**THE ACADEMY FOR GIFTED CHILDREN – P.A.C.E.**
**CURRICULUM OUTLINE AND EVALUATION**
**GRADE 7 COMPUTER SCIENCE II**

## CURRICULUM OUTLINE

By the end of this course, students will:

- solve problems and create computational representations of mathematical situations using coding concepts and skills;
- describe fundamental programming concepts and constructs;
- plan and write simple programs using fundamental programming concepts;
- apply code maintenance techniques and conventions when creating computer programs;
- demonstrate the ability to use different data types, including lists, in computer programs;
- demonstrate the ability to use control structures and simple algorithms in computer programs;
- demonstrate the ability to use subprograms within computer programs;
- design software solutions to meet a variety of challenges;
- investigate how science and technology can be used with other subject areas to address real-world problems

## UNITS OF STUDY

### Unit 1: Programming in Python

By the end of the Unit, students will:

- apply the process of mathematical modelling to represent, analyse, make predictions, and provide insight into real-life situations;
- demonstrate the ability to use integer division and resultant remainders in computer programs;
- design algorithms to solve practical mathematical problems;
- use an operating system to organize computer programs and files logically on local and shared drives;
- use general keyboard shortcuts to perform common tasks (e.g., cut, copy, paste, print, print window, print screen);
- outline and apply strategies to encourage ethical computing practices at home, at school, and at work;
- use correct terminology to describe programming concepts;
- correctly use different types of data (e.g., numbers, text);
- explain the difference between constants and variables used in programming;
- determine the expressions and instructions to use in a programming statement, considering the order in which conditions will be processed;
- describe the function of Boolean operators, comparison, and arithmetic operators, and use them correctly in programming;
- select appropriate structure to be used when solving a coding problem;
- write keyboard input and screen output statements that conform to program specifications;
- write a program that includes a decision structure for two or more choices;
- demonstrate the ability to use existing sub-programs (e.g., random number generator, substring, absolute value) within computer programs;
- use sequence, selection, and repetition control structures to create programming solutions;
- describe fundamental concepts in programming, that are similar across languages;
- work independently to design and write functioning computer programs;
- demonstrate the ability to correct syntax, logic, and run-time errors in computer programs;
- represent the structure and components of a program using pseudocode.

### Unit 2: Strings & Lists

By the end of the Unit, students will:

- use assignment statements correctly with both arithmetic and string expressions in computer programs;
- demonstrate the ability to use non-numeric comparisons (e.g., strings) in computer programs;
- describe the functions and properties of strings in coding;
- apply concatenation and repetition when coding with strings;
- use slicing to pick out parts of a string;
- use repetition structures to search and work with strings;

- identify which string-based functions to use to solve problems;
- describe the function and purpose of escape characters;
- develop a strategy to solve a problem involving strings;
- describe the function and properties of lists in Python;
- describe the parallels between working with strings and lists in Python;
- use knowledge of the similarities between strings and lists to solve list-based problems with strings-based strategies;
- determine when to use built-in functions and methods for working with lists;
- apply functions and methods for working with strings and lists to solve applicable problems;
- develop a strategy to solve problems involving lists;
- describe the structure of lists and related concepts, including elements, indexes, and bounds;
- write programs that declare, initialize, modify, and access lists;
- describe and use lists of compound data types (e.g., numbers, strings) in a computer program;
- demonstrate the ability to add, change, or delete items in a list of elements;
- demonstrate the ability to use search and sort routines (e.g., string1.index("cool"), list1.sort()) in a program.

### Unit 3: While Loops

By the end of the Unit, students will:

- read and alter existing code, including code that involves events influenced by a defined count and/or sub-program and other control structures, and describe how changes to the code affect the outcomes and the efficiency of the code;
- describe the functions and purpose of the 'while' loop;
- prevent infinite repetition errors;
- apply the 'break' and 'else' statements in the appropriate contexts;
- solve various problems involving 'while' loops;
- troubleshoot issues with 'while' loops.

### Unit 4: Computer Science Unplugged

By the end of the Unit, students will:

- describe how binary numbers are used to represent more complicated constructs;
- represent more complicated constructs with dichotomous symbols;
- describe how computers store images using numbers;
- create simple images using coding methods;
- describe how computers store lots of information in limited space using compression;
- code data before it is stored;
- estimate information using strategies informed by Information Theory;
- create and use decision trees;
- describe the function of, and use, sorting algorithms, linear search methods, binary search methods and hashing.

