Lycée | International School
Michel Lucius

7IEC Introduction to Coding Syllabus

Introduction to Coding is the perfect introduction for students to learn how to develop computational thinking skills and program using a text-based programming language.

Students will understand the basics of Python, a text-based programming language, and its advantages over block-based languages such as Scratch. Along the way, they will gain experience with using variables, IF statements and simple loops through a variety of fun and engaging coding exercises and computational-thinking focused activities and worksheets.

This unit has been sequenced and scaffolded in a careful manner that gently reinforces and encourages students' progress and learning. Interactive demos, animated slides and videos are provided for every lesson, and the course is summative assessed through a coding project.

All programming is done within the browser using replit.com. No software installation or sign-up is required.

Assessments:

- An interactive text adventure game which can optionally be integrated in a cross-discipline manner:
    - Science: sci-fi themed adventure set on a distant planet or in a futuristic laboratory.
    - History: adventure in a historical period or event, such as the Roman Empire, the Middle Ages, or World War II.
    - Geography: adventure that takes players on a journey around the world, visiting different countries and landmarks
    - English: literacy-based exploration of text themes

Overall Student Learning Objectives

- Students gain an introductory understanding of programming using a text-based coding language.
- Students experience success in coding small programs and completing exercises.
- Students incorporate visual and literacy based creative elements into their programs.
- Students have fun and gain confidence with programming and appreciate the relevance it has in their lives.

Computational Thinking and Programming Learning Objectives:

- 7CT.01 Follow, understand, edit, and correct algorithms that are presented as flowcharts.
- 7CT.02 Know how to create algorithms using flowchart symbols.
- 7CT.03 Follow and understand the logic of AND, OR, NOT.
- 7CT.04 Understand and use selection statements, limited to IF, THEN, ELSE, presented as flowcharts.
- 7CT.05 Predict the outcome of flowcharts that use selection.
- 7CT.06 Explain the importance of pattern recognition when designing solutions to tasks.
- 7CT.07 Follow, understand, edit, and correct algorithms that use sub-routines.
- 7CT.08 Select and use appropriate constructs in algorithms written as flowcharts, limited to sequence and selection.
- 7CT.09 Select and use appropriate comparison operators in algorithms, limited to <,>, <=, >=, == (equal to) and != (not equal to).

- 7P.01 Identify and describe data types in text-based programs, including Integer, Real and String.
- 7P.02 Know how to develop text-based programs that use input and output.
- 7P.03 Know how to develop text-based programs using data types, including Integer, Real, and String.
- 7P.04 Know how to use variables in text-based programs.
- 7P.05 Know how to develop text-based programs that use different arithmetic operators, including +, −, *, /.
- 7P.06 Evaluate prototypes for software development projects.
- 7P.07 Explain the purpose of project plans for software development projects.
- 7P.08 Know how to apply test plans.
- 7P.09 Understand how errors can be introduced into programs.
- 7P.10 Know how to systematically identify and debug errors in text-based programs.
- 7P.11 Know how to develop programs for a physical computing device to generate multiple outputs, based on multiple inputs.

Learning Objectives per Lesson

The table below outlines the 8-lesson sequence that comprise this unit of learning.

Lesson	Learning Objectives
1	***Introduction to CS in Schools*** ● Writing and understanding your first Program: Hello, world! ● Playing with and modifying an existing program
2	***Displaying Text on the Screen and Input*** ● Introduction to "Whitespace" in code ● Understand what Error Messages are and how they help ● Learn how programs flow ● print() - Displaying text on the screen ● input() - Used to pause and wait for [ENTER] key
3	***Colour your World!*** ● Display text in different colours, highlights and styles ● Use and understand how the "+" symbol concatenates style constants to strings
4	***Input and Introduction to Variables*** ● Be introduced to variables as a way to store values ● Accept string input() from a user and store it into a variable. Use of the "=" character to assign values ● Using the print() with "+" character to display the value inside the variable on the screen
5	***Programs that Make Decisions*** ● Introduction to flowcharts - How they are used to represent the flow of a program ● if statements - How we can use them to make choices in a program ● Introduction of "==", used to compare if one expression is equivalent to another ● The importance of indentation in if statements
6	***Round and Round We Go (Loops)*** ● How loops are represented in flowcharts ● Using while as an introduction to loops
7	***Introduction to the Assignment*** ● Be provided with an outline of the assignment rubric, code examples, and template; ● Commence working on the assignment
8	***Working Lesson and Finale*** ● Continuing working on the assignment ● Next steps beyond CS in Schools Year 7 ● Farewells!

Overview of Lesson Format

Each of the lessons in the core syllabus generally follow the same format.

They each address the 5 Es of Learning: Engage, Explore, Explain, Elaborate, Evaluate.

Section	**Description**
Learning Objectives	An overview of key learning goals is outlined
Engage Coding Demo	Pre-written code, related to this lesson's topic, is demoed to the students
Explore Student Tweaking of Code Demo	Students tweak the demoed code to customize it.
Explain Theory and Concepts	Formal concepts and explanations are taught. Co-construction of programs is done here.
Elaborate Exercise 1	Students undertake 3 exercises that allows them to gradually construct and demonstrate their understanding of the content: ● Exercise 1 is a scaffolded activity that requires "fill-in-the-blanks" to provide a gentle introduction to individual construction. ● Exercise 2 requires students to complete some pre-written code. ● Exercise 3 requires students to write code from scratch.
Elaborate Exercise 2	
Elaborate Exercise 3	
Explain and Elaborate (as needed)	Walkthroughs of solutions are provided for Exercise 3. This is useful for students who wish to revisit topics to clarify their understanding or to catch students up who may have missed classes etc.
Explore and Extended Extension Exercise	Students who complete all the exercises quickly and easily and have demonstrated mastery of the concepts covered, can undertake the optional extension exercises to deepen and/or broaden their understanding.
Evaluate Show and Tell, Formal Reflection	Students present and talk about their work and learning process to the class, linking it back to the topic(s) covered.