XV. GIMNAZIJA
International Baccalaureate Department
Diploma Programme

COMPUTER SCIENCE SL
Year 1 and 2

COURSE OUTLINE 2025-2027

WHAT IS THE COURSE ABOUT?

Computer Science is a Group 4 subject in the IB Diploma Programme. It is a practical and problem-solving discipline focused on the principles of computing, the architecture and operation of computer systems, and the design of computational solutions. The course develops students' understanding of fundamental computer science concepts and computational thinking. Students explore how computer systems represent and process data, how devices communicate over networks, how data is structured and managed, and how computational methods solve real-world problems.

Practical application forms a core component of the syllabus. Students design algorithms and build working software solutions using Python. Through hands-on programming, they learn to analyse problems, design structured solutions, implement and test code, and evaluate final outcomes.

Additionally, the curriculum covers databases, machine learning, and artificial intelligence, while encouraging discussion on emerging technological trends. Special emphasis is placed on the ethical, social, economic, and environmental impacts of technology on individuals and society.

Throughout the course, students develop computational thinking skills. This includes the ability to:

- identify and specify a problem in a computational context
- decompose a complex problem into smaller parts
- recognize patterns and use abstraction and generalization to simplify problems and tasks
- design algorithms and computational solutions in generic language
- implement solutions using an appropriate programming language
- test and evaluate solutions
- consider the limitations and possible improvements of a solution.

AIMS

The aims of the Computer Science course are to enable students to:

- develop conceptual understanding and make connections between different areas of computer science and other subjects
- acquire and apply the knowledge, methods, tools and techniques used in computer science
- develop computational thinking and problem-solving skills
- design, implement, test and evaluate computational solutions
- approach unfamiliar problems logically, creatively and with perseverance
- develop practical programming skills
- understand the possibilities and limitations of computer systems
- evaluate the impact of existing and emerging technologies
- communicate ideas, methods and solutions clearly and appropriately
- develop awareness of the environmental, economic, cultural, social and ethical implications of computer science.

OBJECTIVES

Upon completion of the course, students are expected to demonstrate proficiency across four main domains:

1. Knowledge and understanding

Students should be able to:

- demonstrate knowledge and understanding of relevant facts, concepts and principles
- use appropriate computer science terminology
- demonstrate understanding of appropriate methods and techniques
- describe and explain operation and use of computer systems.

2. Application and use

Students should be able to:

- apply computer science concepts and principles in familiar and unfamiliar situations
- use appropriate methods and techniques to solve computational problems
- apply computational thinking to the development of solutions
- use appropriate methods to present information and communicate solutions.

3. Analysis, construction and evaluation

Students should be able to:

- analyze computational problems and proposed solutions
- construct appropriate algorithms and computational solutions
- interpret and evaluate information, algorithms and programs
- develop suitable testing strategies
- evaluate solutions against requirements and expected success criteria
- identify limitations and possible improvements.

4. Computational problem-solving

Students should be able to:

- specify and decompose problems
- design algorithms
- implement solutions in Python
- test and debug programs systematically
- evaluate the effectiveness of computational solutions.

ASSESSMENT:

Knowledge and understanding – 40% of class grade

- **Unit tests** are written after the completion of each major unit and usually last one or two school periods.
- **Quizzes** are shorter assessments, normally given approximately twice per month, depending on the unit. They usually contain short-answer questions and have a working time of approximately 15 minutes. The grade may be based on the average result of two consecutive quizzes.

Tasks include definitions, explanations and questions related to the topics covered in class.

Application of knowledge / Practical work – 20% of class grade

Assessment includes computational thinking and algorithmic tasks, data representation, programming, program execution and testing, completed primarily in Python.

In Year 2, practical work also includes work on the Computational Solution, the internally assessed component of the IB course.

Engagement – 5% of class grade

Engagement includes presentations, short assignments, class activities, collaborative work and participation in lessons.

Semester / Mock Examinations – 35% of class grade

Students write semester and mock exams based on the structure and requirements of IB examination papers.

