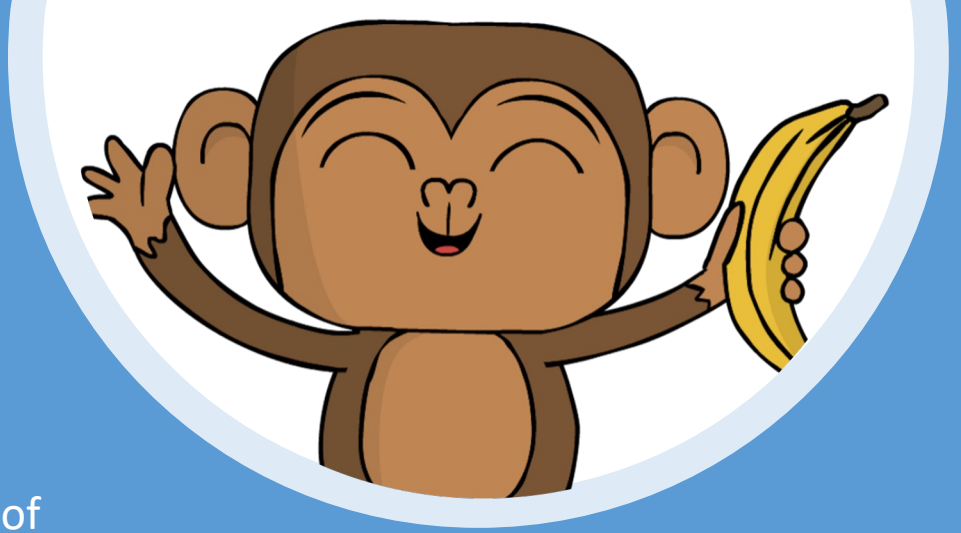
CodeMonkey
LESSON PLANS

Beaver Achiever

Chapter 1

Sequencing and Loops

LESSON PLANS



2345 Yale St., 1st floor
Palo Alto, CA 94306
info@codemonkey.com
www.codemonkey.com

Thank you for choosing to teach your students coding with Beaver Achiever! In this course your students will use block-based coding to help the beaver fix different sections of a dam. In the process they will learn about the programming topics of planning, sequencing, and creating loops which introduce students to fundamental concepts and terms from the world of computers and programming.

This course is recommended for 6-10 year-olds, with basic reading skills.

The course covers a total of 40 challenges spanning over 9 lessons. Each lesson is made up of 3 parts, the introduction, playtime, and debriefing, and is designed to be 45 minutes long.

At the end of the course, you can assign a quiz that includes 5 challenges to test your students' knowledge.

To learn how to set up a class, please read A Beginner's Guide to CodeMonkey. The guide can also be found in the Teacher's Resources Menu on your homepage.

Please email us at info@codemonkey.com for any questions you may have along the way.

Have fun!

The CodeMonkey Team

LESSON PLANS

Table of Contents

Lesson	Topic	Challenges	Slide
Lesson 1 – Introduction to Computers & Coding	Introduction	--	5
Lesson 2 – Meet the Beaver	Sequencing	1-5	13
Lesson 3 – I Have a Plan	Sequencing	6-10	22
Lesson 4 – On Repeat	Loops	11-13	28
Lesson 5 – One More Time	Loops	14-18	37
Lesson 6 – Practice Makes Perfect	Loops	19-23	44
Lesson 7 – And Again…	Loops	24-28	51
Lesson 8 – Hula Loop	Nested Loops	29-34	58
Lesson 9 – Can You Do it Like This?	Increment Loops	35-40	66
Quiz		--	74
Reference Card		--	75

Lesson 1 – Introduction to Computers & Coding

Through this lesson, students will understand what a computer is and how almost everything in our world is digitized or computerized.

Some of your students may be familiar with the terms “coding” and “programming” and feel comfortable working with computers. For others, learning to code may be an intimidating experience. As educators, our goal is to help students learn and explore a variety of subjects, including computer science. We want to create an environment where students can learn from their mistakes and build their foundational knowledge to create something new.

