Middle School (Grades 7 – 8) Coding Curriculum Guide (2022 Revision II)

LINDEN PUBLIC SCHOOLS
LINDEN, NEW JERSEY

DR. MARNIE HAZELTON
SUPERINTENDENT

DENISE CLEARY
ASSISTANT SUPERINTENDENT

JOSEPH SCALDINO
SUPERVISOR OF TECHNOLOGY

The Linden Board of Education adopted the revised curriculum guide on:
Date: _____September 1, 2022_____

EDUCATION EQUITY: The Linden Public School District guarantees each student equal educational opportunity regardless of age, race, color, creed, religion, gender, language, affectional or sexual orientation, ancestry, national origin, marital or economic status. For information, contact District Educational Equity Officer at (908) 486-2800 x 8307.

NONDISCRIMINATION: The Linden Public School District does not discriminate against handicapped persons in admission or access to or treatment or employment in its programs, activities, and vocational opportunities. For information contact District Public 504 Officer at (908) 486-2800 x 8025.

This page has been left intentionally blank.

Table of Contents

Acknowledgements

The first version of this curriculum guide was developed by Howard Schulz and Holly Sepulveda in 2017 under the approval of Michael Walters, former Director of Technology for Linden Public Schools. This 2022 curriculum revision has been completed by Durell Burress under the approval of Joseph Scaldino, Supervisor of Instructional Technology for Linden Public Schools, based on current research and the data, experiences, feedback, and information retrieved from LPS stakeholders.

This curriculum guide also includes content and information from the following:

⇒ Amazon Future Engineer
⇒ Apple Learn to Code: Swift
⇒ Code.org's Annually-Updated Express Course
⇒ Google CS First Curriculum
⇒ Creative Computing Curriculum (Scratch 3.0)
⇒ 9.3 21st Century Life and Careers (2014)
⇒ 2020 New Jersey Student Learning Standards – Career Readiness, Life Literacies, and Key Skills

Stage 1: Desired Results

Apple Learn to Code 1 & 2

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.1 Computer Science

Computing Systems

◊ 8.1.8.CS.4: Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.

Networks and the Internet

◊ 8.1.8.NI.1: Model how information is broken down into smaller pieces, transmitted as addressed packets through multiple devices over networks and the Internet, and reassembled at the destination.
◊ 8.1.8.NI.2: Model the role of protocols in transmitting data across networks and the Internet and how they enable secure and errorless communication.

Data & Analysis

◊ 8.1.8.DA.5: Test, analyze, and refine computational models.

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.

CSTA COMPUTER SCIENCE K-12 STANDARDS

Algorithms & Programming

- 2-AP-10: Use flowcharts and/or pseudocode to address complex programs as algorithms.
- 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
- 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
- 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
- 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
- 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
- 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
- 2-AP-17: Systematically test and refine programs using a range of test cases.
- 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
- 2-AP-19: Document programs in order to make them easier to follow, test, and debug.

Computing Systems

- 2-CS-03: Systematically identify and fix problems with computing devices and their components.

ISTE Standards for Students

1.1 Empowered Learner

- ◊ 1.1.a: Students articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.
- ◊ 1.1.c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
- ◊ 1.1.d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.4 Innovative Designer

- ◊ 1.4.a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- ◊ 1.4.b: Students select and use digital tools to plan and manage a design process that considers design constraints and calculated risks.
- ◊ 1.4.c: Students develop, test and refine prototypes as part of a cyclical design process.
- ◊ 1.4.d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

1.5 Computational Thinker

- ◊ 1.5.a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- ◊ 1.5.b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- ◊ 1.5.c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- ◊ 1.5.d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6 Creative Communicator

◊ 1.6.c: Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

9.3 – CAREER & TECHNICAL EDUCATION (CTE) CONTENT AREA: 21ST CENTURY LIFE AND CAREERS

◊ 9.3.IT-PRG.6: Program a computer application using the appropriate programming language.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – CAREER READINESS, LIFE LITERACIES, AND KEY SKILLS

◊ 9.4.8.CI.3: Examine challenges that may exist in the adoption of new ideas (e.g., 2.1.8.SSH, 6.1.8.CivicsPD.2).
◊ 9.4.8.CI.4: Explore the role of creativity and innovation in career pathways and industries
◊ 9.4.8.CT.1: Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).
◊ 9.4.8.CT.2: Develop multiple solutions to a problem and evaluate short- and long-term effects to determine the most plausible option (e.g., MS-ETS1-4, 6.1.8.CivicsDP.1).
◊ 9.4.8.CT.3: Compare past problem-solving solutions to local, national, or global issues and analyze the factors that led to a positive or negative outcome.
◊ 9.4.8.GCA.2: Demonstrate openness to diverse ideas and perspectives through active discussions to achieve a group goal.
◊ 9.4.8.IML.9: Distinguish between ethical and unethical uses of information and media (e.g., 1.5.8.CR3b, 8.2.8.EC.2).
◊ 9.4.8.TL.5: Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.

ESSENTIAL QUESTIONS

◊ How are simple everyday tasks achieved?
◊ How can the sequencing of commands result in multiple actions?
◊ How much detail and precision are needed for coding?
◊ Why is debugging an important part of coding?
◊ When is it more appropriate to call a function than to loop a sequence of commands?
◊ How can code navigate a change of circumstances as they change from true to false or vice versa?

- ◊ How do logical operators make conditional code more efficient?
- ◊ How do computers abstract the details of code?
- ◊ When is it more appropriate to use a while loop than to use a for loop?
- ◊ How are algorithms used to make coding more efficient?
- ◊ How can assigning variables (names or values) in our code be helpful?
- ◊ How are features and properties defined and utilized in coding?
- ◊ Why are parameters useful?
- ◊ How do arrays relate to types, initialization, and variables?

