NEW MILFORD PUBLIC SCHOOLS

New Milford, Connecticut

Introduction to Programming

April 2021

BOE Approved August 2021

BOE Approved August 2021

Author of Course Guide

Shana Bergonzelli-Graham

BOE Approved August 2021

New Milford’s Mission Statement

The mission of the New Milford Public Schools, a collaborative partnership of students, educators, family and community, is to prepare each and every student to compete and excel in an ever-changing world, embrace challenges with vigor, respect and appreciate the worth of every human being, and contribute to society by providing effective instruction and dynamic curriculum, offering a wide range of valuable experiences, and inspiring students to pursue their dreams and aspirations.

Introduction to Programming

9-12

The focus of this course will be on problem-solving and introducing students to computational thinking in a fun, hands-on way. The emphasis will be on identifying common patterns in programs such as sequence, decision-making and repetition by analyzing existing programs and games. Students will be given various coding problems to solve and develop games and applications in the process.

BOE Approved August 2021

Pacing Guide

1	Algorithms, Sequence and the Problem Solving Process	4 Weeks (11 Days) • Coding environment, Algorithms and Problem Solving Process Intro (2 Days) • Positioning and Sizing Simple Shapes (1.5 days) • Using Colors and Fills with shapes (1.5 Days) • More Advanced Shapes, Text and Sequence (2 Days) • Cumulative End of Unit Exercises (1 Day) • Collaborative and Individual Creative Tasks (2 Days) • Review/Quizzes (1 Days)
2	Variables, Mathematical operations, Debugging and Input (mouse input)/Output	4 Weeks (10 Days) • Imitation Game Movie + Questions(3 days) • Defining and Calling Functions (1 Day) • Handling Mouse Click Events and Variables(1 Day) • More variables and Shape Properties (1 Day) • Cumulative End of Unit Exercises (1 Day) • Collaborative and Individual Creative Tasks (2 Days) • Review/Quizzes (1 Days)
3	Conditionals and Helper functions	3 Weeks (8 Days) • Robot Exercises (1 day) • Handling Mouse Move Events(.5 Days) • Simple Conditionals (if and if-else (1 Day) • Defining and Calling Helper Functions (1.5 Days) • Cumulative End of Unit Exercises (1 Day) • Collaborative and Individual Creative Tasks (2 Days) • Review/Quizzes (1 Day)
4	Methods, Multi-branched Conditionals and Key Input	3.5 Weeks (9 Days) • Handling Key Events (1 Day) • Complex (Multi-branched) Conditionals (1 Day) • Calling Methods(1 Day) • Cumulative End of Unit Exercises (2 Days) • Collaborative and Individual Creative Tasks (2 Days) • Review/Quizzes (1 Day)
5	Complex Conditionals	1.5 weeks (6 Days) • Compound Conditionals (Logical Expressions) (1 Day) • Complex Key Events (1 Day) • Cumulative End of Unit Exercises (1 Day) • Collaborative and Individual Creative Tasks (2 Days)

		● Review/Quizzes (1 Day)
6	Review, Current Events and Final Project	4 Weeks (10 Days) ● Current Event Projects (4 Days) ● Final Cumulative Project Design, code and Writeup(4 Days) ● Final Exam Review (2 days)

Unit 1: Algorithms, Sequence and the Problem Solving Process

Stage 1 Desired Results		
ESTABLISHED GOALS **CSTA K-12 Computer Science Standard**s 2-AP-10 Use flowcharts and/or pseudocode to address complex problems as algorithms. 2-AP-13 Decompose problems and subproblems into parts to facilitate the design, implementation, and review of programs. 3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests 3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools. 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	***Transfer*** *Students will be able to independently use their learning to…* Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS *Students will understand that…* ● Computer Programming is about solving problems. ● It is important to thoroughly understand the problem one is trying to solve and clarify assumptions before going about solving it. ● Planning and designing code is an important step of the problem solving process. ● We apply concepts of programming in our everyday lives. ● Computer programmers should always consider the ethical and social implications related to technology use. ● Computer programs are executed sequentially	ESSENTIAL QUESTIONS *Students will keep considering…* ● How can algorithms lend themselves to reusability? ● How do we use algorithms every day, and in what ways are they incorporated into programming? ● How can learning to program affect the outlook one has on how to complete tasks and solve problems? ● In what ways can a computer programmer's tasks be compared to other occupations such as architects and artists? ● What makes a problem hard or easy for a computer to solve?

