Union County Educational Services Commission
High School Course Syllabus

Title: Introduction to Computer Science

Timeline: Full Year; 5 Credits

Course Description:
Introduction to Computer Science empowers students to create authentic artifacts and engage with computer science as a medium for creativity, communication, problem solving, and fun. The first semester of Introduction to Computer Science introduces students to computer science as a vehicle for problem solving, communication, and personal expression. As a whole, this semester focuses on the visible aspects of computing and computer science, and encourages students to see where computer science exists around them and how they can engage with it as a tool for exploration and expression. Where the first semester centers on the immediately observable and personally applicable elements of computer science, the second semester asks students to look outward and explore the impact of computer science on society. Students will see how a thorough user-centered design process produces a better application, how data is used to address problems that affect large numbers of people, and how physical computing with circuit boards allows computers to collect input and return output in a variety of ways.

Course Outline:

I. Sequencing
II. Sprites & Events
III. Commands
IV. Planning

Refer to the attached curriculum map for a detailed outline of course objectives.

Curriculum Alignment:
New Jersey Student Learning Standards - Computer Science & Design Thinking
New Jersey Student Learning Standards - Career Readiness, Life Literacies & Key Skills

Grading Procedures:

Do Now	10%
Participation	20%
Class Assignments	50%
Assessments	20%

Adoption Date:
June 2024

Union County Educational Services Commission
Curriculum Mapping– Introduction to Computer Science

	Unit 1	**Unit 2**	**Unit 3**	**Unit 4**
Number of Weeks	approx.. 10 weeks	approx.. 10 weeks	approx.. 10 weeks	approx.. 10 weeks
Topic	Sequencing	Sprites & Events	Loops & Conditionals	Functions & Variables
Essential Question	How do puzzles help us to understand the world around us?	How can we express ourselves creatively?	How can we develop solutions to complex problems?	How can we account for different possible outcomes while planning?
Big Ideas	Problem-Solving	Programming	Commands	Planning
Standards	8.1.12.DA.1: Create interactive data visualizations using software tools to help others better understand real world phenomena 8.1.12.DA.6: Create and refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process. CLKS- Use technology to enhance productivity increase collaboration and communicate effectively Students find and maximize the productive value of existing and new technology to accomplish workplace tasks and solve workplace problems. They are flexible and adaptive in acquiring new technology. They are proficient with ubiquitous technology applications. They understand the	8.1.12.DA.6: Create and refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process. 8.1.12.AP.1: Design algorithms to solve computational problems using a combination of original and existing algorithms. 8.1.12.AP.7: Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users. CLKS- Use technology to enhance productivity increase collaboration and communicate effectively Students find and maximize the productive value of existing and new technology to accomplish workplace	8.1.12.AP.1: Design algorithms to solve computational problems using a combination of original and existing algorithms. 8.1.12.AP.2: Create generalized computational solutions using collections instead of repeatedly using simple variables. 8.1.12.AP.5: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects 8.1.12.AP.7: Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users. CLKS- Use technology to enhance productivity	8.1.12.AP.2: Create generalized computational solutions using collections instead of repeatedly using simple variables. 8.1.12.AP.5: Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects 8.1.12.AP.7: Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users. 8.2.12.NT.1: Explain how different groups can contribute to the overall design of a product. 8.1.12.CS.4: Develop guidelines that convey systematic troubleshooting

	inherent risks-personal and organizational-of technology applications, and they take actions to prevent or mitigate these risks.	tasks and solve workplace problems. They are flexible and adaptive in acquiring new technology. They are proficient with ubiquitous technology applications. They understand the inherent risks-personal and organizational-of technology applications, and they take actions to prevent or mitigate these risks.	increase collaboration and communicate effectively Students find and maximize the productive value of existing and new technology to accomplish workplace tasks and solve workplace problems. They are flexible and adaptive in acquiring new technology. They are proficient with ubiquitous technology applications. They understand the inherent risks-personal and organizational-of technology applications, and they take actions to prevent or mitigate these risks.	strategies that others can use to identify and fix errors. CLKS- Use technology to enhance productivity increase collaboration and communicate effectively Students find and maximize the productive value of existing and new technology to accomplish workplace tasks and solve workplace problems. They are flexible and adaptive in acquiring new technology. They are proficient with ubiquitous technology applications. They understand the inherent risks-personal and organizational-of technology applications, and they take actions to prevent or mitigate these risks.
Content	Algorithm Bug Debugging Sequencing Program Programming	Behavior Sprite Program Algorithm Code	Loop Repeat Command Conditions Conditionals	Function Variable Prompt
Skills	Translate movements into a series of commands. Identify and locate bugs in a program. Predict where a program will fail, modify an existing program to solve errors, and reflect on the debugging	Define sprite as a character or object on the screen that can be moved and changed. Create new sprites and assign them costumes and behaviors. Create an animation using sprites and behaviors.	Identify the benefits of using a loop structure instead of manual repetition. Break down a long sequence of instructions into the largest repeatable sequence. Differentiate between commands that need to be	Use functions to simplify complex programs. Use pre-determined functions to complete commonly repeated tasks. Categorize and generalize code into useful functions.

	process in an age-appropriate way. Order movement commands as sequential steps in a program. Represent an algorithm as a computer program. Develop problem-solving and critical thinking skills by reviewing debugging practices. Create a program to complete an image using sequential steps. Break complex shapes into simple parts.	Create an interactive animation using events. Program solutions to problems that arise when designing a virtual pet, like feeding it or monitoring its happiness. Develop programs that respond to timed events and user input. .	repeated in loops and commands that should be used on their own. Break complex tasks into smaller repeatable sections. Recognize large repeated patterns as made from smaller repeated patterns. Describe when a loop, nested loop, or no loop is needed. Define circumstances when certain parts of a program should run and when they shouldn't. Build programs with the understanding of multiple strategies to implement conditionals. Nest conditionals to analyze multiple value conditions using if, else if, else logic. Pair a loop and conditional statement together. Solve puzzles using a combination of looped sequences and conditionals.	Recognize when a function could help to simplify a program. Use variables to hold words and phrases. Use variables in conjunction with prompts. Assign values to existing variables. Utilize variables in place of repetitive values inside of a program. Use variables to change values inside of a loop. Identify areas where they can use variables to modify quantities during runtime. Examine code to find places where variables can be substituted for specific values.
Assessments and Projects	CreateArt with Code	Create an Animated Character Sprite	Conditional Challenge	Program Design w/Interactive Gallery Walk
Resources	Code.org : Lesson 1 Lesson 2	Code.org : Lesson 5 Lesson 6	Code.org : Lesson 10 Lesson 11	Code.org : Lesson 20 Lesson 21

	Lesson 3 Lesson 4	Lesson 7 Lesson 8 Lesson 9	Lesson 12 Lesson 13 Lesson 14 Lesson 15 Lesson 16 Lesson 17 Lesson 18 Lesson 19	Lesson 22 Lesson 23 Lesson 24 Lesson 25