ENDURING UNDERSTANDING

- ◊ Complex coding can be solved through maps, outlines, and pseudocode.
- ◊ Breaking down complex problems into smaller parts makes it easier to find solutions.
- ◊ Systematic approaches help to solve problems step-by-step.
- ◊ Documentation and journaling help programmers test and debug code.
- ◊ Compiling code and using abstraction are necessary for more efficient code-writing.
- ◊ Computing enables people to use creative development processes to create computational artifacts for creative expressions or to solve a problem.

STUDENTS WILL KNOW

- ◊ Coding is telling a computer what to do.
- ◊ Developers write code to build their own apps and games.
- ◊ A command is a specific action for a computer to perform.
- ◊ A sequence is the order in which commands are given.
- ◊ A bug is an error in your code.
- ◊ Debugging is finding errors within lines of code.
- ◊ A function is a collection of commands grouped together and given a name.
- ◊ A for loop is a group of commands that can be set to repeat for a specific number of times.

- ◊ A condition is something you test that results in true or false.
- ◊ Conditional code is a block of code that will run only if something is true.
- ◊ A Boolean is a value that can only be either true or false.
- ◊ A logical operator is a symbol or words like “and,” “or,” and “not” that connects two or more Booleans to make conditional decisions more specific.
- ◊ A while loop is a loop that runs a block of code as long as a given condition is true. When the condition is false, the loop stops running.
- ◊ An algorithm is a step-by-step set of rules or instructions.
- ◊ Pseudocode is an informal description of code or a concept that’s intended for human reading.
- ◊ A variable is a named container that stores a value. The value can change over time.
- ◊ A type is a named grouping of properties (the features) and methods (the behaviors) of a kind of data.
- ◊ Initialization is the act of creating a new instance of a type, which includes setting initial values for any properties of the type.
- ◊ String stores a series for characters. Int stores an integer. Bool stores a value of true or false.
- ◊ A parameter is extra information that gets passed to a function
- ◊ An array is a collection that stores an ordered list of items.
- ◊ An index is the number that represents the position of an item in an array.

STUDENTS WILL BE ABLE TO

- ◊ Describe and demonstrate the use of commands, sequences, loops, functions, conditions, logical operators, algorithms, parameters, types, initialization, arrays, and indexes in Swift coding.
- ◊ Evaluate their efficiency in writing Swift code.
- ◊ Identify ways to be more efficient in coding.
- ◊ Design portfolios to communicate their understanding of concepts.
- ◊ Critique peers’ lines of code by making recommendations for improvement.
- ◊ Collaborate with peers to solve complex coding puzzles.
- ◊ Categorize every day actions and associate them to coding concepts.
- ◊ Write and modify lines of Swift code.
- ◊ Understand, communicate, and translate Swift language.

The Complete Middle School Study Guide: Everything You Need to ACE Computer Science and Coding in One Big Fat Notebook

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.1 Computer Science

Computing Systems

- ◊ 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.

Networks and the Internet

- ◊ 8.1.8.NI.1: Model how information is broken down into smaller pieces, transmitted as addressed packets through multiple devices over networks and the Internet, and reassembled at the destination.
- ◊ 8.1.8.NI.2: Model the role of protocols in transmitting data across networks and the Internet and how they enable secure and errorless communication.

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.

Data & Analysis

- ◊ 8.1.8.DA.2: Explain the difference between how the computer stores data as bits and how the data is displayed.
- ◊ 8.1.8.DA.3: Identify the appropriate tool to access data based on its file format.

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.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.2 Design Thinking

Engineering Design

- ◊ 8.2.8.ED.2: Identify the steps in the design process that could be used to solve a problem.

Interaction of Technology and Humans

- ◊ 8.2.8.ITH.3: Evaluate the impact of sustainability on the development of a designed product or system.
- ◊ 8.2.8.ITH.4: Identify technologies that have been designed to reduce the negative consequences of other technologies and explain the change in impact.
- ◊ 8.2.8.ITH.5: Compare the impacts of a given technology on different societies, noting factors that may make a technology appropriate and sustainable in one society but not in another.

Nature of Technology

◊ 8.2.8.NT.4: Explain how a product designed for a specific demand was modified to meet a new demand and led to a new product.

Effects of Technology on the Natural World

◊ 8.2.8.ETW.2: Analyze the impact of modifying resources in a product or system (e.g., materials, energy, information, time, tools, people, capital).
◊ 8.2.8.ETW.3: Analyze the design of a product that negatively impacts the environment or society and develop possible solutions to lessen its impact.
◊ 8.2.8.ETW.4: Compare the environmental effects of two alternative technologies devised to address climate change issues and use data to justify which choice is best.

Ethics & Culture

◊ 8.2.8.EC.1: Explain ethical issues that may arise from the use of new technologies.
◊ 8.2.8.EC.2: Examine the effects of ethical and unethical practices in product design and development.

CSTA COMPUTER SCIENCE K-12 STANDARDS

Algorithms & Programming

◊ 2-AP-10: Use flowcharts and/or pseudocode to address complex programs as algorithms.
◊ 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
◊ 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
◊ 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
◊ 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
◊ 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
◊ 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
◊ 2-AP-17: Systematically test and refine programs using a range of test cases.

◊ 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
◊ 2-AP-19: Document programs in order to make them easier to follow, test, and debug.

Computing Systems

◊ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Impacts of Computing

◊ 2-IC-23: Describe tradeoffs between allowing information to be public and keeping information private and secure.

Networks & the Internet

◊ 2-NI-05: Explain how physical and digital security measures protect electronic information.