- **Year 1**: End-of-year examination – 35%
- **Year 2**: Mock Paper 1 – 20% and Mock Paper 2 – 15%

Formative assessment is used throughout the course to monitor students' progress and provide feedback. It may include class exercises, homework, programming tasks, practice examination questions and other activities. Formative assessment does not necessarily result in a grade.

GRADING SCALE

All assessed work is marked using the 1–7 grading scale.

grades	unit tests (%), quizzes	semester exams (%)
7	90-100	86-100
6	78-89	71-85
5	65-77	55-70
4	51-64	45-54
3	39-50	31-44
2	26-38	15-30
1	0-25	0-14

At the end of the academic year, the final class grade is calculated as follows:

Year 1	Year 2
Class grade - 65% End-of-year examination - 35%	Class grade - 65% Mock exam Paper 1 - 20% Mock exam Paper 2 - 15%

The 65% class grade consists of **Knowledge and understanding (40%)**, **Application of knowledge/Practical work (20%)** and **Engagement (5%)**.

FINAL IB ASSESSMENT – STANDARD LEVEL

The final IB assessment consists of:

- **Paper 1 – 35%** (1 hour 15 minutes): Theme A – Concepts of computer science, including the prescribed case study.
- **Paper 2 – 35%** (1 hour 15 minutes): Theme B – Computational thinking and problem-solving, including algorithmic and programming tasks. Students at XV. gimnazija use Python.
- **Computational Solution – 30%**: an individual internally assessed project in which students design, develop, test and evaluate a computational solution to a real-world problem.

The Computational Solution is internally assessed by the teacher and externally moderated by the IB.

IMPLEMENTATION

XV. gimnazija offers the Standard Level (SL) Computer Science course only.

DP Year 1: 4 lessons per week
DP Year 2: 3 lessons per week

The course is taught over two academic years. Lessons include theoretical work, computational problem-solving, practical programming, individual and collaborative activities, preparation for the case study and work on the Computational Solution.

COURSE CONTENT

The Diploma Programme Computer Science syllabus is organized around two main themes.

Theme A – Concepts of Computer Science

Theme A is concerned primarily with how computer systems work and how computing technologies are used.

At Standard Level it consists of:

- A.1 Computer fundamentals
- A.2 Networks
- A.3 Databases
- A.4 Machine learning

Theme B – Computational Thinking and Problem-Solving

Theme B is concerned with computational approaches to solving problems and the development of algorithms and programs.

At Standard Level it consists of:

- B.1 Computational thinking
- B.2 Programming
- B.3 Object-oriented programming

The syllabus also includes a prescribed case study, an individual Computational Solution and the Collaborative Sciences Project.

AT XV. gimnazija the syllabus content is organized into the following teaching units:

Unit 1 – Computer Fundamentals

Introduction of basic principles of computer systems and computer organization.

Students study the main components of computer systems and their functions, the relationship between hardware and software, and the ways in which data and instructions are represented and processed by computers.

The unit provides the basic technical knowledge required for understanding the operation of modern computing systems and forms a foundation for later topics in the course.

Unit 2 – Data Representation

Develop understanding of how different types of data are represented within computer systems.

Students examine different forms of data representation and the relationship between the information used by people and its representation within a computer. Appropriate problem-solving exercises are used to develop students' ability to work with different representations of data.

Although taught as a separate unit for practical purposes, data representation forms part of the broader study of computer fundamentals.

Unit 3 – Computational Thinking and Algorithms

The aim of this unit is to develop a systematic approach to computational problem-solving.

Students learn to specify and decompose problems, identify patterns, use abstraction and generalization and develop algorithms for solving problems. They represent, trace and evaluate algorithms and consider the efficiency and suitability of different approaches.

Particular attention is given to developing logical and algorithmic thinking before and alongside the implementation of solutions in a programming language.

Unit 4 – Programming in Python

Development of practical programming skills and the ability to translate algorithms into working programs.

