Handling | CORS in Rust |
| 182 | Secure HTTP Headers (Helmet) | Secure Headers (cpp-httplib) | Secure Headers (tower-http) |
| 183 | Rate Limiting | Rate Limiting (cpp-httplib) | Rate Limiting in Rust |
| **📡 WebSockets & Real-time Communication** |  |  |  |
| 184 | WebSockets (websockets) | WebSockets (uWebSockets) | WebSockets (tokio-tungstenite) |
| 185 | Streaming & Server-Sent Events (SSE) | SSE Implementation | SSE in Rust |
| **⚡ API Performance & Optimization** |  |  |  |
| 186 | Caching (Redis, Memcached) | Caching API Responses | Rust Caching (moka) |
| 187 | API Compression (gzip, brotli) | Compression in APIs | Compression in Rust |
| 188 | Pagination & Rate Limits | Efficient Pagination | Rust Pagination Strategies |
| 189 | Load Balancing & Scaling | Load Balancing Techniques | Scaling Rust APIs |
| **🔄 API Design Best Practices** |  |  |  |
| 190 | REST vs GraphQL vs gRPC | GraphQL Basics | gRPC in Rust |
| 191 | Versioning APIs | API Versioning Techniques | Rust API Versioning |
| 192 | Throttling & Circuit Breakers | Circuit Breaker Pattern | Circuit Breaker in Rust |
| **🔍 API Testing & Monitoring** |  |  |  |
| 193 | Unit & Integration Testing (PyTest) | C++ API Testing (Catch2) | Rust API Testing (tokio-test) |
| 194 | Load Testing (Locust, JMeter) | Load Testing (ApacheBench) | Load Testing in Rust |
| 195 | Logging & Monitoring | API Monitoring (Prometheus) | API Logging (tracing) |
| **📡 Microservices & Service Discovery** |  |  |  |
| 196 | API Gateways & Reverse Proxy | Nginx API Gateway | API Gateways in Rust |
| 197 | Microservices Architecture | Microservices in C++ | Microservices in Rust |
| 198 | Service Discovery (Consul, Eureka) | Service Discovery in C++ | Service Discovery in Rust |
| **📌 Miscellaneous Topics** |  |  |  |
| 199 | Working with Webhooks | Webhooks in C++ | Webhooks in Rust |
| 200 | GraphQL API Development | GraphQL with C++ (libgraphqlparser) | GraphQL with Rust (juniper) |
| 201 | gRPC Advanced Concepts | gRPC in C++ | gRPC in Rust |
| 202 | API Documentation (Swagger, OpenAPI) | OpenAPI in C++ | OpenAPI in Rust |
| 203 | Reverse Engineering APIs | Reverse Engineering API Calls | Rust API Reverse Engineering |
| 204 | API Gateways & Service Mesh | Envoy Proxy in C++ | Rust Service Mesh |
| 205 | Rate Limiting at Scale | High-Scale Rate Limiting | Rust Rate Limiting Best Practices |
| 206 | Handling Large File Uploads | Chunked File Uploads | Rust Multipart File Handling |
| 207 | Web API Deployment | Dockerizing APIs | Deploying Rust APIs |
| 208 | API Monetization Strategies | API Subscription Models | API Business Strategies |
| 209 | CI/CD for APIs | API Automation Pipelines | API CI/CD in Rust |
| 210 | API Future Trends | Exploring New API Tech | Next-Gen Rust APIs |

# ****Day 210-250: Performance Optimization & Profiling****

| ****Day**** | ****Topic**** | ****Python**** | ****C++**** | ****Rust**** |
| --- | --- | --- | --- | --- |
| 210 | Introduction to Performance Optimization | Identify Bottlenecks | Profiling Basics | Optimization Strategies |
| 211 | Code Execution Profiling | cProfile, line\_profiler | gprof, perf | cargo flamegraph |
| 212 | Function Call Optimization | Function Call Overhead | Inlining & constexpr | Function Monomorphization |
| 213 | Memory Profiling Basics | memory\_profiler, tracemalloc | Valgrind, Heaptrack | dhat, heaptrack |
| 214 | Heap vs Stack Optimization | Object Allocation | Manual Memory Management | Ownership System |
| 215 | Garbage Collection Optimization | gc.disable(), \_\_slots\_\_ | Manual Cleanup (delete) | RAII (Drop Trait) |
| 216 | CPU Caching & Locality | NumPy Optimizations | Cache Efficiency | Struct Field Ordering |
| 217 | SIMD & Vectorization | NumPy, Numba | AVX, SSE | packed\_simd, std::simd |
| 218 | Parallel Execution Basics | multiprocessing | OpenMP, TBB | rayon, tokio |
| 219 | Thread Synchronization | threading.Lock() | std::mutex, std::atomic | std::sync::Mutex, RwLock |
| 220 | Lock-Free Programming | GIL Workarounds | std::atomic | AtomicUsize, crossbeam |
| 221 | Memory-Mapped Files | mmap | mmap, mmap64 | mmap-rs |
| 222 | Zero-Cost Abstractions | Avoiding Python Overhead | Templates, Move Semantics | Traits & Generics |
| 223 | String Performance | ''.join() Optimization | std::string\_view | Cow<str> |
| 224 | Regex Performance | re.compile() Optimization | std::regex\_constants::optimize | regex crate tuning |
| 225 | File I/O Optimization | Buffered I/O (io.BufferedReader) | fstream Optimizations | BufReader |
| 226 | Networking Performance | asyncio | asio | tokio |
| 227 | Debugging Performance Issues | py-spy | gdb, perf | cargo-criterion |
| 228 | Profiling with Flame Graphs | py-spy | FlameGraph, perf | cargo flamegraph |
| 229 | Efficient Use of Caches | functools.lru\_cache | Manual Cache Management | cached crate |
| 230 | Avoiding Memory Fragmentation | Object Pools | Custom Allocators | bumpalo |
| 231 | Improving Cold Start Time | Lazy Imports | Link-Time Optimization | lazy\_static |
| 232 | Link-Time & Whole-Program Optimization | Cython, PyPy | -flto, ThinLTO | cargo opt-level |
| 233 | Kernel-Level Profiling | perf | eBPF, ftrace | perf |
| 234 | Profiling Syscalls & Disk I/O | strace, lsof | strace, iostat | strace |
| 235 | Measuring Network Latency | ping, iperf | tcpdump, wireshark | tokio-console |
| 236 | Distributed System Performance | dask, ray | ZeroMQ, MPI | actix |
| 237 | Benchmarking Code Effectively | timeit, pytest-benchmark | google/benchmark | criterion.rs |
| 238 | Binary Size Reduction | Minifying with PyOxidizer | strip, -Os | cargo-strip |
| 239 | Inlining & Interprocedural Optimization | JIT Compilation | inline, forceinline | #[inline(always)] |
| 240 | Comparing Optimization Strategies | Case Study | Case Study | Case Study |
| 241-250 | **Advanced Performance Topics (Deep Dive into Profiling, Kernel Optimization, and Distributed Performance Analysis)** |  |  |  |