Title: “Easy as 1, 2, 3”	Targeted Grade: 6 -12 Lexile: 610L - 1000L
Author(s): TAI-0104	Time Expectancy: 60 - 90 minutes Depth of Knowledge (DOK 1, 2, or 3): 3
Computer Science Learning Objectives: Student will: ● relate a simple, commonly-used task sequence to the functions carried out by and capabilities of computers and the programs that run them. ● discriminate between the tasks that are intuitively human in nature and cannot be completed by computers; evaluate criteria needed for a computer to perform tasks. ● edit and improve sequential tasks.	
Concepts/Keywords: computer, input, output, sequence, algorithm, variable, loop	
K-12 CSTA Identifier(s)	**Standard(s)** *and Descriptive Statement(s)*
1A-AP-08	**-Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.** (Subconcepts: Algorithms; Practice: 4.4)
1A-AP-09	**-Model the way programs store and manipulate data by using numbers or other symbols [graph coordinates] to represent information.** (Subconcepts: Variables; Practice: 4.4)
1A-AP-10	**-Develop programs with sequences and simple loops, to express ideas or address a problem.** (Subconcepts: Control; Practice: 5.2)
1A-AP-11	**-Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.** (Subconcepts: Modularity; Practice: 3.2)
1A-AP-12	**-Develop plans that describe a program’s sequence of events, goals, and expected outcomes.** (Subconcepts:Program Development; Practice: 5.1, 7.2)
1A-AP-14	**-Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.** (Subconcepts: Program Development; Practice: 6.2)
K-12 Computer Framework(s)	Practice # and Statement(s)
P1. Fostering an Inclusive Computing Culture	1. **Include the unique perspective of others** and reflect on one’s own perspectives when designing and developing computational products.

P3. Recognizing and Designing Computational Problems	1. **Identify complex, interdisciplinary, real-world problems** that can be solved computationally. 2. **Decompose complex real-world problems** into manageable subproblems that could integrate existing solutions or procedures.
ISTE Standards	Standard(s)/Statement(s)
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.	5d. Students understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated solutions.
6. Creative Communicator: Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats and digital media appropriate to their goals.	6c. Students communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.
7. Global Collaborator: Students use digital tools to broaden their perspectives and enrich their learning by collaborating with others and working effectively in teams locally and globally.	7c. Students contribute constructively to project teams, assuming various roles and responsibilities to work effectively toward a common goal.
Additional Content Standard #(s)	Standard(s)/Statement(s)
NGSS: MS-ETS1-4	Develop a model to generate data for iterative testing and modification of proposed object, tool, or process such that an optimal design can be achieved. **SEP**: Developing and Using Models **DCI**: ETS1.B: Developing Possible Solutions & ETS1.C: Optimizing the Design Solution
CCSS-ELA: L.7.3a. Knowledge of Language	Choose language that expresses ideas precisely and concisely, recognizing and eliminating wordiness and redundancy.
RI.7.3. Key Ideas and Details	Analyze the interactions between individuals, events, and ideas in a text (e.g., how ideas influence individuals or events, or how individuals influence ideas or events).
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 overall purpose of "Easy as 1, 2, 3" is to allow students to model two things: 1) a computer and how it functions, and 2) a programmer and how (ultimately) they are the ones controlling the tasks performed by the computer. The lesson culminates with a short reading about historical inventions leading up to and including modern computers. It provides an opportunity for students to explore the timeline of computer development and programming throughout the world. It is important to note that computers were developed in times of need (connecting historical engineering and programming events into the daily lives of humans) and creative curiosity.

In Part I of this lesson, students are asked to write the steps needed to perform two familiar tasks: either making a peanut butter and jelly sandwich, or brushing their teeth. (This is a very common introductory lesson used to teach programming). The student task sequence for the first exercise is to be placed in the left side of the table on page 1.