	Acquisition	
	Students will know… ● The four steps of the problem solving process and how to apply it. ● Key terminology related to algorithms, the problem solving process and sequence. ● How to use an IDE and a debugger to compile, execute and test programs. ● Operations that computers can easily execute vs. ones that are not as easily executed (application of a heuristic). ● How text and numbers are represented in code.	*Students will be skilled at…* ● Determining what is/is not a computer program (ie, a video game or an app is a computer program. A photo stored on a smartphone is not.) ● Conceiving a sequence of steps to solve a problem individually. ● Relating the logical structures in programming algorithms to real-life situations. ● Working together in a group to decompose a problem and synthesize necessary steps for solving a problem with brevity and clarity. ● Writing code that solves a graphics-based problem using sequence according to given specifications. ● Expressing sequence and program stop/start in a program using a flowchart. ● Using number and text literals within simple function calls to solve problems. ● Identifying and fixing common syntax errors involving sequence and simple commands. ● Evaluating the output of programs that include sequence and number/text literals

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that uses sequence and graphical concepts to render an image A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details re:how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that renders an image Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Student draws on their own interests to brainstorm, design, code and test a program that uses graphical calls to render an image of their choosing using shapes, colors and other graphical elements Products and performances generated by student: A complete, error-free program that uses sequence and graphical concepts to correctly render an image of the student's choosing Standards/criteria for judging success: According to Rubric

A M, A M, A A A M, A	Precise use of computing vocabulary in context Correct application of the sequential structure as evidenced by a successful program compilation and comparison of actual output with expected output Effective application of the problem-solving process to brainstorm, design, code and test a program Correct differentiation between what is/is not a computer program Correct differentiation between problems that are easily/not easily solved by computers Correct analysis of the output of a given program or code snippet	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Summative coding challenges based on unit content. ● Individual and Collaborative Creative Tasks (detailed above) ● Hands on Python Finch robot problems related to unit content ● Online quiz consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
	Summary of Key Learning Events and Instruction	Progress Monitoring
A A A M, T A A M M, T A A M, T	● Teacher presentation of a slidedeck with graphical and visual. examples of coding concepts as well as important vocabulary. ● Student completion of a graphic organizer that summarizes characteristics of a computer program and categorizes examples from a list. ● Student collaboration to solve brainteasers involving sequence and to determine the steps needed to solve the problem correctly and efficiently. ● Student completion of a list of steps for a simple activity such as making toast followed by decomposition of the problem (ie, making toast) into steps, synthesis of individual steps and creation of a poster that demonstrates a clear, easy to understand way to solve the problem. ● Teacher demonstration of how to use the coding environment to compile programs, identify errors, and execute programs. ● Teacher demonstration of programs that involve sequence with student discussion of how the programming statements affect the output. ● Teacher-provided examples of correct solutions to problems involving sequence (coding and brainteaser/logical)a well as alternatives. ● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to sequence. ● Student comparison of the output of their program with expected output to determine logical errors. ● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code. ● Students pair/group to discuss and correct completed work.	● Question of the Day and interactive questions embedded in slidedeck ● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring) ● Interactive notes and checkpoints on programming topics ● Interactive questioning competitions such as Kahoot or Quizlet ● Exit Ticket Answers ● Summative assessments (quizzes, unit tests)

Unit 2: Variables, Mathematical operations, Debugging and Input (mouse input)/Output