ISTE Standards for Students

1.1 Empowered Learner

◊ 1.1.a: Students articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.
◊ 1.1.c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
◊ 1.1.d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.4 Innovative Designer

◊ 1.4.d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

1.5 Computational Thinker

- ◊ 1.5.a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- ◊ 1.5.b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- ◊ 1.5.c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- ◊ 1.5.d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6 Creative Communicator

- ◊ 1.6.c: Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

9.3 – CAREER & TECHNICAL EDUCATION (CTE) CONTENT AREA: 21ST CENTURY LIFE AND CAREERS

- ◊ 9.3.IT.6: Describe trends in emerging and evolving computer technologies and their influence on IT practices.
- ◊ 9.3.IT.12: Demonstrate knowledge of the hardware components associated with information systems.
- ◊ 9.3.IT.13: Compare key functions and applications of software and determine maintenance strategies for computer systems.
- ◊ 9.3.IT-PRG.6: Program a computer application using the appropriate programming language.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – CAREER READINESS, LIFE LITERACIES, AND KEY SKILLS

- ◊ 9.4.8.CI.3: Examine challenges that may exist in the adoption of new ideas (e.g., 2.1.8.SSH, 6.1.8.CivicsPD.2).
- ◊ 9.4.8.CI.4: Explore the role of creativity and innovation in career pathways and industries
- ◊ 9.4.8.CT.1: Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).

- ◊ 9.4.8.CT.2: Develop multiple solutions to a problem and evaluate short- and long-term effects to determine the most plausible option (e.g., MS-ETS1-4, 6.1.8.CivicsDP.1).
- ◊ 9.4.8.CT.3: Compare past problem-solving solutions to local, national, or global issues and analyze the factors that led to a positive or negative outcome.
- ◊ 9.4.8.GCA.2: Demonstrate openness to diverse ideas and perspectives through active discussions to achieve a group goal.
- ◊ 9.4.8.IML.9: Distinguish between ethical and unethical uses of information and media (e.g., 1.5.8.CR3b, 8.2.8.EC.2).
- ◊ 9.4.8.TL.5: Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.

ESSENTIAL QUESTIONS

- ◊ What is computer science?
- ◊ What is a computer?
- ◊ How do humans interact with computers?
- ◊ How is data stored?
- ◊ How is information collected and used?
- ◊ What is the engineering design process?
- ◊ Why is testing important?
- ◊ Why should design be user-centered?
- ◊ How does collaboration benefit coding and programming projects?
- ◊ How can information be used to help humans create?
- ◊ How is data communicated between humans and computers?
- ◊ What are different ways data can be collected and analyzed?
- ◊ How do programmers limit the number of problems in their programs?
- ◊ Why is documentation important in coding?
- ◊ Why is there a large variety of computer languages?
- ◊ What is computational thinking?
- ◊ What principles in programming apply universally?
- ◊ How can Scratch be used to develop programming skills?
- ◊ What is Python?

◊ What are the elements of web development?

ENDURING UNDERSTANDING

◊ Computer Science has five major areas of study.
◊ Studying and understanding human-computer interaction is critical to making improvements to computing systems and computing software.
◊ People use computer programs to process information to gain insight and knowledge.
◊ Computing facilitates exploration and the discovery of connections in information.
◊ Computing enhances communications, interaction, and cognition.
◊ Computing enables innovation in nearly every field.
◊ Programs are developed, maintained, and used by people for different purposes.
◊ Computing has global effects – both beneficial and harmful – on people and society.

STUDENTS WILL KNOW

◊ A computer is a device that stores and process information.
◊ Computing systems are all the basic hardware and software that work together to make a computer run.
◊ Computer science is more about creating than consuming.
◊ Programs are sets of instructions that have been translated into commands a computer can understand (code).
◊ An algorithm is a list of steps or instructions written in human language that tells a person how to complete a task.
◊ Data is raw, unorganized facts.
◊ Analysis is organizing, describing, and understanding data.
◊ Networks are group of connected devices.
◊ The internet Is the worldwide network that connects millions of computers.
◊ The user interface includes all parts of a computing system that you use to operate the computer.
◊ A graphical user interface is a type of user interface that uses icons and symbols on a screen instead of just plain text.
◊ A command line interface uses only text to operate a computer.
◊ Troubleshooting is take a systematic, or step-by-step approach to solving errors.

- ◊ Debugging is finding and correcting errors within a program's code.
- ◊ Decomposition is breaking a problem down into simple parts.
- ◊ Pattern recognition is identifying what different problems have in common.
- ◊ Abstraction is separating details that matter from details that are not important.
- ◊ Algorithm design is creating a solution with simple steps that anyone can follow.
- ◊ A variable is a container for storing a value and its name is an identifier.
- ◊ An assignment operator is the "=" symbol, which assigns values to variables.
- ◊ String variables can store any kind of character.
- ◊ Booleans can only be true or false.
- ◊ An array stores an entire list of information.
- ◊ Condition statements run a chunk of code when a certain condition is met following the IF…THEN format.
- ◊ Loop statements allow for the repeating of a chunk of code many times.
- ◊ Events are actions that cause something to happen within a program.
- ◊ A procedure is a piece of code that can be used over and over.

STUDENTS WILL BE ABLE TO

- ◊ Identify and explain the five concept areas of computer science.
- ◊ Contrast between consumers and creators.
- ◊ Describe the impacts of computing on their daily lives.
- ◊ Define key vocabulary terms associated to each of the units.
- ◊ Contrast between software and hardware.
- ◊ Describe the four major ideas in computational thinking.
- ◊ Contrast between the different types of loop statements.
- ◊ Program in Scratch.

Code.org's Annual Express Course Curriculum

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.1 Computer Science

Computing Systems

◊ 8.1.8.CS.4: Systematically apply troubleshooting strategies to identify and resolve hardware and software problems in computing systems.

Data & Analysis

◊ 8.1.8.DA.1: Organize and transform data collected using computational tools to make it usable for a specific purpose.
◊ 8.1.8.DA.4: Transform data to remove errors and improve the accuracy of the data for analysis.
◊ 8.1.8.DA.5: Test, analyze, and refine computational models.