Once a good list of steps is compiled (by the whole group), the instructor follows the steps provided by the class. This activity demonstrates the need for algorithms to be precise, unambiguous instructions that can be followed literally by a computer. It is important that the steps are followed literally. Many times, this activity shows that the instructor (who is acting like a computer) cannot perform the entire task; sometimes the task cannot be performed at all because the instructor (aka computer) cannot access the materials and they are not told how or where to access them, how to hold or use the materials in the way they are intended to be used. Do not be surprised if the whole jar of peanut butter is placed (unopened) into the sandwich!

As the instructor is performing the sequence of tasks, students are asked to make notes that are intended to be shared upon "completion". Once the demonstration is concluded, students are then asked to write the task sequence of the other exercise. This can be done outside of class if time is limited. Students will write the second task sequence in the right-hand column of the table on page 1. It is important that student work for Part I be placed in the table on page 1. This allows the instructor to compare the level of detail and assess student learning from this task.

In Part II of this lesson, students will continue with learning the need for detailed instructions. Students are assigned as either an “INNIE” or an “OUTIE” to create simple drawings of animals in fifteen minutes. The “INNIE” is given a card with the drawing, and they are asked to provide the instructions for their “OUTIE” partners to draw without being shown the card. At the end of the fifteen minutes students switch roles and create a new drawing. If time permits, the instructor can model how to provide instructions in the drawing of a cartoon mouse using the X/Y coordinate plane. An overall goal from this activity is to allow students to become more familiar with each other.

Extending Student Thinking:

- Part I: Emphasize that students revise and improve their step lists based on insight gained from seeing a demonstration, modeling an iterative design process that programmers use. The students are programmers even if they didn’t know it!
- Part II: Provides practice creating and following algorithms orally, which reflects how programmers verbally communicate instructions to coworkers when writing formal code. This oral facet of algorithm development is an important skill.

Rationale/Background: The purpose of this lesson is to model the tasks performed by a computer that are directed by humans in lines of code. Variables, loops, start commands, logarithms, inputs and outputs are also modeled. Students will become acquainted with other students and the overall “tone” of the classroom will be set in this meeting.

Teacher (Required) Materials/Resources: Hard copy of this lesson for the instructor and each student sheet (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). The instructor will need a means of projecting, reviewing and recording the sequence of steps compiled by the whole class for making a peanut butter and jelly sandwich and brushing your teeth.

“Easy As 1-2-3 Appendix 1”- Images for Innie/Outie Activity: This document contains images of a ladybug, a grasshopper, a pig, a giraffe, an elephant, and a lizard.

“Easy as 1-2-3 Appendix 2”- 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. This document can be copied with two pages per side.

Student Materials:

1. An iPad, laptop or desktop computer with drawing application [such as Notability], or paper, crayons and markers.
2. If class is conducted online, Internet access. If the class is conducted online, breakout rooms (in Google Meet) are ideal for Part 2.
3. Either hard copy or digital copy (that will allow writing on [such as Notability]) of “Student Sheet: Easy as 1, 2, 3”
4. For Part II: Either hard or digital copy of one of the images in the “Easy As 1-2-3 Appendix 1” document. Cut or or crop images to provide to students.

5. For “EXTEND YOUR THINKING: Digital Genesis Unveiled.”: 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 “Easy as 1-2-3 Appendix 2.”

Guided Practice/Instructor Procedures:

A) Introduction and Motivation: One of the main goals of this lesson is to provide an environment for the students that is inclusive, nurturing and engaging. Students should become used to working with partners and in collaborative groups that are changing throughout the lesson when possible.

An appropriate “ice-breaker” activity is encouraged prior to starting the lesson. It is ideal if the “ice-breaker” activity allows for students to share their interest in computer science and why they are taking this course. A comprehensive list of IceBreaker topics/ideas can be found here: “Ice-Breaker-Ideas-For-Leaders” found at: (http://blogs.shu.edu/greeklife/wp-content/blogs.dir/367/files/2013/08/Ice-Breaker-Ideas-for-Leaders.pdf).

