Universal Programming Fundamentals

Core Programming Foundations

- Core Programming Concepts

- Syntax & Semantics
- Execution Flow (Top-Down, Event-Driven, Parallel)
- Expressions & Statements

- Variables & Memory Management

- Variable Scope (Global, Local, Block-level)
- Constants vs. Variables
- Garbage Collection vs. Manual Memory Management
- Mutable vs. Immutable Data

- Data Types & Structures

- Primitive Types (Numbers, Strings, Booleans)
- Composite Types (Lists, Maps, Sets, Tuples)
- Custom Data Structures
- Type Systems (Static vs. Dynamic Typing)

- Control Structures

- Conditional Statements (if-else, switch-case)
- Loops (for, while, do-while)
- Iterators & Generators
- Exception Handling (Try-Catch-Finally)

- Functions & Modularization

- Function Declarations & Expressions
- Recursion & Iterative Processing
- Higher-Order Functions & Callbacks
- Closures & Scope Chains

- Object-Oriented vs. Functional Programming

- Encapsulation, Abstraction, Inheritance, Polymorphism
- Functional Concepts (Pure Functions, First-Class Functions, Currying)
- Composition vs. Inheritance

- Working with Data (I/O Operations)

- File Handling (Read, Write, Append)
- Streams & Buffers
- Networking (APIs, HTTP Requests, WebSockets)
- Database Operations (CRUD, Transactions, Indexing)

- Concurrency & Parallelism

- Multi-threading vs. Multi-processing
- Asynchronous Programming (Callbacks, Promises, Async/Await)
- Event Loops & Schedulers

- Error Handling & Debugging

- Types of Errors (Syntax, Runtime, Logical)
- Debugging Tools & Techniques
- Logging & Monitoring

- Performance & Optimization

- Time & Space Complexity (Big-O Notation)
- Caching & Lazy Loading
- Code Profiling & Benchmarking

- Security Best Practices

- Input Validation & Sanitization
- Secure Authentication & Authorization
- Preventing SQL Injection, XSS, CSRF
- Secure Code Review & Threat Modeling

- Code Organization & Software Engineering Principles

- Modularization & Code Reusability
- Design Patterns (Singleton, Factory, Observer)
- SOLID Principles & Clean Code

- Compilation & Execution Models

- Interpreted vs. Compiled Languages
- JIT (Just-In-Time) Compilation
- Static & Dynamic Linking

- Deployment & Software Lifecycle

- Build Systems & CI/CD
- Containerization (Docker, Kubernetes)
- Version Control & Collaboration (Git)
- Software Testing (Unit, Integration, E2E)

Common Functionalities in All Programming Languages

- Date & Time Handling

- Getting the current date and time
- Formatting dates & timestamps
- Time zones and conversions
- Date arithmetic (adding/subtracting days, months, years)

- Variables & Data Types

- Declaring Variables (Dynamic vs. Static Typing)
- Primitive Data Types (Numbers, Strings, Booleans)
- Complex Data Types (Arrays, Objects, Dictionaries, Sets)
- Mutability vs. Immutability

- Methods & Functions

- Defining and calling functions/methods
- Function parameters & return values
- Default arguments & function overloading
- Anonymous functions (Lambdas, Closures)

- Classes & Object-Oriented Concepts

- Defining classes & creating objects
- Class attributes & methods
- Constructors & destructors
- Inheritance, Polymorphism, and Interfaces

- File Structure & Organization

- Project Structure (Organizing code into modules & packages)
- File Handling (Reading/Writing files)
- Configuration Files & Environment Variables

- Import & Export Mechanisms

- Importing built-in and third-party libraries
- Exporting and reusing modules
- Namespaces & scope resolution

- Running Programs

- Running scripts in an interpreter vs. compiled execution
- Command-line execution vs. GUI-based execution
- Debugging tools & logging

- Building & Deploying Programs

- Compilation vs. Interpretation
- Code Optimization & Minification
- Package managers & dependencies
- Deployment strategies (Local, Cloud, CI/CD)