Title: “HOPportunities”	Targeted Grade: 6-12 Lexile: 610L-800L
Author(s):	Time Expectancy: ~90 minutes Depth of Knowledge (DOK 1, 2, or 3): 2-3
Computer Science Learning Objectives: Student will: ● demonstrate the ability to design and create new sprites with specific attributes, applying knowledge of Scratch interface elements ● develop coding skills by constructing interactive programs for character communication and animation, utilizing Scratch programming blocks effectively.	
Concepts/Keywords: sprites, program, coding, animation, programming blocks, loops	
K-12 CSTA Identifier(s)	**Standard(s)** *and Descriptive Statement(s)*
2-AP-10 2-AP-11 2-AP-12 1A-AP-14 1A-IC-18	**-Use flowcharts and/or pseudocode to address complex problems as algorithms.** (Subconcept: Algorithms; Practice: 4.4, 4.1) **-Create clearly named variables that represent different data types and perform operations on their values.** (Subconcept: Variables; Practice: 5.1, 5.2) **-Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals.** (Subconcept: Control; Practice: 5.1, 5.2) **-Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.** (Subconcept: Program Development; Practice: 6.2) **-Keep login information private, and log off of devices appropriately.** (Subconcepts: Safety Law & Ethics; Practice: 7.3)
K-12 Computer Framework(s)	Practice # and Statement(s)
P1. Fostering an Inclusive Computing Culture P4. Developing and Using Abstractions	1. **Include the unique perspective of others** and reflect on one’s own perspectives when designing and developing computational products. 1. **Extract common features** from a set of interrelated processes or phenomena. 3. **Create modules** and **develop points of interaction** that can apply to multiple situations and reduce complexity.

P5. Creating Computational Artifacts	2. **Create a computational artifact** for practical intent, expression, or to address a societal issue.
P6. Testing and Refining Computational Artifacts	1. **Systematically test** computational artifacts by considering all scenarios and using test cases. 2. **Identify and fix errors** using a systematic process. 3. **Evaluate and refine** a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.
ISTE Standards	Standard(s)/Statement(s)
2. Digital Citizen: Students recognize the rights, responsibilities and opportunities of living, learning and working in an interconnected digital world, and they act and model in ways that are safe, legal and ethical.	2d. Students manage their personal data to maintain digital privacy and security and are aware of data-collection technology used to track their navigation online.
4. Innovative Designer: Students use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions.	4c. Students develop, test and refine prototypes as part of a cyclical design process. 4d. Students exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.
5. Computational Thinker: Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.	5c. Students break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving. 5d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.
Additional Content Standard #(s)	Standard(s)/Statement(s)
NGSS: MS-ETS1-3	Analyze data from tests to determine similarities and differences among several design solutions to identify the best characteristics of each that can be combined into a new solution to better meet the criteria for success. **SEP**:Developing Models **DCI**: ETS1.C: Optimizing the Design Solution
MS-ETS1-4	Develop a model to generate data for iterative testing and modification of a proposed object, tool, or process such that an optimal design can be achieved. **SEP**: Developing and Using Models **DCI**: ETS1.B: Developing Possible Solutions

CCSS-ELA: RI.6.1 Key Ideas and Details RI.6.2 Key Ideas and Details	a Cite textual evidence to support analysis of what the text says explicitly as well as inferences drawn from the text Determine a central idea of a text and how it is conveyed through particular details; provide a summary of the text distinct from personal opinions or judgments.
State (or International) Standard(s): (TBD and identified by location of instructor utilizing lesson).	
References	K-12 CSTA Standards: Computer Science Teachers Association (2017). *CSTA K–12 Computer Science Standards, Revised 2017.* Retrieved from https://csteachers.org/k12standards/. K-12 Computer Science Framework: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf Next Generation Science Standards: https://www.nextgenscience.org/standards/standards Common Core State Standards for ELA: http://www.thecorestandards.org/ELA-Literacy/ ISTE Standards: https://www.iste.org/standards/for-students Bloom's Digital Taxonomy Verbs: https://libguides.bc.edu/c.php?g=628962&p=4506921
Lesson Resources/Folder Access (Link)	

Overview: The objective of the "HOPportunities" lesson is to to advance students' proficiency in Scratch by guiding them through the creation of an animated scenario involving two grasshopper sprites. Students will progressively navigate two programming phases with their own computational challenges and space of digital expression. The final section is another opportunity for students to gain insight into the evolution of computing technology.

As they bring their characters to life and orchestrate a delightful tale of mentorship, students will solidify their grasp of programming principles while expanding their creative and problem-solving skills. The "HOPportunities" lesson offers an exciting opportunity for students to deepen their Scratch proficiency and explore the captivating realm of animated storytelling.

