De La Salle School Computer Science Curriculum Year 11

2d Practical Programming Skills: This topic develops pupils' practical programming skills through ongoing tasks completed throughout Year 11 using Python. Pupils will design, write, test, and refine programs to solve problems using a high-level text-based language. These skills are assessed as part of Component 2 of the qualification, particularly Section B.

Year 11 Term 1	Year 11 Term 2	Year 11 Term 3
2.1 - Algorithms **2.1.1 Computational Thinking** This topic introduces the fundamental principles of computational thinking, which are essential for solving problems in Computer Science. Pupils will learn how to use abstraction, decomposition, and algorithmic thinking to break down problems, identify key information, and design clear step-by-step solutions. These skills underpin both programming and exam success. **2.1.2 Problem Solving and Algorithms** This topic builds pupils' understanding of how to analyse problems and design clear solutions. Pupils will learn how to identify inputs, processes, and outputs, and represent problems using structure diagrams. They will develop and refine algorithms using pseudocode, flowcharts, and high-level programming languages. Pupils will also learn to spot common syntax and logic errors, and use trace tables to test and follow the steps of an algorithm. **2.1.3 Searching and Sorting Algorithms** This topic introduces common searching algorithms, including linear search and binary search. Pupils will learn the steps of each algorithm, when they can be used, and any requirements needed before applying them. They will practise applying these algorithms to	**2.2.3 Additional Programming Techniques** This topic builds pupils' programming skills by introducing string manipulation, file handling (open, read, write, close), records, SQL queries (SELECT, FROM, WHERE), and arrays (1D and 2D). Pupils will also learn how to use functions and procedures to write structured programs, and how to generate random numbers. They will apply these techniques in a high-level programming language to solve problems effectively. **2.3.1 Defensive Design** This topic explores how programs are designed to handle errors and misuse. Pupils will learn about anticipating misuse, authentication, and input validation to ensure programs work correctly with different types of data. They will also study maintainability features such as using subprograms, naming conventions, indentation, and comments to make code easier to read and improve. **2.3.2 Testing** This topic covers the purpose of testing in software development and the different types of testing, including iterative and final (terminal) testing. Pupils will learn to identify syntax and logic errors and understand different types of test data, including normal, boundary, invalid, and erroneous data. They	**2.5 – Programming languages and Integrated Development Environments** **2.5.1 Languages** This topic explores different types of programming languages, including high-level and low-level languages, and their characteristics and purposes. Pupils will learn about the role of translators and the differences between compilers and interpreters, including their advantages and disadvantages. **2.5.2 The Integrated Development Environment (IDE)** This topic introduces the tools available in an Integrated Development Environment (IDE), including the editor, error diagnostics, run-time environment, and translators. Pupils will learn how each tool supports program development and gain practical experience using them in an IDE.

data sets and identifying them from code or pseudocode. **2.2.2 Data Types** This topic covers the different data types used in programming, including integer, real, boolean, character, and string. Pupils will learn how to use these in a high-level programming language and select suitable data types for different situations. They will also understand casting and how data types can be temporarily changed when needed.	will also learn how to refine algorithms and create test plans to ensure programs work correctly. **2.4 – Boolean logic** **2.4.1 Boolean Logic** This topic covers Boolean logic using the operators AND, OR, and NOT. Pupils will learn how to create and interpret logic diagrams and truth tables, including combining multiple logic gates to solve problems. They will also apply logical operators in truth tables and understand how each gate works.	
Knowledge, Skills and Understanding		

2.1.1 Computational Thinking
 	
- Principles of computational thinking:
  o Abstraction
  o Decomposition
  o Algorithmic thinking
  o Understanding how these principles are used to define and refine problems
- Applying computational thinking to break down and solve problems systematically
 	
2.1.3 Searching and Sorting Algorithms
 	
- Understand linear and binary search
- Know when algorithms can be used
- Apply searches to data sets
- Identify algorithms from code or pseudocode
- Sorting algorithms are not required
 	
2.2.1 Programming Fundamentals
 	
- Variables, constants, inputs, outputs, assignment
	2.2.3 Additional Programming Techniques
	
	- String manipulation
	- File handling
	- Records and SQL
	- 1D and 2D arrays
	- Functions and procedures
	- Random numbers
	
	2.3.1 Defensive Design
	
	- Anticipating misuse and authentication
	- Input validation
	- Maintainability through:
	- Subprograms
	- Naming conventions
	- Indentation
	- Commenting
		2.5.1 Languages
		
		- High-level and low-level languages
		- Purpose of translators
		- Compilers and interpreters
		- Differences and characteristics of each
		
		2.5.2 The Integrated Development Environment (IDE)
		
		- Editor
		- Error diagnostics
		- Run-time environment
		- Translators

- Sequence, selection, iteration - Arithmetic operators - Boolean operators (AND, OR, NOT)	**2.3.2 Testing** - Purpose of testing - Iterative and final testing - Syntax and logic errors - Test data: normal, boundary, invalid - Refining algorithms **2.4.1 Boolean Logic** - Logic diagrams using AND, OR, NOT - Truth tables - Combining Boolean operators - Solving problems using truth tables	

Assessment and Assessment Focus

- Each sub-unit includes one 40-mark online team assessment to check pupils' understanding of the key concepts covered. The quiz provides instant feedback so pupils can quickly identify strengths and areas for improvement.
- Pupils will also complete regular knowledge recall activities at the start of lessons, usually consisting of five short questions based on previous learning. These help teachers identify gaps in understanding and decide whether pupils are ready to move on.
- Mini whiteboards will be used during lessons to quickly assess pupils' progress and understanding in real time.
- Teachers will provide demonstrations and use knowledge organisers to help pupils focus on the most important content and key terminology.
- Pupils will complete weekly homework using Smart Revise, which provides exam-board-created practice questions. Pupils receive instant feedback, a breakdown of strengths and weaknesses, and personalised revision lists. Teachers can also use homework leader boards to motivate and encourage pupils.
- At the end of the unit, pupils will complete a 45-minute exam under formal exam conditions. The paper is created using the OCR Exam Builder and assesses knowledge and understanding from the whole unit.
- The end-of-unit assessment will be marked by the teacher and used in a follow-up therapy lesson, where pupils review mistakes and improve gaps in their learning.