Stage 1 Desired Results		
ESTABLISHED GOALS **CSTA K-12 Computer Science Standard**s 2-AP-11 Create clearly named variables that represent different data types and perform operations on their values 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. 3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions 3A-AP-22 Design and develop	***Transfer*** *Students will be able to independently use their learning to…* Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS ● Programs should be written in such a way that they can be understood by programmers other than the author. ● An important part of programming is identifying and fixing errors in code (debugging). ● Input and output are important concepts to consider when determining the flow of information. ● Functions are shortcuts for the statements they contain and make our code easier to understand, maintain and debug. ● Information is represented in programs via data structures like variables. ● Data structures can represent different kinds of data (numerical, text, etc). ● Abstraction of the data structures in a computer program means that the structures themselves	ESSENTIAL QUESTIONS ● When developing programs, how do we present necessary information in a clear way to enhance the user experience? ● Why are coding conventions important to follow? ● How do functions help us decompose problems and reduce code complexity? ● How are variables used to maintain and update program state? ● How is basic math used within programs to solve problems?

computational artifacts working in team roles using collaborative tools 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs. 3A-IC-24 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices	can stand for any relevant value. ● Giving our program good test cases based on the program specifications allows us to thoroughly test necessary functionality ● We can modify object properties based on program specifications to solve a problem ● Mathematical operations can be used to transform data within a program ● Stereotypes in computer science are prevalent but not indicative of reality	
	Acquisition	
	Students will know… ● Key terminology related to functions, mouse press/release input events, object properties and variables ● Necessary keywords for defining variables and defining/using functions ● The rules for naming functions ● The rules for naming variables ● The rules for including statements within a function ● How to define a variable correctly ● How to define a function correctly ● How to use a function/variable name to clearly communicate its purpose ● How top-level code and functions differ ● What the purpose of the pass statement is ● How to define event handlers that are called when mouse press/release events occur ● How parameters are use to pass information to functions when they are called ● How variables and literals can both be passed to functions as parameters ● How to express basic arithmetic operations within programs	*Students will be skilled at…* ● Documenting and formatting their code so it is easily understandable by other programmers. ● Inputting mouse position information from the user on a click/release ● Defining functions to group statements together in order to reduce code complexity ● Using functions within top-level code ● Expressing flow of control through functions in a program using a flowchart ● Defining and using variables to store information and maintain program state ● Defining simple test cases to test program functionality ● Identifying and fixing common syntax errors involving functions, mouse press/release input events, object properties and variables . ● Printing to the console to debug logical and runtime program errors ● Identifying what information functions need to meet their specifications ● Using parameters to pass necessary information to functions

	● How the personal computer and computer science have evolved considerably since their inception	● Using arithmetic operators within programs to transform data ● Evaluating the output of code that includes variables, function calls and arithmetic operations

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that uses variables, functions, mouse input event handlers and changing object properties to create an interactive graphical program A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free The program must incorporate functions to reduce code complexity and explain rationale for doing so Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details outlining how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that accepts user input and changes the properties of graphical objects within the program in response to the input. The program must also incorporate functions effectively and notes must demonstrate how test cases were used to test functionality Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Students draw on their own interests to brainstorm, design, code and test a program that accepts user input and changes the properties of graphical objects within the program in response to the input. Products and performances generated by students: A complete, error-free program that accepts user input and changes the properties of graphical objects within the program in response to the input. The program must also incorporate functions and a list of test cases used to test functionality must be provided. Standards/criteria for judging success: According to Rubric

A M, A M, A A A M, A M, A	Precise use of computing vocabulary in context Correct application of function definition and calls, variable definition and usage, object property modification and mouse press/release event handlers as evidenced by a successful program compilation and comparison of actual output with expected output Effective application of the problem-solving process to brainstorm, design, code and test a program Correct differentiation between function and top-level code Correct application of function and variable naming rules Effective generation of test cases based on program specifications that allow for thorough testing of program functionality Correct analysis of the output of a given program or code snippet	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Summative coding challenges based on unit content. ● Individual and Collaborative Creative Tasks (detailed above) ● Hands on Python Finch robot problems related to unit content ● Online quiz consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan

