Salesian International School 2025-2026 Module Rubric					
Year	10	**Course**	AG YR 10 Computer Science	**Credits**	3
Module Title	Computer Science			**Required Materials**	
Unit Summary	In Year 10 Computer Science, students develop strong problem-solving and algorithmic thinking skills by exploring various computer programming topics using the Python programming language. Python is a robust, industry-standard programming language with a relatively simple syntax and structure that is well-suited to both beginners and experienced programmers alike. Throughout the year, students will write programs to solve problems related to topics with diverse applications in science, technology, engineering, art, and math (STEAM). By the end of the second term, students will be prepared to continue their computer science studies in the WACE Year 11 Computer Science course in Term 3.			*Laptop* *Backup Charger* *Internet access* *Headphones* *Notebook/Binder* *Writing implements* *Folder*	

Assessment Basis				**Unit Contents**	
	Knowledge Fundamentals	**Application Communication**	**Inquiry Creativity**	**Key Topics Covered:** **Term 1:** Algorithms and Programming Understanding Data: Storage and Manipulation Control Structures: Decision Making	**Term 2:** Control Structures: Repetition Functions Data Containers: Lists
Research **Level 3: Creative Thought**	**A3 (10)** • I can combine computer science concepts learned across different topics to solve complex, multi-step computer science problems. • I can apply computer science concepts in unfamiliar contexts that require me to think abstractly, creatively, and flexibly.	**B3 (13)** • I can present multiple methods for solving problems and compare their efficiency across a variety of measures. • I can collaborate with others to refine and explain various problem-solving approaches.	**C3 (23)** • I can solve complex, real-world problems by writing computer programs that utilize original and creative problem-solving techniques. • I can consider how my code can be used incorrectly or maliciously by the user, and take appropriate measures to safeguard my program from crashing or working in unintended ways. • I can go beyond the project specifications and implement creative and relevant features that improve the user experience.	**Learning Objectives:** **Students will develop:** • Their logical thinking and algorithmic thinking abilities in order to break down, large, complex problems into a sequence of simple and easy to execute steps or instructions. Logical thinking is at the core of writing of algorithms and computer programs that can be used to solve real-world problems in a variety of contexts. • Their resilience as both a thinker and problem solver. If is often the case when writing computer programs that students will encounter errors (problems) in their code that will prevent further progress until properly resolved. Approaching these situations with patience, curiosity, and an iterative design approach will allow students to continue to make progress with their programs despite the most difficult of setbacks. • Their creative thinking abilities by writing programs that utilize original and creative problem-solving approaches. Students will also design and create programs that are intuitive and easy to use from a user interface (UI) design perspective. Technical skills: • The ability to design and write algorithms in both English and pseudocode to solve a variety of real-world problems. • The ability to transcribe algorithms from English and pseudocode into a high-level programming language such as Python. • The ability to distinguish between quanlitative and quantatative data in computer science. • The ability to manipulate quantitative data using a variety of arithmetic operators and functions. • The ability to manipulate qualitative data using a variety of String functions, methods, and operators. • The ability to utilize control structures such as if, else, and else if statements in order to manipulate the flow of a program depending on the value of specific data. • The ability to utilize nested control structures to implement significantly more complex decisions in a program. • The ability to employ while and for loops in order to repeat essential code segments. • The ability to use Python's built-in and library functions, as well as design and implement their own user-defined functions. • The ability to store data in a variety of Python's data structures, including lists, sets, and dictionaries.
Application **Level 2: Critical Thought**	**A2 (7)** • I can use Python's built-in operators and functions in a variety of contexts beyond those taught in class. • I can apply concepts and techniques learned in class to solve intermediate, multi-step computer science problems. • I can identify and resolve logical errors in code.	**B2 (10)** • I can clearly explain the steps in my problem-solving approach. • I can use a variety of measures, including code readability, run-time efficiency, and space efficiency, to justify my problem-solving approach. • I can effectively describe and justify my user interface design choices. • I can effectively describe any assumptions I made in my problem-solving and design approach.	**C2 (17)** • I am capable of solving problems by writing Python programs that utilize the problem-solving techniques taught in class. • I can design programs that are intuitive to use and demonstrate an attention to good user interface design principles. • I can write full and complete programs that are free from compile-time and run-time errors to satisfy the assignment requirements.		
				Notes, Means of Assessment	

Foundation **Level 1: Logical Thought**	**A1 (3)** • I can recognize and use basic computer science concepts relevant to the topic. • I can apply the concepts and techniques learned in class to solve basic computer science problems. • I can use hand-tracing to predict the output of algorithms and computer programs. • I can identify and resolve syntax errors in code.	**B1 (7)** • I can identify and describe proper computer science terminology, including Python code structures, operators, functions and methods. • I can use proper computer science terminology when writing comments and documentation for my code.	**C1 (10)** • I can ask relevant questions to clarify and deepen my understanding of computer science concepts. • I can design flow charts and other logic diagrams to represent the control flow of a computer program. • I can use pseudocode, algorithms or code to outline the basic workings of a computer program.	Term 1: • Daily Work and participation • Written review questions • Programming exercises • Programming projects • Exams Term 2: • Daily Work and participation • Written review questions • Programming exercises • Programming projects • Exams	
	Basics	**Development**	**Judgment**		