The overall purpose of the lesson is to demonstrate and instill in students that computers can perform: only the things they are told to do. Very specific instructions, and instructions that have two options (“yes/no”, “this/that” and “either/or”). They should realize that task sequences can become very lengthy, and the more detailed the instructions, the more accurately the computer will perform the task.

Students should also learn that task sequences can always be improved; codes and lines of code must consistently be refined. Students are introduced to “variables” and the need to identify them before running a task sequence, “loops” and the need for shortening code.

B) Lesson Body:
Part 1: The following steps are suggestions for the instructor that have proven to be good teaching strategies:

1. Divide the class in half. Assign half of the class to write the steps of the peanut butter and jelly sandwich, and half of the class to write the steps for brushing your teeth. (**Note: peanut allergies [of students AND instructors] should be considered prior to this lesson**). Two instructors are ideal in this lesson. One instructor will be responsible for the “peanut butter and jelly group”, and the other instructor will be responsible for the “brushing teeth group”. Each instructor will carry out step 2 (below).
2. Review the steps with the whole group and allow all students to contribute. Once the steps are compiled, the instructor will then carry out the agreed-upon steps provided by the students. This demonstration is meant to show flaws in student sequences, and all steps (or logarithms) can always be improved. As students watch the instructor carry out the agreed-upon task sequences, it is important for students to make notes where errors are made and how they can be corrected. (This is debugging!).
3. Ask students to now write the task sequence for the other task. If time permits, this can be done during the class meeting, or as an assignment prior to the next class. Steps written for the other task should be more detailed and likely will be lengthier.

Part 2: Students will be paired (or at most in groups of three). One student will be given a cartoon drawing of an animal, and they will be asked to verbally give drawing instructions to the other student. The student giving the instructions (the “INNIE”) cannot share specific, cluing details to the student interpreting and making the drawing (the “OUTIE”). Students are given fifteen minutes to communicate and make the drawing. If time permits, the students will switch roles and complete the task again. On the second iteration, the instructor should assess improvement.

To take this lesson to another level, the instructor can revisit the task sequence steps that are completed the second time. As a class, identify the parts of the sequence that represent the identification of variables, introduction of any loops, and suggest students add functions into the task sequence.

C) Lesson Closure: To conclude this lesson, ask students to read the timeline about computer history and answer the reading comprehension questions. Students should be shown the “Computer Hope” website and be encouraged to explore computer development events.

Student Misconceptions:

- Computers can "think for themselves" and perform tasks as intended rather than exactly as programmed. Students have to learn that computers execute literal instructions.
- Computers will fix or compensate for imprecise or vague directions on their own. Students have to provide detailed, unambiguous directions for a computer to follow successfully.
- Technology advances automatically, without human input. Exploring computer history and evolution can help students realize computers are a recent invention, arising from a long series of discoveries, needs and innovations.

Reading Selection: Please ask students to type in the following link, https://www.computerhope.com/history/index.htm#timeline, or provide a hard copy of “Easy as 1-2-3 Appendix 2” if Internet is not available and it is expected that this portion is completed at home. Reading comprehension questions for this reading selection are incorporated into the final page(s) of the student sheet.

Assessment:

A) Student assessment (by instructor):

Informal Assessment: Instructor will continuously monitor student progress, engage in conversations about student work, and assist in answering questions about the development of task sequences. It is important that the instructor gather as much information possible about: student interests, student fluency using technology in general, and student comfort in large groups and small groups.

Formal Assessment: Instructor will evaluate mastery of initial command sequencing by reviewing student responses on their student sheet and when compiling the whole-group list of steps in “Making a Peanut Butter and Jelly Sandwich” or “Brushing Your Teeth” tasks.

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: Next Lesson: “Scratch 0.0: Scratching the Surface”

Look-Ahead: In the next lesson, students will need access to the Internet, and either a digital or paper copy (best if in color) of “Scratch 0.0: Scratching the Surface Using Scratch”. Prior to this lesson, determine if another IceBreaker activity will be utilized, and obtain the needed supplies for that particular IceBreaker activity. The instructor will also need white computer paper, pencils and potentially crayons or markers.