Readington Township Public Schools

Coding 2
(2nd year Coding Students- 8th grade)

Authored by: Lora Petersen

Reviewed by: Sarah Pauch
Supervisor of Math, Science, and Technology

Approval Date: September 13, 2022

Readington Township Public Schools
www.readington.k12.nj.us

I. OVERVIEW

Coding 2 is the 3rd and final cycle course in Computers offered in RMS. New approaches necessary for solving the critical challenges that we face as a society will require harnessing the power of technology and computing. Rapidly changing technologies and the proliferation of digital information have permeated and radically transformed learning, working, and everyday life. To be well-educated, global-minded individuals in a computing-intensive world, students must have a clear understanding of the concepts and practices of computer science. As education systems adapt to a vision of students who are not just computer users but also computationally literate creators who are proficient in the concepts and practices of computer science and design thinking, engaging students in computational thinking and human-centered approaches to design through the study of computer science and technology serves to prepare students to ethically produce and critically consume technology. (New Jersey Department of Education)

II. STUDENT OUTCOMES (2020 New Jersey Student Learning Standards – Computer Science)

The course objectives will cover but are not limited to these standards:

Computing Systems

8.1.8.CS.1:	Recommend improvements to computing devices in order to improve the ways users interact with the devices.
8.1.8.CS.3:	Justify design decisions and explain potential system trade-offs.
8.1.8.CS.4:	Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.

Impacts of Computing

8.1.8.IC.1:	Compare the trade-offs associated with computing technologies that affect individual's everyday activities and career options.
8.1.8.IC.2:	Describe issues of bias and accessibility in the design of existing technologies.

Algorithms & Programming

8.1.8.AP.1:	Design and illustrate algorithms that solve complex problems using flowcharts and/or pseudocode.
8.1.8.AP.2:	Create clearly named variables that represent different data types and perform operations on their values.
8.1.8.AP.3:	Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
8.1.8.AP.4:	Decompose problems and sub-problems into parts to facilitate the design, implementation, and review of programs.
8.1.8.AP.5:	Create procedures with parameters to organize code and make it easier to reuse.
8.1.8.AP.6:	Refine a solution that meets users' needs by incorporating feedback from team members and users.
8.1.8.AP.7:	Design programs, incorporating existing code, media, and libraries, and give attribution.
8.1.8.AP.8:	Systematically test and refine programs using a range of test cases and users.
8.1.8.AP.9:	Document programs in order to make them easier to follow, test, and debug.

III. COURSE OBJECTIVES

Foundations of Coding

- Introduction to Various Programming Languages
  Students will be able to:
  - Learn the syntax of various programming languages
  - Languages may include, but are not limited to:
    - JavaScript
    - C++
  - Become familiar with the coding environments used in class
  - Discuss the concept of planning and its importance in coding

- Loops
  Students will be able to:
  - Define loop as a programming term
  - Understand why using loops in programming is more efficient
  - Discuss proper syntax for loops

- Variables
  Students will be able to:
  - Define variable as a programming term
  - Discuss how and why we use variables in programming
  - Learn how to use additional functions

- Arrays
  Students will be able to:
  - Understand the concept of an array and its elements
  - Learn how to use indexes to access array elements
  - Learn how to define new arrays
  - Work with arrays that contain objects of different types

- For Loops
  Students will be able to:
  - Identify the difference between a simple loop and a for loop
  - Discuss how and when to use for loops
  - Learn how to use the loop variable in other ways than passing it as an argument
  - Practice for loops
  - Use nested loops

- Functions
  Students will be able to:
  - Make comments in their code
  - Work with functions and comments
  - Read and write functions
  - Review, practice, and deepen their expertise in functions
  - Learn and practice returning values from a function they write
  - Practice returning values from functions and calling functions in new syntax

- <u>Until Loops</u>

  Students will be able to:
  - Work with until loops
  - Practice until loops definitions
  - Write an until loop from scratch
  - Use functions with until loops

- <u>Conditionals:</u>

  Students will be able to:
  - Learn about conditionals and use them in code
  - Practice using if statements
  - Use if-else statements
  - Write code that uses if-else from scratch
  - Use if-else within a function definition
  - Use multiple conditions
  - Use if-else inside a for loop

- <u>Logical Operators:</u>

  Students will be able to:
  - Use the and, or, and not operators
  - Practice using logical operators

- <u>Comparing Values</u>

  Students will be able to:
  - Learn how to compare values in programming and practice using them
  - Learn about the less-than operator

- <u>Debugging</u>

  Students will be able to:
  - Define Debugging
  - Learn various debugging techniques

- <u>Design and Concept Development</u>

  Students will be able to:
  - Understand and utilize the steps in an engineering design process (similar to the one below) :
    - Ask: Identify the Need and Constraints
    - Research the Problem
    - Imagine: Develop Possible Solutions
    - Plan: Select a Promising Solution
    - Create: Build a Prototype
    - Test and Evaluate Prototype
    - Improve: Redesign as Needed
  - Plan in advance for an ongoing assignment
  - Explain how system limitations can affect project design
  - Describe how compromise can help keep a project on track and inspire creativity
  - Design, develop and produce several projects using all information learned in this course

IV. STRATEGIES

Strategies may include but are not limited to:

· Group discussions
· Teacher presentation
· Student projects
· Guided groups
· One to one instruction
· Interactive SmartBoard lessons
· Tutorials
· Online practice

V. EVALUATION

Assessments may include but are not limited to:

· Teacher Observations
· Class Participation
· Class Discussions
· Class Assignments
· Homework Assignments
· Notebooks
· Student Projects
· Tests and Quizzes
· Anecdotal Records
· Presentations

VI. REQUIRED RESOURCES

· Vidcode
· Code Car
· Chromebooks
· Headphones

Supplemental Resources may include, but are not limited to:

· Code Monkey
· Code Master
· Makey Makey Classic
· Ozobots
· Tablets
· Tynker
· MIT App Lab and App Companion
· Common Sense Media
· Code.org
· Beanz Magazine
· Kahoot
· Blooket
· Brain Pop
· Khan Academy

VII. SCOPE AND SEQUENCE

Additional time will be spent on reviewing concepts that may need to be revisited.

- <u>Foundations of Coding (20 days)</u>
  - ·Introduction
  - · Loops
  - · Variables
  - · Arrays
  - · For Loops
  - · Functions
  - ·Until Loops
  - · Conditionals:
  - · Logical Operators
  - · Comparing Values
  - · Debugging

- <u>Design and Concept Development</u> (20 days)
  - · Design Process
  - · Coding in Various Languages
  - · Design, Produce, and Share Multiple Projects