Code	***Pre-Assessment***	
	● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
  A A A M M, T A A M, T M, T	Summary of Key Learning Events and Instruction ● Teacher presentation of a slidedeck with graphical and visual examples of coding concepts as well as important vocabulary. ● Student collaboration to solve brain teasers involving functional decomposition or variables and determine the steps needed to solve the problem correctly and efficiently. ● Teacher demonstration of programs that involve variables, functions and mouse events with student discussion of how the programming statements affect the output. ● Teacher-provided examples of correct solutions to problems involving variables, mouse input, functions and object properties(coding and brainteaser/logical)as well as alternatives. ● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to variables, functions or mouse events. ● Student comparison of the output of their program with expected output to determine logical errors. ● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code. ● Students pair/group to discuss and correct completed work. ● Students watch and discuss a movie that discusses computer science concepts and history (such as the Imitation Game) to examine how Computer Science has evolved as a discipline and how inherent biases exist within it.	Progress Monitoring ● Question of the Day and interactive questions embedded in slidedeck ● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring) ● Interactive notes and checkpoints on programming topics ● Interactive questioning competitions such as Kahoot or Quizlet ● Exit Ticket Answers ● Summative assessments (quizzes, unit tests)

Stage 1 Desired Results		
ESTABLISHED GOALS	***Transfer***	
CSTA K-12 Computer Science Standards 2-AP-14 Create procedures with parameters to organize code and make it easier to reuse. 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. 3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions	*Students will be able to independently use their learning to…* Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS *Students will understand that…* ● Functionality that is implemented on a mouse move/drag is different than what would be implemented on a press/release ● It is essential that programs make decisions in order to implement necessary functionality. ● Conditional statements are how programs make decisions ● Decisions in a program can have one or more than one path. ● Relational operators are used to compare numerical values to make decisions. ● Helper functions can be called from within other functions to further decompose problems	ESSENTIAL QUESTIONS *Students will keep considering…* ● How do different mouse events control what happens in a real-life program? ● How can programmers use mouse events to effectively serve a program's purpose? ● How can I use boolean logic in real life to make decisions and solve problems? ● How can I write code that uses conditionals in a way that makes my program easy to understand?

	Acquisition	
3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects. 3A-AP-18 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs. 3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	*Students will know…* ● Key terminology related to conditionals, helper functions and mouse drag/move events ● Necessary keywords for defining conditionals and helper functions ● How to correctly define a boolean expression using relational operators ● What the six relational operators are and how to express them in code ● How to correctly define an if statement in code ● How to indent statements inside of if structures to contain the statements within the body of the structure ● How to differentiate between statements in a function or top-level code that are/are not included within an if statement ● How to define event handlers that are called when mouse move/drag events occur ● How the mouse position can be stored/updated as the result of contiguous mouse move/drag events	*Students will be skilled at…* ● Documenting and formatting their code so it is easily understandable by other programmers. ● Inputting mouse position information from the user on a move/drag ● Using, storing and updating mouse position data on a move/drag event to implement necessary program functionality ● Writing if statements using boolean expressions to make decisions in a program that have one or two paths ● Expressing a one or two-path decision in a program using a flowchart ● Determining when it is appropriate to define helper functions ● Using helper functions within other functions to reduce code complexity ● Evaluating the output of code that includes conditionals and helper functions

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that mouse move/drag event handlers that require decisions to be made using if statements in an interactive graphical program A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free The program must incorporate if statements and specify the differing output of the program based on the results of the if statement Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details outlining how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that accepts user input and makes a decision regarding the output of the programming response to the input entered. Notes must demonstrate how test cases were used to test functionality Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Students draw on their own interests to brainstorm, design, code and test a program that accepts user input as mouse moves or drags and makes a decision in the program using an if statement based on the value of the input. Products and performances generated by students: A complete, error-free program that accepts user input as mouse moves or drags and makes a decision in the program using an if statement based on the value of the input. A list of test cases used to test functionality must be provided. Standards/criteria for judging success: According to Rubric

