School Connect Online

SCO INTERNATIONAL CODING OLYMPIAD

CLASS 12 OFFICIAL SYLLABUS

2026-27 • chapter-wise notes, learning outcomes and exam pattern

Designed for schools, teachers, parents and students preparing for senior-level coding Olympiad readiness.

- structured global pathway covering data structures, algorithms, AI/ML, statistics and programming
- chapter-wise notes and learning outcomes for classroom planning and independent study
- aligned with computational thinking, secure coding, data literacy and college-readiness skills

Data Structures	Algorithms	Python	C	Swift
PHP	SQL	AI / ML	Web Apps	Security

PDF-ready • Editable Word document • Official SCO cover-page style • Text-based, not a full-page image

Purpose of the Olympiad

The SCO International Coding Olympiad for Class 12 builds advanced computational thinking, code-reading accuracy, algorithmic reasoning and practical software confidence. The syllabus is designed for school students preparing for college-level programming, applied computer science, data science, AI foundations and safe web development.

Exam Pattern

Detail	Description
Exam Name	SCO International Coding Olympiad
Class / Eligibility	Class 12
Duration	60 minutes
Type of Exam	Objective type MCQ
Number of Questions	50 questions
Sections	Advanced Data Structures; Algorithms and AI Basics; Practical Applications and Programming; Achievers Section
Mode	Online / school-administered as notified by SCO

Chapter-wise Syllabus with Small Notes and Learning Outcomes

No.	Chapter	Small Notes for Learning	Learning Outcomes
1	Advanced Python Programming and Code Tracing	Focuses on lists, indexing, functions, loops, recursion, decorators, generators and reliable output prediction.	Trace Python code accurately, identify syntax/logic errors and explain output using language rules.
2	Algorithmic Complexity and Big-O Reasoning	Builds speed and memory reasoning for loops, nested loops, recursion and common data-processing patterns.	Compare algorithms using time/space complexity and select efficient approaches for given constraints.
3	Arrays, Strings and Memory Models	Covers array indexing, dynamic arrays, string handling, bounds errors and memory-safe thinking.	Detect out-of-bounds, null-termination and memory-management issues in code snippets.
4	Linked Lists and Pointer-based Structures	Introduces singly linked lists, node manipulation, insertion, traversal and edge cases.	Reason about linked-list operations, pointer updates and complexity for insert/search/delete cases.
5	Trees and Binary Search Trees	Covers root/leaf concepts, traversal orders, BST properties and skewed-tree behavior.	Use traversal logic to predict order, explain BST search behavior and identify recursion errors.
6	Graphs, BFS, DFS and Representations	Develops graph literacy using adjacency lists, matrices, traversal and cycle-related reasoning.	Choose suitable representations, trace BFS/DFS and apply graph complexity analysis.
7	Recursion, Dynamic Programming and Memoization	Explains recursion trees, base cases, repeated subproblems and caching strategies.	Identify when memoization improves performance and reason about recursive outputs.
8	Data Handling with NumPy, Pandas and Statistics	Introduces mean, median, standard deviation, grouping, filtering and basic analytics.	Interpret short data-science code and explain computed outputs or summaries.
9	AI/ML Foundations and Model Evaluation	Covers supervised learning, training data, imbalanced classes, accuracy limits, precision, recall and F1.	Evaluate ML outputs responsibly and select appropriate metrics for real-world scenarios.

No.	Chapter	Small Notes for Learning	Learning Outcomes
10	SQL, Databases and Query Optimization	Covers SELECT, WHERE, GROUP BY, HAVING, joins, aggregates, subqueries and indexing.	Read SQL queries, predict results and identify efficient indexing or filtering strategies.
11	Web Development, APIs and Security	Includes HTML/CSS basics, REST methods, JSON responses, prepared statements, JWT validation and rate limiting.	Apply secure web practices and recognize common vulnerabilities in application code.
12	Multi-language Programming: C, Swift and PHP	Builds comparative reasoning across C pointers, Swift optionals/generics and PHP database access.	Analyze language-specific code snippets and choose safe, correct implementations.
13	Advanced Systems and Concurrency	Introduces async tasks, race conditions, lock-free structures, resource pools and distributed-system reasoning at an Olympiad-appropriate level.	Identify concurrency hazards and reason about correctness under shared-state or distributed conditions.
14	Achievers Section: Integrated Problem Solving	Combines algorithms, security, databases, AI and debugging in longer scenario-based questions.	Solve multi-step problems and justify choices using accurate computational reasoning.

Section-wise Weighting Guidance

Section	Topics Covered	Suggested Question Range	Learning Emphasis
Advanced Data Structures	Arrays, strings, linked lists, trees, graphs, complexity	15-18	Core structures, traversal, edge cases and output prediction
Algorithms and AI Basics	Control structures, recursion, statistics, AI/ML, data science	12-15	Algorithmic thinking and model interpretation
Practical Applications and Programming	Python, C, Swift, PHP, SQL, web apps and APIs	10-13	Real-world code reading, debugging and secure coding
Achievers Section	Integrated, longer and higher-order scenarios	8-10	Creative reasoning, optimization and professional judgement

Pedagogical Notes for Schools and Teachers

- Use code tracing before coding: students should predict output, explain each step and then verify by running code.
- Balance language syntax with transferable ideas: arrays, loops, recursion, data structures and security principles apply across languages.
- Assess reasoning, not memorization: include why an option is wrong, especially in code debugging and security questions.
- Use real-world contexts such as APIs, login systems, databases, analytics dashboards and AI model evaluation.
- For Grade 12, include college-readiness depth but keep MCQs clear, deterministic and unambiguous.

Reference Alignment Used for Curriculum Quality

- Computer Science Teachers Association: K-12 Computer Science Standards.
- K-12 Computer Science Framework: computational thinking and CS practices.
- AI4K12: Five Big Ideas in Artificial Intelligence and grade-band progressions.

- NIST NICE Framework: cybersecurity knowledge and skills language for education and workforce readiness.
- OWASP Top 10: secure web application risk awareness and prevention.
- College Board AP Computer Science A: programming, data structures, algorithms and code analysis expectations.