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.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.2 Design Thinking

Engineering Design

◊ 8.2.8.ED.2: Identify the steps in the design process that could be used to solve a problem.

CSTA COMPUTER SCIENCE K-12 STANDARDS

Algorithms & Programming

◊ 2-AP-10: Use flowcharts and/or pseudocode to address complex programs as algorithms.
◊ 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
◊ 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
◊ 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
◊ 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.
◊ 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
◊ 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
◊ 2-AP-17: Systematically test and refine programs using a range of test cases.
◊ 2-AP-18: Distribute tasks and maintain a project timeline when collaboratively developing computational artifacts.
◊ 2-AP-19: Document programs in order to make them easier to follow, test, and debug.

Data & Analysis

◊ 2-DA-08: Collect data using computational tools and transform the data to make it more useful and reliable.
◊ 2-DA-09: Refine computational models based on the data they have generated.

ISTE Standards for Students

1.1 Empowered Learner

- ◊ 1.1.a: Students articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.
- ◊ 1.1.c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
- ◊ 1.1.d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.4 Innovative Designer

- ◊ 1.4.a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- ◊ 1.4.d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

1.5 Computational Thinker

- ◊ 1.5.a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- ◊ 1.5.b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- ◊ 1.5.c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- ◊ 1.5.d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6 Creative Communicator

- ◊ 1.6.c: Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

9.3 – CAREER & TECHNICAL EDUCATION (CTE) CONTENT AREA: 21ST CENTURY LIFE AND CAREERS

◊ 9.3.IT-PRG.6: Program a computer application using the appropriate programming language.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – CAREER READINESS, LIFE LITERACIES, AND KEY SKILLS

◊ 9.4.8.CI.3: Examine challenges that may exist in the adoption of new ideas (e.g., 2.1.8.SSH, 6.1.8.CivicsPD.2).
◊ 9.4.8.CI.4: Explore the role of creativity and innovation in career pathways and industries
◊ 9.4.8.CT.1: Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).
◊ 9.4.8.CT.2: Develop multiple solutions to a problem and evaluate short- and long-term effects to determine the most plausible option (e.g., MS-ETS1-4, 6.1.8.CivicsDP.1).
◊ 9.4.8.CT.3: Compare past problem-solving solutions to local, national, or global issues and analyze the factors that led to a positive or negative outcome.
◊ 9.4.8.GCA.2: Demonstrate openness to diverse ideas and perspectives through active discussions to achieve a group goal.
◊ 9.4.8.IML.9: Distinguish between ethical and unethical uses of information and media (e.g., 1.5.8.CR3b, 8.2.8.EC.2).
◊ 9.4.8.TL.5: Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.

ESSENTIAL QUESTIONS

◊ Why is sequence important in coding?
◊ How does the debugging process help fix errors in code?
◊ How does a computer navigate algorithms?
◊ How can programming affect graphic animation?
◊ What is the benefit of using loops?
◊ How do we define when parts of code should run and when they should not?
◊ What are the different ways to simplify long sequences of code?
◊ How can data be modified within code?

ENDURING UNDERSTANDING

- ◊ Complex coding can be solved through maps, outlines, and pseudocode.
- ◊ Breaking down complex problems into smaller parts makes it easier to find solutions.
- ◊ Systematic approaches help to solve problems step-by-step.
- ◊ Documentation and journaling help programmers test and debug code.
- ◊ Compiling code and using abstraction are necessary for more efficient code-writing.
- ◊ Computing enables people to use creative development processes to create computational artifacts for creative expressions or to solve a problem.

STUDENTS WILL KNOW

- ◊ Sequencing is putting code in the correct order for computers to read
- ◊ Debugging is finding errors in code.
- ◊ Algorithms tell a computer how to navigate instructions.
- ◊ Sprites are graphics with modifiable attributes and behaviors.
- ◊ An event is an action that causes something else to happen.
- ◊ Using loops avoids the tedious and inefficient approach of repeating several lines of commands.
- ◊ Conditional code allows the program to operate in different scenarios or under various circumstances.
- ◊ Categories of code can be utilized as functions to call
- ◊ Variables modify stored data.

STUDENTS WILL BE ABLE TO

- ◊ Identify and locate bugs in a program.
- ◊ Translate movements into a series of commands.
- ◊ Modify an existing program to solve errors.
- ◊ Predict where a program will fail.

- ◊ Reflect on the debugging process in an age-appropriate way.
- ◊ Develop problem solving and critical thinking skills by reviewing debugging practices.
- ◊ Order movement commands as sequential steps in a program.
- ◊ Represent an algorithm as a computer program.
- ◊ Break complex shapes into simple parts.
- ◊ Create a program to complete an image using sequential steps.
- ◊ Create new sprites and assign them costumes and behaviors.
- ◊ Define “sprite” as a character or object on the screen that can be moved and changed.
- ◊ Create an animation using sprites, and behaviors.
- ◊ Create new sprites and assign them costumes and behaviors.
- ◊ Create an interactive animation using events.
- ◊ Develop programs that respond to timed events.
- ◊ Develop programs that respond to user input.
- ◊ Create an interactive virtual pet using events, behaviors, variables, and custom art.
- ◊ Program solutions to problems that arise when designing a virtual pet, like feeding it or monitoring its happiness.
- ◊ Create dance animations with code
- ◊ Develop programs that respond to timed events
- ◊ Develop programs that respond to user input
- ◊ Break down a long sequence of instructions into the largest repeatable sequence.
- ◊ Employ a combination of sequential and looped commands to reach the end of a maze.
- ◊ Identify the benefits of using a loop structure instead of manual repetition.
- ◊ Differentiate between commands that need to be repeated in loops and commands that should be used on their own.
- ◊ Identify the benefits of using a loop structure instead of manual repetition.
- ◊ Break complex tasks into smaller repeatable sections.
- ◊ Identify the benefits of using a loop structure instead of manual repetition.
- ◊ Recognize large, repeated patterns as made from smaller repeated patterns.
- ◊ Break apart code into the largest repeatable sequences using both loops and nested loops.
- ◊ Describe when a loop, nested loop, or no loop is needed.
- ◊ Recognize the difference between using a loop and a nested loop.
- ◊ Define circumstances when certain parts of a program should run and when they shouldn't.