A M, A M, A A M, A M, A	Precise use of computing vocabulary in context Correct application of if statements and mouse move/drag event handlers as evidenced by a successful program compilation and comparison of actual output with expected output Effective application of the problem-solving process to brainstorm, design, code and test a program Correct application of if statement structure and indentation Effective generation of test cases based on program specifications that allow for thorough testing of program functionality Correct analysis of the output of a given program or code snippet	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Summative coding challenges based on unit content. ● Individual and Collaborative Creative Tasks (detailed above) ● Hands on Python Finch robot problems related to unit content ● Online quiz consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
A A A M M, T A A M, T	Summary of Key Learning Events and Instruction ● Teacher presentation of a slidedeck with graphical and visual examples of coding concepts as well as important vocabulary. ● Student collaboration to solve brain teasers involving decisions or conditional statements and determine the steps needed to solve the problem correctly and efficiently. ● Teacher demonstration of programs that involve if statements and mouse move/drag events with student discussion of how the programming statements affect the output. ● Teacher-provided examples of correct solutions to problems involving if statements and mouse move/drag events as well as alternatives. ● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to if statements and mouse move/drag events. ● Student comparison of the output of their program with expected output to determine logical errors. ● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code. ● Students pair/group to discuss and correct completed work.	Progress Monitoring ● Question of the Day and interactive questions embedded in slidedeck ● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring) ● Interactive notes and checkpoints on programming topics ● Interactive questioning competitions such as Kahoot or Quizlet ● Exit Ticket Answers ● Summative assessments (quizzes, unit tests)

Unit 4: Object Methods, Multi-branched Conditionals, and Key Input

Stage 1 Desired Results		
ESTABLISHED GOALS **CSTA K-12 Computer Science Standard**s 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. 3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions 3A-AP-18 Create artifacts by using	***Transfer***	
	Students will be able to independently use their learning to… Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS *Students will understand that…* ● Multi-branched conditional statements allow for more than one or two options for the result of a decision ● Sequence is important when ordering boolean expressions in a multi-branched conditional statement ● Subsequent branches in a multi-branched conditional are only evaluated if the boolean expressions of the previous branches evaluate to false ● Key events are another type of input that programs can accept ● Specific methods for specific objects exist in order to clearly restrict what operations can be	ESSENTIAL QUESTIONS *Students will keep considering…* ● How do different key events control what happens in a real-life program? ● How can programmers use key events to effectively serve a program's purpose? ● How can we apply the logic of multi-branched conditionals in our daily lives? ● How can I write code that uses multi-branched conditionals in a way that makes my program easy to understand? ● How does object-oriented programming help us maintain and update state for objects used in out programs?

procedures within a program, combinations of data and procedures, or independent but interrelated programs. 3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	performed on what class of object ● Certain methods are global to all objects	
	Acquisition	
	Students will know… ● Key terminology related to multi-branched conditionals, methods, object oriented programming and key press/release events ● Necessary keywords for defining multi-branched conditionals, custom properties for objects/apps and calling methods ● How to correctly define multi-branched conditionals using the if-elif-else structure ● How to define a default branch for a multi-branched conditional using the else clause ● How to indent statements inside of if-elif structures to contain the statements within the body of the structure ● How to define a custom property for an object that stores necessary information ● How to call methods on objects and global methods on the app object. ● How to use special methods that return boolean values to obtain information on a given object such as whether a point is located within a shape. ● How to define event handlers that are called when key press/release events occur	*Students will be skilled at…* ● Documenting and formatting their code so it is easily understandable by other programmers. ● Inputting key state information from the user on a press/release ● Using, storing and updating key state data on a press/release event to implement necessary program functionality ● Writing multi-branched if statements using boolean expressions to make decisions in a program that have more than two paths ● Expressing a multi-branched decision in a program using a flowchart ● Identifying and fixing common syntax errors involving multi-branched if statements, key press/release input events, custom object properties and methods. ● Changing properties on an object in code in response to events that occur in the program ● Calling methods on objects to maintain or update object state ● Calling methods that return a boolean value and using the returned value in a meaningful way in the program ● Evaluating the output of code that includes a multi-branched conditional