Objectives

In this lesson, students will:

- Understand what a computer is
- Define coding and computer programming
- Learn what is an algorithm

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-CS-02 • 1A-DA-05 • 1A-AP-08 • 1A-AP-11 • 1A-AP-12 • 1A-AP-15	• 1B-CS-01 • 1B-CS-02 • 1B-AP-08 • 1B-AP-13 • 1B-AP-16 • 1B-AP-17

Part 1: 15 Minutes

Introduction

PREPARE IN ADVANCE:

For this activity, please prepare Images of different computers over time (from the first computer to the ones we use today). See next slide for examples.

Discussion	**15 mins.**

Your class' understanding of computers and technology may differ from student to student. The following discussion will help you understand their level.

Lead a discussion with your class about the computers in their lives by asking the following:

- Open with a leading question: *"Imagine that a kid from the Stone Age is coming to class and you need to explain to him or her what a computer is. What would you say?"*
    1. Your students will probably start with answers related to what you use computers for.
    2. Try to lead with questions that will focus them on defining what computers are. Provide them with an example such as is a computer an object or person? Is it mechanical or digital? Is it durable or fragile?
- What types of devices they have at home?
    1. Present different computer images - desktops, tablets, smartphones, etc. (see next slide)
- What are they used for?
    1. What do other family members do?
    2. Examples could be playing games, searching the web, watching videos, working, etc.
- Mention - Computers today go beyond the standard devices we all know. For example, autonomous cars, smart TVs, robots, etc. are all computers.
- Ask them what is connected to their computer. Examples could be a keyboard, mouse, screen, camera, etc.
    1. This is a good point to speak about the difference between software and hardware - explain that we develop the software and without software, computers are just a machines (or hardware) that cannot do anything.

Part 1: 15 Minutes

Introduction cont.

Good to Know in Advance:

- The first computer was invented more than 70 years ago, it was the size of a room
- The evolution of computers started from very large computers to small devices like the tablets we use today. They became smaller and more efficient over time

Useful links:

Computer facts for kids

Next steps in computers evolution

Part 2: 25 Minutes

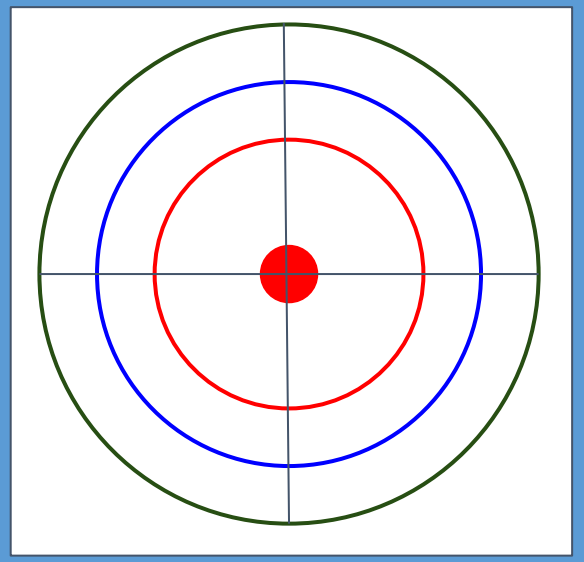
Playtime

Introduction	
Computers maybe smart, but they need our instructions in order to perform operations - from startup and shut down to opening a word document or playing a game. We need to give clear instructions as computers can't think for themselves, they do whatever we tell them to do. Giving instructions to the computer is called computer programming or coding. Computer program is based on an algorithm - a list of unambiguous instructions that you can follow to finish a task. For example, baking a cake, preparing a sandwich, directions to go to school.	
Play	**10 mins.**
Ask a volunteer to write (or describe) the route from the class to the cafeteria, step by step. Discuss with the class - is this accurate enough? can a guest find his way to the cafeteria? Algorithms should be accurate and clear, for humans and for computers.	

Part 2: 25 Minutes

Playtime cont.

Play	**10 mins.**
Game #1 - Instruction-based drawing 1. Print or draw a shape like the bullseye on the right (simplify it as you see fit) 2. Ask for a volunteer 3. Give the volunteer the drawing without presenting to the class 4. Handout paper and pens to all your students 5. Ask the volunteer to guide the class to draw the bullseye, without showing the paper to them and without saying what it is	
Discuss	**5 mins.**
Ask students to present their drawings and have the volunteer show the bullseye. ● How many drawings look like the original? How come? ● What was difficult about the game? Ask both the volunteer and the class.	