## ASSESSMENT AND EVALUATION

### ASSESSMENT FOR LEARNING

| | Term I | Term II | Term III |
|---|---|---|---|
| Knowledge & Understanding | Term I Check-in (Discussion)<br>Sample Code Submission (Student Product) | Term II Check-in (Discussion)<br>Pseudocode Check (Obs.) | Term III Check-in (Discussion)<br>Pseudocode Check (Obs.) |
| Application | Term I Check-in (Discussion)<br>Sample Code Submission (Student Product) | Term II Check-in (Discussion)<br>Pseudocode Check (Obs.) | Term III Check-in (Discussion)<br>Pseudocode Check (Obs.) |
| Thinking/Inquiry | Term I Check-in (Discussion)<br>Sample Code Submission (Student Product) | Term II Check-in (Discussion)<br>Pseudocode Check (Obs.) | Term III Check-in (Discussion)<br>Pseudocode Check (Obs.) |
| Communication | Term I Check-in (Discussion)<br>Sample Code Submission (Student Product) | Term II Check-in (Discussion)<br>Pseudocode Check (Obs.) | Term III Check-in (Discussion)<br>Pseudocode Check (Obs.) |

**ASSESSMENT AS LEARNING**

| | Term I | Term II | Term III |
|---|---|---|---|
| Knowledge & Understanding | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.)<br>Cryptography Program (Obs.) |
| Application | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.)<br>Cryptography Program (Obs.) |
| Thinking/Inquiry | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.)<br>Cryptography Program (Obs.) |
| Communication | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.) | In-Class Examples (Obs.)<br>Exit Cards (Obs.)<br>Cryptography Program (Obs.) |

**ASSESSMENT OF LEARNING (EVALUATION)**

| | Term I | Term II | Term III |
|---|---|---|---|
| Knowledge & Understanding | 25%<br>• Pseudocode Assignment (Discussion)<br>• For Loops Assignment (Student Product)<br>• Programming in Python Test (Student Product)<br>• Strings Coding Assignment (Obs.) | 25%<br>• Strings Written Assessment (Student Product)<br>• Strings & Lists Test (Student Product)<br>• Lists Coding Assignment (Obs.)<br>• Present Your Code (Discussion) | 25%<br>• Cryptography Program (Student Product)<br>• Python Game (Student Product)<br>• Programming a Microprocessor (Discussion)<br>• Presentations (Obs.) |
| Application | 25%<br>• Pseudocode Assignment (Discussion)<br>• For Loops Assignment (Student Product)<br>• Programming in Python Test (Student Product)<br>• Strings Coding Assignment (Obs.) | 25%<br>• Strings Written Assessment (Student Product)<br>• Strings & Lists Test (Student Product)<br>• Lists Coding Assignment (Obs.)<br>• Present Your Code (Discussion) | 25%<br>• Cryptography Program (Student Product)<br>• Python Game (Student Product)<br>• Programming a Microprocessor (Discussion)<br>• Presentations (Obs.) |
| Thinking/Inquiry | 25%<br>• Pseudocode Assignment (Discussion)<br>• For Loops Assignment (Student Product)<br>• Programming in Python Test (Student Product)<br>• Strings Coding Assignment (Obs.) | 25%<br>• Strings Written Assessment (Student Product)<br>• Strings & Lists Test (Student Product)<br>• Lists Coding Assignment (Obs.)<br>• Present Your Code (Discussion) | 25%<br>• Cryptography Program (Student Product)<br>• Python Game (Student Product)<br>• Programming a Microprocessor (Discussion)<br>• Presentations (Obs.) |

| | | | |
|---|---|---|---|
| Communication | 25%<br><br>• Pseudocode Assignment (Discussion)<br>• For Loops Assignment (Student Product)<br>• Programming in Python Test (Student Product)<br>• Strings Coding Assignment (Obs.) | 25%<br><br>• Strings Written Assessment (Student Product)<br>• Strings & Lists Test (Student Product)<br>• Lists Coding Assignment (Obs.)<br>• Present Your Code (Discussion) | 25%<br><br>• Cryptography Program (Student Product)<br>• Python Game (Student Product)<br>• Programming a Microprocessor (Discussion)<br>• Presentations (Obs.) |
| **Total** | **100%** | **100%** | **100%** |

**Final Score = (Term I + Term II + Term III) ÷ 3**

**RESOURCES**

- Python (free download)
- Digital Textbook (free download)
- Ontario Mathematics Guidelines 2020
- Ontario Science and Technology Guidelines 2022
- Locally Designed Resource by P.A.C.E.