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that uses key press/release event handlers that require decisions to be made using multi-branched if statements and custom object properties in an interactive graphical program A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free The program must incorporate multi-branched if statements and specify the differing output of the program based on the results of the multi-branched if statement Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details outlining how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that accepts key input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a multi-branched conditional and must involve using or changing the value of custom object properties. Notes must demonstrate how test cases were used to test functionality Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Students draw on their own interests to brainstorm, design, code and test a program that accepts key input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a multi-branched conditional and must involve using or changing the value of custom object properties. Products and performances generated by students: A complete, error-free program that accepts key input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a multi-branched conditional and must involve using or changing the value of custom object properties. A list of test cases used to test functionality must be provided. Standards/criteria for judging success: According to Rubric

A M, A M, A A M, A M, A	Precise use of computing vocabulary in context Correct application of multi-branched if statements key press/release event handlers and custom object properties as evidenced by a successful program compilation and comparison of actual output with expected output Effective application of the problem-solving process to brainstorm, design, code and test a program Correct application of multi-branched if statement structure and indentation Effective generation of test cases based on program specifications that allow for thorough testing of program functionality Correct analysis of the output of a given program or code snippet	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Summative coding challenges based on unit content. ● Individual and Collaborative Creative Tasks (detailed above) ● Hands on Python Finch robot problems related to unit content ● Online quiz consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
	Summary of Key Learning Events and Instruction	Progress Monitoring
A	● Teacher presentation of a slidedeck with graphical and visual examples of coding concepts as well as important vocabulary.	● Question of the Day and interactive questions embedded in slidedeck
A	● Student collaboration to solve brain teasers involving multi-branched decisions or conditional statements and determine the steps needed to solve the problem correctly and efficiently.	● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring)
A	● Teacher demonstration of programs that involve multi-branched if statements, app/object properties, methods and key press/release events with student discussion of how the programming statements affect the output.	● Interactive notes and checkpoints on programming topics ● Interactive questioning competitions such as Kahoot or Quizlet
M	● Teacher-provided examples of correct solutions to problems involving multi-branched if statements, app/object properties, methods and key press/release events as well as alternatives.	● Exit Ticket Answers ● Summative assessments (quizzes, unit tests)
M, T	● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to variables, functions or mouse events.	
A	● Student comparison of the output of their program with expected output to determine logical errors.	
A	● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code.	
M, T	● Students pair/group to discuss and correct completed work.	

Stage 1 Desired Results		
ESTABLISHED GOALS **CSTA K-12 Computer Science Standard**s 2-AP-12 Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals 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. 3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using	***Transfer*** *Students will be able to independently use their learning to…* Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS *Students will understand that…* ● Logical operators are used to create complex conditionals that combine multiple boolean expressions ● Nested or compound conditionals can be used when the result of one boolean expression must be tested first before the result of another can be analyzed. ● Often we can create a conditional equivalent to a nested or compound conditional by using logical operators and sequence. ● Key hold events are another type of input that programs can accept	ESSENTIAL QUESTIONS *Students will keep considering…* ● How do different key events control what happens in a real-life program? ● How can programmers use key events to effectively serve a program's purpose? ● How can we apply the logic of complex and compound conditionals in our daily lives? ● How do we effectively use the words AND, OR and NOT to change the meaning of what we are communicating? ● How can I write code that uses compound and complex conditionals in a way that makes my program easy to understand?

events to initiate instructions 3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools 3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.	***Acquisition***	
	Students will know… ● Key terminology related to complex and compound conditionals, logical operators and key hold events ● Necessary keywords for complex and compound conditionals ● How to define and structure complex and compound conditionals ● How to use indentation to make it clear which statements belong to which conditionals ● How to define an event handler for a key hold event. ● What the three common logical operators are and how they are evaluated ● How the three logical operators are used to transform the meaning of boolean expressions ● The logical difference between exclusive and inclusive or. ● How to express a given programming scenario using AND, OR and NOT logic in a boolean expression	*Students will be skilled at…* ● Documenting and formatting their code so it is easily understandable by other programmers. ● Inputting key state information from the user on a key hold ● Using, storing and updating key state data on a key hold event to implement necessary program functionality ● Writing complex and nested if statements using boolean expressions to make decisions in a program that have codependencies or more than one criterion to test ● Identifying and fixing common syntax errors involving complex and compound if statement and key hold events. ● Expressing complex and compound conditionals in a program using a flowchart ● Evaluating the output of code that includes compound and complex conditionals