- ◊ Determine whether a conditional is met based on criteria.
- ◊ Solve puzzles using a combination of looped sequences and conditionals.
- ◊ Translate spoken language conditional statements into a program.
- ◊ Distinguish between loops that repeat a fixed number of times and loops that repeat as long as a condition is true.
- ◊ Use a while loop to create programs that can solve problems with unknown values.
- ◊ Distinguish between loops that repeat a fixed number of times and loops that repeat as long as a condition is true.
- ◊ Use a while loop to create programs that can solve problems with unknown values.
- ◊ Define circumstances when certain parts of a program should run and when they shouldn't.
- ◊ Determine whether a conditional is met based on criteria.
- ◊ Build programs with the understanding of multiple strategies to implement conditionals.
- ◊ Translate spoken language conditional statements and loops into a program.
- ◊ Nest conditionals to analyze multiple value conditions using if, else if, else logic.
- ◊ Pair a loop and conditional statement together.
- ◊ Use functions to simplify complex programs.
- ◊ Use pre-determined functions to complete commonly repeated tasks.
- ◊ Recognize when a function could help to simplify a program.
- ◊ Use pre-determined functions to complete commonly repeated tasks.
- ◊ Categorize and generalize code into useful functions.
- ◊ Recognize when a function could help to simplify a program.
- ◊ Actions Use variables to hold words and phrases.
- ◊ Use variables in conjunction with prompts.
- ◊ Create a clicker game in Sprite Lab where sprites can be removed to score points
- ◊ Create a variable that stores information and changes over time
- ◊ Assign values to existing variables.
- ◊ Use variables to change values inside of a loop.
- ◊ Utilize variables in place of repetitive values inside of a program.
- ◊ Examine code to find places where variables can be substituted for specific values.
- ◊ Identify areas where they can use variables to modify quantities during runtime.
- ◊ Determine starting value, stopping value, and stepping value for a for loop.
- ◊ Recognize when to use a for loop and when to use other loops such as repeat and while loops.

- ◊ Recognize when to use a for loop and when to use other loops such as repeat and while loops.
- ◊ Use for loops to change loop several times with different values.
- ◊ Overcome obstacles such as time constraints or bugs.

Amazon Future Engineer

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.1 Computer Science

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.

Networks and the Internet

- 8.1.8.NI.1: Model how information is broken down into smaller pieces, transmitted as addressed packets through multiple devices over networks and the Internet, and reassembled at the destination.
- 8.1.8.NI.2: Model the role of protocols in transmitting data across networks and the Internet and how they enable secure and errorless communication.
- 8.1.8.NI.1: Model how information is broken down into smaller pieces, transmitted as addressed packets through multiple devices over networks and the Internet, and reassembled at the destination.
- 8.1.8.NI.2: Model the role of protocols in transmitting data across networks and the Internet and how they enable secure and errorless communication.

Impacts of Computing

- 8.1.8.IC.2: Describe issues of bias and accessibility in the design of existing technologies.

Data & Analysis

- 8.1.8.DA.5: Test, analyze, and refine computational models.
- 8.1.8.DA.6: Analyze climate change computational models and propose refinements.

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.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – COMPUTER SCIENCE AND DESIGN THINKING – 8.2 Design Thinking

Engineering Design

- ◊ 8.2.8.ED.1: Evaluate the function, value, and aesthetics of a technological product or system, from the perspective of the user and the producer.
- ◊ 8.2.8.ED.2: Identify the steps in the design process that could be used to solve a problem.
- ◊ 8.2.8.ED.4: Investigate a malfunctioning system, identify its impact, and explain the step-by-step process used to troubleshoot, evaluate, and test options to repair the product in a collaborative team.

Interaction of Technology and Humans

- ◊ 8.2.8.ITH.1: Explain how the development and use of technology influences economic, political, social, and cultural issues.
- ◊ 8.2.8.ITH.2: Compare how technologies have influenced society over time.
- ◊ 8.2.8.ITH.3: Evaluate the impact of sustainability on the development of a designed product or system.

◊ 8.2.8.ITH.4: Identify technologies that have been designed to reduce the negative consequences of other technologies and explain the change in impact.
◊ 8.2.8.ITH.5: Compare the impacts of a given technology on different societies, noting factors that may make a technology appropriate and sustainable in one society but not in another.

Nature of Technology

◊ 8.2.8.NT.3: Examine a system, consider how each part relates to other parts, and redesign it for another purpose.
◊ 8.2.8.NT.4: Explain how a product designed for a specific demand was modified to meet a new demand and led to a new product.

Effects of Technology on the Natural World

◊ 8.2.8.ETW.4: Compare the environmental effects of two alternative technologies devised to address climate change issues and use data to justify which choice is best.

Ethics & Culture

◊ 8.2.8.EC.2: Examine the effects of ethical and unethical practices in product design and development.

CSTA COMPUTER SCIENCE K-12 STANDARDS

Algorithms & Programming

◊ 2-AP-10: Use flowcharts and/or pseudocode to address complex programs as algorithms.
◊ 2-AP-11: Create clearly named variables that represent different data types and perform operations on their values.
◊ 2-AP-12: Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.
◊ 2-AP-13: Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs.
◊ 2-AP-14: Create procedures with parameters to organize code and make it easier to reuse.