In Part I, "Backyard Drama," students create two sprites and learn about backdrop creation for a new program in Scratch. Students will use what they learned in the previous lesson about sprite manipulation to create an animation of an elder grasshopper teaching a younger grasshopper to jump across flowers. The program will enhance students' understanding of the essential concepts of animation, coordinate manipulation, sequencing and the significance of loops in coding.

In Part II, "Grasshoppers in Petal Pursuit," students will modify their code to create another grasshopper interaction and animation. This introduces students to utilize previously created code in new ways, and encourage them to use their creativity for storytelling. Students will demonstrate their achievement by presenting a fully animated exchange between the grasshopper characters, showcasing their proficiency in sequencing, debugging, and creative coding.

Lastly, in "EXTEND YOUR THINKING: Innovations' Computing Roots", students explore a brief timeline and are asked to think critically about the early computer history in the 1930s and early 40s. Students will describe innovative technologies foundational to current computing systems and abilities.

"HOPportunties" should allow students to grow confidence in their ability to write and manipulate code blocks to create complete programs, They will begin to connect the importance of both structured computational thinking and creativity for writing code. As students deepen their knowledge of computer history, they should recongize the diverse needs technology has been created to meet, and even begin to imagine new opportunities for which it can be used.

Rationale/Background: The purpose of this lesson is to build upon students' foundational knowledge of Scratch programming acquired in previous lessons and guide them towards a more sophisticated level of understanding. As students continue their journey in computational thinking and coding, the "HOPportunities" lesson offers an engaging pathway to explore sprite animation and sequence control. With the establishment of core coding concepts, such as variables, loops, and interactivity, students are poised to advance their coding skills while undertaking an imaginative task.

The students are familiar with basic Scratch functions, sprite creation, and manipulation of code blocks. They have already demonstrated their ability to navigate the Scratch interface and execute simple commands. Leveraging this prior knowledge, the "HOPportunities" lesson challenges students to extend their capabilities by animating a scenario that captures their creativity and curiosity. By catering to their existing knowledge and leveraging their interests, the "HOPportunities" lesson ensures a relevant and meaningful learning experience that supports student growth in both computational thinking and creative expression.

Teacher (Required) Materials/Resources: Hard copy or digital copy of this lesson for the instructor and each student (if teaching this lesson in person), Internet access, a means of meeting virtually (either by Google Meet, Zoom [although Zoom is not preferred due to threat of hacking and inappropriate disruptions], or FaceTime (depending on the size of the intended class).

"HOPportunities Appendix 1"- Internet-Free Supplement: This document is provided in the event that any student does not have access to the Internet in their home environment and the "Extend Your Thinking" portion is assigned to be completed as homework.

Student Materials:
1. An iPad, laptop or desktop computer.
2. Internet access.
3. Either hard copy or digital copy (that will allow writing on [such as Notability]) of "Student Sheet: HOPportunities".

4. Scratch accounts created during a previous lesson.
5. Ability to screenshot or photograph their assembled block sequences for each tutorial category.
    a. Encourage students to look up how to screenshot on their own devices, and assist when necessary.
    b. Example: Use the Google search engine to search “how to screenshot on Windows 10”
6. For “Extend Your Thinking: Innovations’ Computing Roots”: Either hard copy or digital copy of reading material. Students may follow directions to the link provided on their student sheet, or print a hardcopy from “HOPportunities Appendix 1.”

<u>Guided Practice/Instructor Procedures</u>:

A) <u>Introduction and Motivation</u>
    1. Begin by engaging students in a brief discussion about their previous experience with Scratch and sprite creation. Encourage them to share any unique characters they have designed and any exciting interactions they have programmed.
    2. Introduce the concept of storytelling through coding by mentioning that they will create an animated dialogue between grasshopper characters.
    3. Explain that this storytelling will involve coding dialogues, animations, and interactions to craft a narrative.

B) <u>Lesson Body</u>

Part I: Backyard Drama
    1. Instruct students to create a new project and name it appropriately, ensuring they understand how to access the project's workspace.
    2. Lead students in exploring the backdrop options and guide them in selecting the "Flowers" backdrop.
    3. Instruct students to create two grasshoppers as described in their Student Sheets.
    4. Direct students to design code sequences for both grasshoppers as outlined in their Student Sheets.
        a. Encourage creativity in designing the interactions while ensuring the correct sequence of actions.
    5. High-level learners who may finish quickly can be paired with other students. Instruct them to support the peer, answer questions, and debug errors.
    6. Facilitate discussion about the coding process:
        a. Prompt students to reflect on their coding process.
        b. “What challenges did you encounter while coding the mentorship scenario?”
        c. “Did you use loops? Why or why not?”
        d. “Did you use grouping or ungrouping? Why or why not?”
        e. “How did you use coordinates in this scenario?”
    7. Assure that students have pasted a screenshot of their program’s code on their Student Sheet before moving to the next section.

