School Connect Online

SCO INTERNATIONAL CODING OLYMPIAD

CLASS 8 SYLLABUS

Official Syllabus | Programming, Data, Web and Game Logic

Official SCO cover format with academic, ready question layout.

- Designed for Class 8 learners developing programming, debugging, web, data and game logic skills.
- Compact question numbering, clean question blocks, answer key and explanations for website download.
- Aligned to middle-school computational thinking, C++/Kotlin/Python/XML, web basics, game logic and responsible coding practice.

Programming	C++	Kotlin	Python	Turtle	
Scratch	JavaScript	SQL	Game Logic	SCO	

School Connect Online

SCO International Coding Olympiad - Class 8

The Class 8 SCO International Coding Olympiad strengthens programming readiness by combining variables, loops, arrays, conditions, C++ basics, Kotlin basics, Python logic, structured data, XML, SQL, JavaScript DOM awareness, Turtle programming, Scratch/game concepts and debugging. The syllabus is designed for learners transitioning from introductory coding to more rigorous computational reasoning and application-oriented problem solving.

Exam Snapshot

Field	Official Detail
Exam Name	SCO International Coding Olympiad
Class / Eligibility	Class 8
Duration	60 minutes
Type of Exam	Objective Type MCQ
Recommended Questions	50 questions for full official/school practice paper; class-ready papers may be configured as per published exam plan
Core Sections	Programming Basics; Advanced Coding Concepts and Debugging; Application, Game and Data Reasoning; Achievers Section
Syllabus Topics	Variables, loops, conditions, arrays, C++ basics, Kotlin basics, Scratch sprites, Turtle programming, XML, SQL, JavaScript, game loop, collision detection, pathfinding and debugging

Pedagogical Purpose

- Move learners from syntax recognition to reasoning about execution, errors, output and algorithmic correctness.
- Develop debugging habits through purposeful error-analysis questions in Python, C++, Kotlin, JavaScript, PHP and SQL contexts.
- Connect computing to visible outcomes such as webpages, XML data, Turtle drawings, game loops, sprites, collision detection and AI pathfinding.
- Prepare learners for advanced Class 9-12 computing topics including data science, application development, secure coding and AI foundations.

Chapter-wise Syllabus with Small Notes and Learning Outcomes

No.	Chapter	Small Notes for Learning	Learning Outcomes
1	Programming Basics	Covers variables, expressions, loops, conditions, arrays/lists, input-output and basic syntax reading across beginner-friendly code snippets.	Trace simple programs, predict outputs and identify how values change through statements and loops.
2	C++ Programming Foundations	Introduces typed variables, arrays, loops, functions, references, overloading, memory safety concepts and common syntax errors.	Recognize correct C++ syntax, identify out-of-bounds access, understand references and reason about simple C++ outputs.
3	Kotlin Programming Foundations	Introduces var and val, type inference, nullable values, ranges, loops and safe handling of null values.	Differentiate mutable and read-only values, understand nullable types and select safe Kotlin syntax in MCQ contexts.
4	Python Logic, Data Structures and Debugging	Strengthens Python reasoning through lists, dictionaries, lambdas, generators, recursion, identity vs equality and error analysis.	Predict Python outputs, correct common errors and explain why a program fails or produces a particular result.
5	Web and JavaScript DOM Basics	Introduces basic client-side webpage interaction, element selection and safe checks before modifying page content.	Explain how JavaScript interacts with HTML elements and identify null-reference risks in DOM-based code.

No.	Chapter	Small Notes for Learning	Learning Outcomes
6	SQL and Data Queries	Covers SELECT, JOIN, GROUP BY, aggregate functions, subqueries, aliases and query-correction reasoning.	Choose correct query structures and identify aggregation, grouping or ambiguity errors.
7	XML and Structured Data	Introduces XML elements, attributes, root elements, comments, schemas and how structured files store application/game settings.	Read XML snippets, identify valid structure and explain how structured data can support games and applications.
8	Turtle and Visual Programming	Uses Turtle/Scratch-style activities to connect loops, movement, events and shapes to visual programming outcomes.	Predict drawings, identify sprite/event behavior and describe how repeated commands create patterns.
9	Game Development Concepts	Covers game loop, collision detection, random events, level data, high scores, raycasting and basic pathfinding logic.	Explain game-state updating, collision methods and why algorithms such as A* are useful for movement and AI behavior.
10	Achievers Section: Applied Reasoning	Challenges students with multi-step debugging, algorithm choice, secure and efficient programming habits and real-world computing scenarios.	Apply reasoning to unfamiliar snippets, compare possible fixes and justify the most reliable technical choice.

Recommended Assessment Blueprint

Section	Suggested Questions	Focus
Programming Basics	12-15	Variables, loops, arrays/lists, conditions, syntax recognition, output prediction
Advanced Coding Concepts and Debugging	12-15	C++, Kotlin, Python, XML, SQL and error analysis
Application, Game and Data Reasoning	10-12	Game loop, collision detection, Turtle/Scratch, JavaScript DOM, XML/JSON and SQL logic
Achievers Section	8-10	Higher-order debugging, pathfinding, secure coding, algorithm selection and applied case studies

Learning Progression for Class 8

Remember and Understand

Students recognize syntax, language features, tags, commands, selectors, data structures and common terms.

Apply

Students trace code, predict output, choose correct statements and connect programming concepts to webpages or games.

Analyze

Students identify errors such as null access, missing cases, out-of-bounds indexing, invalid grouping and unsafe assumptions.

Create and Extend

Students are encouraged to build simple programs, visual patterns, game logic flows and structured data files for enrichment.

Teacher and School Implementation Notes

- Use short code-tracing warm-ups before MCQ practice so students learn to read code line by line.
- Pair syntax questions with conceptual questions: what the code does, why an error happens, and how to fix it.
- For game-development topics, demonstrate sprite movement, collision, scoring, random events and game loops using simple visual examples.
- For XML/JSON and SQL, show how structured data supports real applications such as games, scoreboards and school records.
- Use the Achievers Section for differentiated practice, coding-club enrichment and high-performing learners.

Global Standard Alignment Snapshot

Reference Area	How this Class 8 syllabus aligns
K-12 Computer Science Framework	Uses computational-thinking practices such as problem decomposition, algorithms, abstraction, testing and communication.
CSTA-style Grade 6-8 Progression	Addresses algorithms and programming, data and analysis, computing systems, impacts of computing and responsible practice.
AI4K12 Readiness	Builds readiness for AI learning through data, patterns, game AI pathfinding, heuristics and responsible technical reasoning.
MDN Web Literacy	Web topics build structured understanding of HTML/DOM/CSS/JavaScript interaction and browser-based application thinking.
OWASP Secure Coding Awareness	SQL/web-security items introduce habits such as parameterized queries and avoiding unsafe direct input use.

Preparation Roadmap for Students

- Week 1: Revise variables, loops, arrays/lists, conditions, output prediction and basic syntax across Python, C++ and Kotlin.
- Week 2: Practice XML, SQL, JavaScript DOM basics, dictionaries, generators, recursion and common runtime errors.
- Week 3: Study Turtle/Scratch, game loops, sprites, collision detection, random events and structured game data.
- Week 4: Practice Achievers-level error analysis, pathfinding, algorithm choice and multi-step case-study questions.

Reference Snapshot

This syllabus is aligned with international K-12 computer science principles and beginner-to-intermediate coding practice. Teachers may support students through browser-based coding tasks, simple Python/C++/Kotlin tracing, XML/SQL examples, Turtle drawings, Scratch-style game events and class discussions on safe, correct and responsible computing.