◊ 2-AP-15: Seek and incorporate feedback from team members and users to refine a solution that meets user needs.
◊ 2-AP-16: Incorporate existing code, media, and libraries into original programs, and give attribution.
◊ 2-AP-17: Systematically test and refine programs using a range of test cases.
◊ 2-AP-19: Document programs in order to make them easier to follow, test, and debug.

Computing Systems

◊ 2-CS-01: Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices.
◊ 2-CS-02: Design projects that combine hardware and software components to collect and exchange data.
◊ 2-CS-03: Systematically identify and fix problems with computing devices and their components.

Data & Analysis

◊ 2-DA-07: Represent data using multiple encoding schemes.

Impacts of Computing

◊ 2-IC-20: Compare tradeoffs associated with computing technologies that affect people's everyday activities and career options.
◊ 2-IC-21: Discuss issues of bias and accessibility in the design of existing technologies.
◊ 2-IC-22: Collaborate with many contributors through strategies such as crowdsourcing or surveys when creating a computational artifact.

Networks & the Internet

◊ 2-NI-04: Model the role of protocols in transmitting data across networks and the Internet.
◊ 2-NI-05: Explain how physical and digital security measures protect electronic information.
◊ 2-NI-06: Apply multiple methods of encryption to model the secure transmission of information.

ISTE Standards for Students

1.1 Empowered Learner

- ◊ 1.1.a: Students articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.
- ◊ 1.1.c: Students use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.
- ◊ 1.1.d: Students understand the fundamental concepts of technology operations, demonstrate the ability to choose, use and troubleshoot current technologies and are able to transfer their knowledge to explore emerging technologies.

1.4 Innovative Designer

- ◊ 1.4.a: Students know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.
- ◊ 1.4.c: Students develop, test and refine prototypes as part of a cyclical design process.
- ◊ 1.4.d: Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

1.5 Computational Thinker

- ◊ 1.5.a: Students formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
- ◊ 1.5.b: Students collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
- ◊ 1.5.c: Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
- ◊ 1.5.d: Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.

1.6 Creative Communicator

1.6.c: Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations

9.3 – CAREER & TECHNICAL EDUCATION (CTE) CONTENT AREA: 21ST CENTURY LIFE AND CAREERS

◊ 9.3.IT.4: Demonstrate positive cyber citizenry by applying industry-accepted ethical practices and behaviors
◊ 9.3.IT-PRG.6: Program a computer application using the appropriate programming language.

2020 NEW JERSEY STUDENT LEARNING STANDARDS – CAREER READINESS, LIFE LITERACIES, AND KEY SKILLS

◊ 9.4.8.CI.3: Examine challenges that may exist in the adoption of new ideas (e.g., 2.1.8.SSH, 6.1.8.CivicsPD.2).
◊ 9.4.8.CI.4: Explore the role of creativity and innovation in career pathways and industries
◊ 9.4.8.CT.1: Evaluate diverse solutions proposed by a variety of individuals, organizations, and/or agencies to a local or global problem, such as climate change, and use critical thinking skills to predict which one(s) are likely to be effective (e.g., MS-ETS1-2).
◊ 9.4.8.CT.2: Develop multiple solutions to a problem and evaluate short- and long-term effects to determine the most plausible option (e.g., MS-ETS1-4, 6.1.8.CivicsDP.1).
◊ 9.4.8.CT.3: Compare past problem-solving solutions to local, national, or global issues and analyze the factors that led to a positive or negative outcome.
◊ 9.4.8.GCA.2: Demonstrate openness to diverse ideas and perspectives through active discussions to achieve a group goal.
◊ 9.4.8.IML.9: Distinguish between ethical and unethical uses of information and media (e.g., 1.5.8.CR3b, 8.2.8.EC.2).
◊ 9.4.8.TL.5: Compare the process and effectiveness of synchronous collaboration and asynchronous collaboration.

ESSENTIAL QUESTIONS

◊ What is computer science?
◊ How can we tell a computer what we want it to do?
◊ What is a growth mindset and how can it help in the world of coding?

- ◊ What is pair programming?
- ◊ How do we code in Scratch?
- ◊ What is the connection between an event and a response?
- ◊ How can we use the XY coordinate grid when communicating with a computer?
- ◊ What does it mean to debug something?
- ◊ What are some debugging strategies?
- ◊ How do broadcasting and receiving blocks work?
- ◊ What are loops and why do we use them?
- ◊ What is the difference between bitmap (raster) and vector graphics?
- ◊ How can we use loops to enhance animation features like costumes and sound?
- ◊ What is "frame rate" and how does it apply to coding and animation?
- ◊ How can websites be used to address problems in the world?
- ◊ What is a conditional statement?
- ◊ What is pseudocode?
- ◊ How does computer science impact real-world current events?
- ◊ What is a boolean operator?
- ◊ How can programs be organized so that common problems only need to be solved once?
- ◊ Who is our "audience" when we write code?
- ◊ How do we define what "successful" is when we are developing a creative program such as a game?
- ◊ What are some strategies we can use to identify where we can make our code better?
- ◊ How are the skills we use in game development also useful in everyday life?
- ◊ Why might a computer program go through several different versions before it reaches its final state?
- ◊ What is a list and how does it relate to variables?
- ◊ How are variables like Mad Libs?
- ◊ Why would we want to use the "random" command?
- ◊ How can we store multiple possible answers in a coding response?
- ◊ What are some ways we see input–output function in the real world?
- ◊ What is the difference between "hardware" and "software?"
- ◊ What are some of the artistic capabilities we have on a computer that we may not have in the non-digital world?
- ◊ In this unit, how do we combine scientific data with unique creative ideas?

◊ How can we simplify a lot of complicated information using software?

ENDURING UNDERSTANDING

◊ Computer programming has social, creative, and practical functions in the world around us.
◊ Collaboration and giving, receiving, and applying feedback are fundamental elements in creating and designing a product.
◊ Various programming features can be used to solve different problems.
◊ Computers can be used for creative expression.
◊ Coding and programming projects must go through a multi-layered process of development before completion.
◊ Technology has an impact on everyone and the environment.
◊ Writing code is similar to composing a narrative.