Students use Python to implement computational solutions. Topics include variables, data types, expressions, input and output, selection, iteration, functions and appropriate data structures.

They also learn to read and trace programs, identify and correct errors, test programs systematically and modify existing programs.

Programming exercises of increasing complexity are used throughout the course to develop confidence and independence in problem-solving.

Unit 5 – Object-Oriented Programming in Python

Introduction to concepts and principles of object-oriented programming and their application in computational problem-solving.

Students work with classes and objects and learn how object-oriented techniques can be used to organize programs and model real-world entities and relationships.

Object-oriented concepts are applied through practical programming activities in Python.

Unit 6 – Networks

This unit introduces computer networks and communication between computer systems.

Students study how devices communicate, how networks are organized and how data is transmitted between systems. They consider relevant network technologies and protocols, as well as issues of reliability, security and performance.

The unit also considers the importance of networks in modern computing and the implications of increasingly connected systems.

Unit 7 – Databases

Understanding basic principles used to organize, store and retrieve data.

Students study database concepts and relational databases, including the organization of data into tables and relationships between data. They consider how databases are designed, queried and maintained and examine issues such as data integrity and security.

Practical activities are used where appropriate to connect database concepts with real-world applications.

Unit 8 – Machine Learning

Introduction to the fundamental concepts and applications of machine learning.

Students consider how computer systems can use data to identify patterns and how machine-learning techniques can be used to make predictions or decisions.

The emphasis is on understanding the underlying concepts, appropriate applications and limitations rather than on the mathematical development of advanced machine-learning models.

Students also consider issues associated with the use of machine learning, including the quality and use of data, bias, reliability and the social and ethical implications of automated systems.

Unit 9 – Case Study

The prescribed case study allows students to apply knowledge and understanding from different parts of the syllabus to a contemporary computer science context.

Students become familiar with the terminology, concepts and technologies relevant to the case study and investigate the wider implications of the issues presented.

Work on the case study requires students to connect different areas of computer science rather than study them as isolated topics.

Unit 10 – Computational Solution

The Computational Solution is an individual practical project completed as part of the internal assessment.

Students select an appropriate real-world problem and apply the computational thinking process to develop a solution. Work on the project includes analysis of the problem, development of an appropriate design, implementation, testing and evaluation.

Students are expected to demonstrate independent problem-solving and appropriate use of programming techniques developed during the course.

COLLABORATIVE SCIENCES PROJECT

The project is an interdisciplinary collaborative activity in which students work together on a scientific or technological problem. Its purpose is to provide students with experience of collaborative work across scientific disciplines and to develop communication, organization and problem-solving skills.

TEACHING AND LEARNING

Students are expected to participate actively in lessons and to develop their skills through regular practice. Programming is learned primarily by writing, testing, debugging and improving programs rather than by studying program code only.

Where appropriate, topics are connected across the syllabus. For example, programming activities may be used when studying databases, networks or machine learning, while computational thinking is applied throughout the course.

Students are encouraged to use correct computer science terminology, explain their reasoning clearly and justify the choices made when developing solutions.

Practical work is an integral part of the course. Students are expected to develop good working habits when designing and implementing programs, including appropriate naming, organization of code, documentation, systematic testing and evaluation.

Students are also encouraged to consider more than one possible solution to a problem and to evaluate solutions in terms of correctness, efficiency, usability and suitability for the intended purpose.

RESOURCES

Students use a combination of:

- teacher-prepared materials
- IB Computer Science materials
- Python programming environments and documentation
- selected textbooks and reference materials
- online educational resources
- the prescribed IB case study
- practice questions and past-paper-style examination materials.

Additional resources are provided according to the requirements of individual units.

TEXTBOOKS:

MacKenty, B., Stephenson, R.: *Computer Science*, Oxford University Press, 2025.

Baumgarten, P., Ganea, I., Turland, C.: *Computer Science*, Hodder Education, 2025.

DOCUMENT ADAPTED FROM

IB Computer Science Guide.

Computer science subject brief: Computer Science – first assessment 2027.