Part II: Grasshoppers in Petal Pursuit
    1. Instructs students that they will now modify their existing code to create the next scenario described on their Student Sheet.

        a. Again, encourage creativity in the design, suggesting students add any elements they find interesting while ensuring the grasshoppers hop on all four flowers and dialogue is included.
    2. Assist students as needed in creating Shared Project Links and ensuring you can see them. Initial Student Sheets.
    3. Higher-level learns can be encouraged to swap their code from their previous program with another student, and use that code to recreate this scenario or a new one they create.
    4. Culminate the coding time by facilitating discussion about creating an animated program, loops, modifying code. Ask questions such as:
        a. "What was different about creating the first and second scenario?"
        b. "Did you change how you used loops, grouping, or coordinates? Why or why not?"
        c. Consider the broader implications of storytelling through coding: "How might this skill be valuable in communicating ideas or narratives in various contexts?"
        d. "Compare and contrast your coding experience in this lesson with your previous experience in sprite creation. How have your skills changed, and what new concepts have you mastered? What concepts are still difficult?"

C) <u>Lesson Closure</u>

1. Summarize the main concepts covered in the lesson: sequences, loops, dialogue, and sprite interactions in creating engaging scenarios.
2. Ask students to reflect on how they have advanced their coding skills by creating today's animations.
    a. "How did you apply our previous lesson today?"
    b. "Are any coding concepts we've learned easier to understand?"
    c. "What concepts are still difficult?"
        i. Ask other students to explain any concepts their peers struggle with, only offering clarification when needed.
3. Extend Your Thinking reading can be completed during the end of class if time allows, or be assigned as homework. Discuss in class if possible.

<u>Student Misconceptions</u>:

- Student may still struggle with debugging, instead immediately asking for help or answers.
    - Walk students through the steps of debugging, such as clearly identifying the problem, review for typos or missing elements, and test small portions at a time.
- Coding is all about individual work, or code is only used for a single program.
    - Explain that reusing one's own code or others is common when coding. Just as loops can be used for repetitive features, it's not always efficient to write brand new code for each program.
- Coding is only about logical or problem-solving.
    - Creativity is often what gives code use! There would be not interactive websites without creative design of code, just like creativity can make the grasshopper animations more interesting.

<u>Reading Selection</u>: Direct students to the provided link or printed "HOPportunities Appendix 1" to read about innovation in computer history. Instruct them complete the table on their Student Sheet about historical events and the event's impact on technological progress.

Assessment:

A) Student assessment (by instructor):

Informal Assessment: The instructor will engage in ongoing formative assessments by actively interacting with students while they work on designing their grasshopper sprites and coding activities within the Scratch platform. The instructor will observe students' level of engagement, their problem-solving strategies, and their interaction with the Scratch interface. The instructor can ask open-ended questions to probe their understanding of key concepts, such as their rationale for choosing specific code blocks or their methods for animating the sprite's movements. This real-time feedback will provide insights into students' grasp of the material and their ability to apply coding techniques effectively. Additionally, the instructor will review the final grasshopper animation to assess students' mastery of Scratch.

Formal Assessment: To formally assess students' comprehension of computational concepts and coding practices, the instructor will guide them in written reflections or class discussions. Students will be asked to elaborate on the role and significance of code blocks in programming, emphasizing how sequences of code contribute to sprite movements and interactions. This assessment will gauge their grasp of foundational coding principles and their ability to articulate these concepts effectively. Additionally, students will share insights from their exploration of the Extend Your Thinking section regarding the historical significance of technological advancements on society. Through these assessments, the instructor can ascertain students' depth of understanding and critical thinking skills in the context of coding and computer history.

B) Instructor Self and Student Evaluation: The instructor is encouraged to complete the following as the lesson is being carried out or reflected after the lesson is completed.

Three Strengths of This Lesson:

1) ____
2) ____
3) ____

Three Elements/Areas for Improvement:

1) ____
2) ____
3) ____

Identification of students (**using initials, not names**) who were not successful in meeting the stated objectives: ____

____

How shortcomings will be addressed prior to starting next session:

____

____

____

Scope and Sequence: Prior Assignment: “iGRASSHOPPER”
Next Lesson: “Jitterbugs”

Look-Ahead: In the next lesson, "Jitterbugs," students will again need access to the Internet and their Scratch accounts. They will continue to explore the use of coding as a communication tool across many disciplines and uses.