STUDENTS WILL KNOW

◊ Technical basics of both what a computer is and how it communicates.
◊ Different strategies for finding and solving bugs in code.
◊ Which program features are best for solving specific problems.
◊ Computer games are developed using computer programming.
◊ The dynamic nature of coding as it calls for repeated testing and feedback.
◊ How binary and variable functions of coding can broaden both the creative and practical uses of programming.
◊ The integrated capabilities that divide hardware and software.
◊ How the internet operates and how it is controlled.

STUDENTS WILL BE ABLE TO

◊ Identify traditional and non-traditional systems as computers.
◊ Solve basic coding problems through problem-solving techniques.
◊ Explain the real-world social, creative, and practical functions of computer science.
◊ Build programs in Scratch that use events to trigger responses.

- ◊ Move sprites around the Scratch Stage using the XY coordinate grid.
- ◊ Debug basic event-based programs in Scratch
- ◊ Utilize loops to create more efficient code.
- ◊ Explain the difference between bitmap (raster) and vector graphics.
- ◊ Build animations in Scratch using different frame rates of their choosing.
- ◊ Build programs in Scratch that use if-then and if-then-else conditional statements.
- ◊ Use pseudocode to plan their programming projects.
- ◊ Build programs in Scratch that use boolean operators.
- ◊ Create more complex maze game-based programs in Scratch through the use of multi-layered interactions between sprites and backdrops.
- ◊ Use environmental input to move sprites through the Scratch Stage.
- ◊ Build programs in Scratch using variables and lists to store and retrieve data.
- ◊ Explain how programming can be used to create visual art pieces.
- ◊ Build programs in Scratch that are self-drawing.
- ◊ Create procedures and custom blocks in their Scratch projects.

Stage 2 – Evidence of Learning

Apple Learn to Code 1 & 2

Formative Assessment Suggestions

- ◊ Question and Answer
- ◊ Class Discussion
- ◊ Online Discussions (Canvas, FlipGrid, Padlet, Cloud-Shared Documents)
- ◊ Journals (Written, Audio, Video)
- ◊ Small Group Discussion
- ◊ Quizzes/Tests
- ◊ Trivia Games (Kahoot!, Gimkit, Blooket, Quizziz)

Authentic Assessment Suggestions

- ◊ Hide and Seek
- ◊ Debug Like a Pro
- ◊ Pattern Maker
- ◊ Scavenger Hunt
- ◊ Hide and Seek Again
- ◊ Who's the Tallest?
- ◊ NewsBot
- ◊ Be an Architect
- ◊ Siri Says
- ◊ Fruity Arrays
- ◊ Coding Challenges/Puzzles

Benchmark Assessment Suggestions

- ◊ Apple Pages Coding Portfolio
- ◊ Milestone Project: World Building with Storytelling

The Complete Middle School Study Guide: Everything You Need to ACE Computer Science and Coding in One Big Fat Notebook

Formative Assessment Suggestions

- ◊ Question and Answer
- ◊ Class Discussion
- ◊ Online Discussions (Canvas, FlipGrid, Padlet, Cloud-Shared Documents)
- ◊ Journals (Written, Audio, Video)
- ◊ Small Group Discussion
- ◊ Quizzes/Tests
- ◊ Trivia Games

Authentic Assessment Suggestions

- ◊ The information obtained from the lessons derived from this book should be applied to the activities, challenges, projects, and puzzles in Learn to Code, Express Course, Scratch, CS First, Minecraft Edu, and RabbidsCoding.
- ◊ CS Career Research Projects

Benchmark Assessment Suggestions

- ◊ Portfolio

Code.org's Annual Express Course Curriculum

Formative Assessment Suggestions

- ◊ Question and Answer
- ◊ Class Discussion
- ◊ Online Discussions (Canvas, FlipGrid, Padlet, Cloud-Shared Documents)
- ◊ Journals (Written, Audio, Video)
- ◊ Small Group Discussion
- ◊ Quizzes/Tests
- ◊ Trivia Games (Kahoot!, Gimkit, Blooket, Quizziz)

Authentic Assessment Suggestions

- ◊ Coding Challenges/Puzzles
- ◊ Custom Maze Designs and Challenges
- ◊ Partner Debug Challenge
- ◊ The Copy Machine
- ◊ So Moving
- ◊ Connect It Back
- ◊ True/False Tag
- ◊ Nesting
- ◊ Draw by Functions

Benchmark Assessment Suggestions

- ◊ Portfolio
- ◊ End of Course Project

Amazon Future Engineer

Formative Assessment Suggestions

- ◊ Question and Answer
- ◊ Class Discussion
- ◊ Online Discussions (Canvas, FlipGrid, Padlet, Cloud-Shared Documents)
- ◊ Journals (Written, Audio, Video)
- ◊ Small Group Discussion
- ◊ Quizzes/Tests
- ◊ Trivia Games (Kahoot!, Gimkit, Blooket, Quizziz)

Authentic Assessment Suggestions

- ◊ Movie Magic!
- ◊ Where is Carmen Scratch-iego?
- ◊ Avatar Builder
- ◊ Abstract Art

Benchmark Assessment Suggestions

- ◊ Portfolio
- ◊ Intersessions Activities

Stage 3 – Learning Plan

Instructional Map

Marking Period 1

- ◊ Apple Learn to Code 1
  - Commands
  - Functions
  - For Loops
  - Conditional Code
  - Logical Operators
- ◊ Everything You Need to Ace CS and Coding in One Big Fat Notebook
  - Unit 1: Computing Systems
  - Unit 2: Data and Analysis
  - Unit 3: Software Engineering
- ◊ Code.org Express Course
  - Sequencing
  - Sprites
  - Events
- ◊ Supplemental (Optional):
  - RabbidsCoding Levels 1-10
  - Minecraft Education Edition Subject Kits
    - Computer Science
    - Digital Citizenship
    - Art & Design
    - Social Emotional Learning
  - Google CS First
    - Subject-Specific Lessons
    - Hour of Code Lessons