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that uses key hold event handlers that require decisions to be made using complex and compound conditionals in an interactive graphical program A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free The program must incorporate multi-branched if statements and specify the differing output of the program based on the results of the compound or complex if statement Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details outlining how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that accepts key hold input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a complex or compound if statement and must involve using or changing the value of custom object properties. Notes must demonstrate how test cases were used to test functionality Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Students draw on their own interests to brainstorm, design, code and test a program that accepts key hold input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a complex or compound if statement and must involve using or changing the value of custom object properties Products and performances generated by students: A complete, error-free program that accepts key hold input and makes a decision regarding the output of the programming response to the input entered. The decision must be in the form of a complex or compound conditional and must involve using or changing the value of custom object properties. A list of test cases used to test functionality must be provided. Standards/criteria for judging success: According to Rubric

A M, A M, A A M, A M, A	Precise use of computing vocabulary in context Correct application of complex and compound if statements, use of logical operators in boolean expressions and key hold event handlers as evidenced by a successful program compilation and comparison of actual output with expected output Effective application of the problem-solving process to brainstorm, design, code and test a program Correct application of complex and compound if statement structure and indentation as well correct use of logical operators in the statement's boolean expression Effective generation of test cases based on program specifications that allow for thorough testing of program functionality Correct analysis of the output of a given program or code snippet	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Summative coding challenges based on unit content. ● Individual and Collaborative Creative Tasks (detailed above) ● Hands on Python Finch robot problems related to unit content ● Online quiz consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
	Summary of Key Learning Events and Instruction	Progress Monitoring
A A A M M, T A A M, T	● Teacher presentation of a slidedeck with graphical and visual examples of coding concepts as well as important vocabulary. ● Student collaboration to solve brain teasers involving complex or compound decisions or conditional statements and determine the steps needed to solve the problem correctly and efficiently. ● Teacher demonstration of programs that involve complex or compound if statements and key hold events with student discussion of how the programming statements affect the output. ● Teacher-provided examples of correct solutions to problems involving complex or compound if statements and key hold events as well as alternatives. ● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to compound and complex if statements and key hold events. ● Student comparison of the output of their program with expected output to determine logical errors. ● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code. ● Students pair/group to discuss and correct completed work.	● Question of the Day and interactive questions embedded in slidedeck ● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring) ● Interactive notes and checkpoints on programming topics ● Interactive questioning competitions such as Kahoot or Quizlet ● Exit Ticket Answers ● Summative assessments (quizzes, unit tests)

Stage 1 Desired Results		
ESTABLISHED GOALS **CSTA K-12 Computer Science Standard**s 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. 3A-AP-15 Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made 3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions 3A-AP-21 Evaluate and refine computational artifacts to make them more usable and accessible.	***Transfer***	
	Students will be able to independently use their learning to… Collaborate with peers or others to solve problems and to develop solutions using technology tools and resources. Use logical and reasoning skills to solve problems in school and real-life Consider implications of personal and professional decisions involving technology	
	Meaning	
	UNDERSTANDINGS *Students will understand that…* ● No new material: Review Unit	ESSENTIAL QUESTIONS *Students will keep considering...* ● No new material: Review Unit
	Acquisition	
	Students will know… No new material: Review Unit	*Students will be skilled at…* No new material: Review Unit

3A-AP-22 Design and develop computational artifacts working in team roles using collaborative tools 3A-IC-24 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices		

