School Connect Online

SCO INTERNATIONAL OLYMPIAD

CLASS 11 OFFICIAL SYLLABUS

SCO International Coding Olympiad | Global Computer Science Readiness Pathway

Designed from Class 11 coding pathways and aligned with computational thinking, algorithms, data, AI literacy and secure software practices.

- academic syllabus for students, teachers and schools globally
- chapter-wise notes with measurable learning outcomes
- balanced coverage of programming, algorithms, data, AI, databases and web development
- PDF-ready layout for website publication and school circulation

Maths	English	Science	Mental Ability	Finance Knowledge
AI	Entrepreneurship	GK	Coding	Life Skills

School Connect Online

SCO International Coding Olympiad - Class 11 Official Syllabus

Class 11 students explore advanced programming topics, data structures, algorithms, application development, introductory AI, data science and secure web-development practices. The syllabus is designed to strengthen computational thinking, multi-language coding confidence and college-ready problem-solving skills.

Exam Format

Details	Description
Exam Name	SCO International Coding Olympiad
Eligibility	Class 11
Duration	60 minutes
Type of Exam	Objective Type
Number of Questions	50 questions
Main Sections	Advanced Programming Concepts; Algorithm Development; Coding Projects; Achievers Section

Global Curriculum Alignment

- Computational thinking: decomposition, abstraction, pattern recognition, algorithm design and evaluation.
- Computer science concepts: data structures, algorithms, programming, data analysis, networks, cybersecurity and impact of computing.
- AI and data literacy: supervised/unsupervised learning concepts, datasets, basic statistics, model evaluation and ethical awareness.
- Practical engineering habits: debugging, testing, secure coding, database safety, performance awareness and documentation.

Section-wise Syllabus Map

Section	Topics Covered	Expected Competence
Advanced Data Structures	Arrays, linked lists, stacks, queues, trees, graphs, hash tables and matrix representations.	Students understand core data-structure properties, choose suitable structures for a problem and trace insert/search/traversal operations.
Algorithms and AI Basics	Control structures, recursion, sorting, searching, greedy methods, dynamic programming, complexity analysis, introductory AI/ML and Python statistics.	Students analyze algorithmic efficiency, trace code, use basic statistical reasoning and interpret introductory machine-learning scenarios.
Practical Applications and Programming	Advanced Python, Swift, C, PHP, SQL, REST/web application logic, database querying, API design and secure coding.	Students connect programming concepts to real-world systems such as dashboards, APIs, e-commerce

		backends and database-driven applications.
Achievers Section	Debugging, code-output prediction, error analysis, optimization, security reasoning and multi-step project case studies.	Students solve complex reasoning problems requiring careful tracing, evidence-based debugging and applied computational judgment.

Chapter-wise Notes and Learning Outcomes

No.	Chapter	Small Notes for Learning	Learning Outcomes
1	Control Structures and Code Tracing	Selection, loops, nested loops and conditions in Python/C-style logic.	Trace code manually, predict output and identify how conditions change program flow.
2	Arrays, Lists and Matrix Handling	Indexing, slicing, list comprehensions, two-dimensional arrays and boundary errors.	Manipulate sequential data, detect off-by-one errors and choose efficient access methods.
3	Linked Lists and Pointers	Node structure, head pointer, insertion, traversal, reversal and memory references.	Explain pointer/reference behavior and analyze linked-list operation complexity.
4	Trees and Graphs	Tree terminology, BST traversal, graph representation, DFS/BFS and adjacency structures.	Interpret hierarchical and network data structures and select suitable traversal strategies.
5	Sorting and Searching Algorithms	Linear/binary search, merge sort, quick sort, heap sort and complexity comparison.	Compare algorithms by time complexity and select an efficient search/sort strategy.
6	Recursion and Dynamic Programming	Recursive functions, memoization, Fibonacci, knapsack and subproblem reuse.	Recognize overlapping subproblems and explain when caching improves performance.
7	Python for Data Science	Pandas, NumPy, filtering, grouping, aggregation, generators and memory-aware processing.	Use data-processing logic to compute summaries and debug type or missing-value problems.
8	Basic Statistics with Python	Mean, median, range, standard deviation, grouped summaries and forecasting basics.	Interpret basic statistical outputs and connect them to data-driven decisions.
9	Introductory AI and Machine Learning	Supervised/unsupervised learning, labeled data, classification, bias and model evaluation.	Identify learning types and reason about training data, imbalance and ethical AI use.
10	Swift Programming	Constants, optionals, closures, protocols, arrays, map/filter and memory management basics.	Read Swift code, safely handle optionals and understand protocol-oriented programming.
11	C Programming	Pointers, arrays, dynamic memory allocation, undefined behavior and memory cleanup.	Trace pointer operations and identify unsafe memory access or missing free operations.

12	PHP and Web Development	Sessions, forms, arrays, headers, database access and server-side performance.	Apply PHP logic to web application scenarios and recognize common implementation issues.
13	SQL and Database Design	SELECT, JOIN, GROUP BY, HAVING, indexes, normalization and window functions.	Write and interpret queries, aggregate grouped data and reason about query performance.
14	Web-based Application Development	REST APIs, HTTP methods/status codes, JSON, backend scaling and caching.	Connect API design choices to real-world web systems and scalable architecture.
15	Secure Coding and Debugging	SQL injection, prepared statements, validation, XSS/CSRF awareness, error analysis.	Identify insecure patterns and choose safer implementation strategies.
16	Achievers Problem Solving	Multi-step code, algorithmic reasoning, case-based debugging and optimization.	Solve higher-order coding problems requiring careful analysis and justification.

Preparation Guidance for Students, Teachers and Schools

Students

Practise tracing code without running it, build small programs for each topic, revise errors and maintain a concept notebook for complexity, syntax, security and data examples.

Teachers

Use the syllabus map to plan weekly micro-lessons, conduct short debugging labs, and connect MCQ practice to real code reviews and mini-projects.

Schools

Run the Olympiad as an academic enrichment programme that supports computer science readiness, computational thinking, AI literacy and responsible digital citizenship.