Marking Period 2

- ◊ Apple Learn to Code 1
  - While Loops
  - Algorithms
- ◊ Everything You Need to Ace CS and Coding in One Big Fat Notebook
  - Unit 4: Algorithms and Programming
  - Unit 5: Universal Programming Principles
- ◊ Code.org Express Course
  - Loops
  - Conditionals
- ◊ Supplemental (Optional):
  - RabbidsCoding Levels 11-20
  - Minecraft Education Edition Subject Kits
    - Computer Science
    - Digital Citizenship
    - Art & Design
    - Social Emotional Learning
  - Google CS First
    - Subject-Specific Lessons
    - Hour of Code Lessons

Marking Period 3

- ◊ Apple Learn to Code 2
  - Variables
  - Types
  - Initialization
  - Parameters
- ◊ Everything You Need to Ace CS and Coding in One Big Fat Notebook

    - Unit 6: Programming with Scratch
    - Unit 7: Programming in Python
    - Unit 8: Web Development
- ◊ Code.org Express Course
    - Functions
    - Variables
    - For Loops
- ◊ Amazon Future Engineer
    - Block A
    - Block B
- ◊ Supplemental (Optional):
    - RabbidsCoding Levels 21-30
    - Minecraft Education Edition Subject Kits
        - Science
        - Math
        - Language Arts
        - History & Culture
        - Climate & Sustainability
        - Equity & Inclusion
    - Google CS First
        - Intermediate Lessons
        - Advanced Lessons
    - Scratch
        - Unit 0: Getting Started
        - Unit 1: Exploring
        - Unit 2: Animations
        - Unit 3: Stories

Marking Period 4

- ◊ Apple Learn to Code 2

- World Building
- Arrays
- Milestone Project

◊ Code.org Express Course
- End of Course Project

◊ Amazon Future Engineer
- Block C
- Block D

◊ Supplemental (Optional):
- RabbidsCoding Levels 31-40 + Sandbox
- Minecraft Education Edition eSports Lessons
- Google CS First
  - Advanced Lessons
- Scratch
  - Unit 4: Games
  - Unit 5: Diving Deeper
  - Unit 6: Hackathon

Modifications/ Differentiation of Instruction

Modification Strategies

- ◊ Extended Time
- ◊ Frequent Breaks
- ◊ Highlighted Text
- ◊ Interactive Notebook
- ◊ Modified Test
- ◊ Oral Directions
- ◊ Peer Tutoring
- ◊ Preferential Seating
- ◊ Re-Driect
- ◊ Repeated Drill/ Practice
- ◊ Shortened Assignments
- ◊ Teacher Notes
- ◊ Tutorials
- ◊ Use of Additional Reference Material
- ◊ Use of Audio Resources
- ◊ Language Translator Tools or Translated Documents

High Preparation Differentiation

- ◊ Alternative Assessments
- ◊ Choice Boards
- ◊ Games and Tournaments
- ◊ Group Investigations
- ◊ Guided Reading
- ◊ Independent Research/ Project
- ◊ Interest Groups

- ◊ Passion Hour
- ◊ Learning Contracts
- ◊ Leveled Rubrics
- ◊ Literature Circles
- ◊ Menu Assignments
- ◊ Multiple Intelligence Options
- ◊ Multiple Texts
- ◊ Personal Agendas
- ◊ Project Based Learning
- ◊ Stations/ Centers
- ◊ Think-Tac-Toe
- ◊ Tiered Activities/ Assignments
- ◊ Varying Graphic Organizers

Low Preparation Differentiation

- ◊ Choice of Book/ Activity
- ◊ Cubing Activities
- ◊ Exploration by Interest (using interest inventories)
- ◊ Flexible Grouping
- ◊ Goal Setting with Student
- ◊ Homework Options
- ◊ Jigsaw
- ◊ Mini Workshops to Extend Skills
- ◊ Mini Workshops to Re-teach
- ◊ Open-ended Activities
- ◊ Think-Pair-Share by Learning Style
- ◊ Think-Pair-Share by Readiness
- ◊ Use of Collaboration
- ◊ Use of Reading Buddies or Coding Buddies

- ◊ Varied Journal Prompts
- ◊ Varied Product Choice
- ◊ Varied Supplemental Materials
- ◊ Work Alone/ Partner/ Group
- ◊ Alternate Materials for Accessibility (Provided by Swift and Code.org)

Horizontal Integration – Interdisciplinary Connections

Math

- ◊ Algebra
- ◊ Algorithms
- ◊ Patterns
- ◊ Measurement

Language Arts

- ◊ Written Language Syntax
- ◊ Punctuation
- ◊ Research

Science

- ◊ Engineering Design Process
- ◊ Scientific Method

History/ Civics/ Business

- ◊ Computer History
- ◊ Privacy Laws & Safety
- ◊ CS Career Exploration

Vertical Integration – Discipline Mapping

- ◊ This middle school coding curriculum has been designed to strengthen the knowledge students' obtained from their experiences with basic coding activities, concepts, and games at the elementary level.
- ◊ This middle school coding curriculum has been designed to build students' interest and foundational knowledge to make an informed decision when selecting a computer science and cybersecurity pathway in the PLTW program adopted in the Linden Public Schools high school.

Additional Materials

- ◊ Apple Learn to Code: XCode
- ◊ TelloEDU Drones
- ◊ MakeBlock Drones
- ◊ MakeBlock Robots
- ◊ CUE Robots
- ◊ Sphero EDU and Sphero PLAY Robots
- ◊ MakeyMakey