Stage 2 – Evidence		
Code	**Evaluative Criteria**	**Assessment Evidence**
M, T	Visual example of planning process when designing and brainstorming the idea for a program that synthesizes all concepts taught in an interactive graphical program A program that accurately fulfills the design detailed in the planning process, is adequately documented and error-free The program must incorporate most concepts taught correctly and be designed so it accepts multiple forms of input and responds differently using multi-branched, complex, and compound conditionals to make programmatic decisions. The program should define custom object properties that are accessed/changed and methods that are called in response to decisions made and use helper functions to reduce the complexity of the code. Accurate explanation of programming concepts learned and how they were applied in the creative project Detailed description of difficulties/opportunities encountered during the problem solving process and specific details outlining how they were resolved or incorporated Detailed list of further enhancements that could be added to their program to increases functionality/user experience in their creative projects	PERFORMANCE TASK(S): *Students will show that they really understand evidence of…* Goal/challenge: A Graphical program that synthesizes most concepts learned over the course of the semester in a cohesive way. Notes must demonstrate how test cases were used to test functionality Role for student: Programmer/Tester Audience for student work: Users of the Software Situation: Students draw on their own interests to brainstorm, design, code and test a program that synthesizes most concepts learned over the course of the semester in a cohesive way. Products and performances generated by students: A complete, error-free program that synthesizes most concepts learned over the course of the semester in a cohesive way. A list of test cases used to test functionality must be provided. Standards/criteria for judging success: According to Rubric

A M, A A M, A M, T M, T	Precise use of computing vocabulary in context Effective application of the problem-solving process to brainstorm, design, code and test a program Correct application of complex and compound if statement structure and indentation as well correct use of logical operators in the statement's boolean expression Effective generation of test cases based on program specifications that allow for thorough testing of program functionality Ability to find, research and present a technology-based current event that includes an article and a video Ability to attend to a presentation, recall important information from the presentation and support opinions and claims with facts from an article and video	OTHER EVIDENCE: ● Monitoring class work through board work, group work, questioning, and circulation ● Check for understanding by going over code examples, and responding to exit tickets ● Differentiate through purposeful or flexible grouping, pair programming and use of visuals/manipulatives. ● Leveled assignments are offered for students who need remediation or more challenge. ● Current Event presentations (graded via rubric) ● Current Event student response questions based on article and video content ● Final summative Individual and Collaborative Creative Tasks (detailed above) ● Final exam consisting of multiple choice questions based on vocabulary and code analysis and open-ended coding challenges based on unit content.

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Practice Final to determine student study goals	
	Summary of Key Learning Events and Instruction	Progress Monitoring
A, T M M, T A A	● Student presentation of an emerging technology that solves a problem or an ethical issue involving technology ● Teacher reviews content by providing examples of correct solutions to summative problems and multiple choice practice questions. ● Student engagement in pair programming using the navigator/driver paradigm to solve coding problems related to summative material. ● Student comparison of the output of their program with expected output to determine logical errors. ● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code.	● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring) ● Interactive questioning competitions such as Kahoot or Quizlet ● Exit Ticket Answers ● Summative assessments (Final Exam)

Stage 3 – Learning Plan		
Code	***Pre-Assessment*** ● Check for prerequisite and prior knowledge via daily warm-up QOTD and questioning activities ● Teacher front-loads students with necessary vocabulary via guided questions and checks for understanding when introducing the topic.	
	Summary of Key Learning Events and Instruction	Progress Monitoring
A	● Teacher presentation of a slidedeck that summarizes the requirements of the final project	● Question of the Day and interactive questions embedded in slidedeck
M, T	● Students present a synopsis of a recent current event that discusses how a critical problem was solved with technology OR dissects an ethical situation involving technology	● Coding practice in an integrated development environment such as CMU CS Academy with teacher observation (embedded comments and real-time monitoring)
M, T	● Students discuss the issues for each current event in a socratic seminar environment. Open-ended questions are posed by the student presenter and the teacher.	● Exit Ticket Answers ● Capstone project-based assessment
M	● Students follow all four steps of the problem solving process to brainstorm, design, execute and test their program code	
M	● Student comparison of the output of their program with expected output to determine logical errors.	
M, T	● Student analysis of errors generated at compile-time to identify and correct syntax errors in their code.	
M, T	● Students present their programs and discuss the purpose of their programs, demonstrate functionality using the input/output of the program and outline their methodology for brainstorming, designing, executing and testing their programs.	