Part 3: 5 Minutes
Debriefing

Discussion	**5 mins.**

Computers can only follow clear instructions written in a programming language - this is a computer program.
A computer program is a list of instructions, step by step, that the computer can follow and perform.

Today we tried to imitate a computer and a programmer, we learned:

- Giving clear instructions is not that simple
- Even simple ideas require thoughtful design on how to translate them into an unambiguous instructions

Lesson 2 – Meet the Beaver

This lesson introduces the students to the fascinating world of coding and the CodeMonkey platform and the Beaver Achiever game.

The students will get to know what block programming is all about and will write their first computer program by completing the first five challenges.

Prior to class, make sure all the students have CodeMonkey credentials - username & password.

Objectives

In this lesson, students will:

- Become familiar with the CodeMonkey platform
- Clearly define sequence of instruction to solve a problem
- Complete challenges 1-5

U.S. Standards Addressed

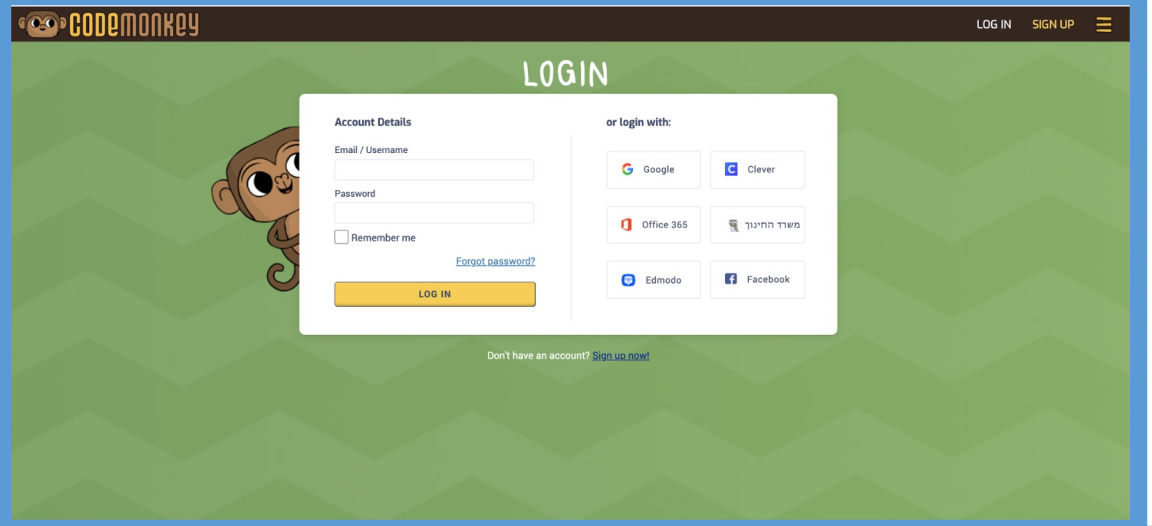
CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-12 • 1A-IC-18	• 1B-AP-13 • 1B-AP-16 • 1B-AP-17

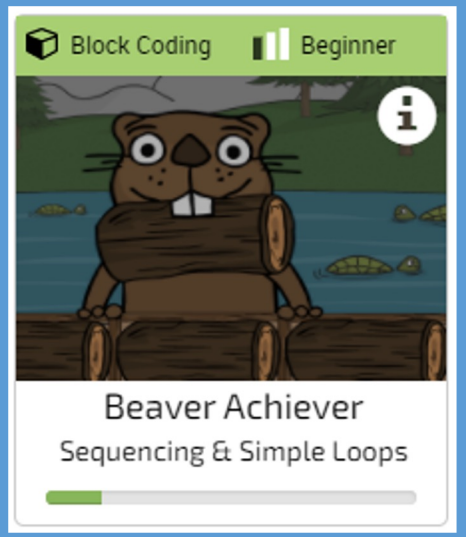
Part 1: 20 Minutes

Introduction

Log-in Information	**5 mins.**

Go to app.codemonkey.com.
Instruct your class on how to log in to their CodeMonkey account.
If your students use usernames and passwords to login, make sure they store their usernames and passwords where they can easily access them in the future. Optional: hand out user log-in cards.
If a student forgets their password, you can reset it by visiting the classroom dashboard, locating the student's username, and clicking on the edit button which will appear if you hover over the username.
After the login process go to the courses area and select the Beaver Achiever Sequencing & Simple Loops course.





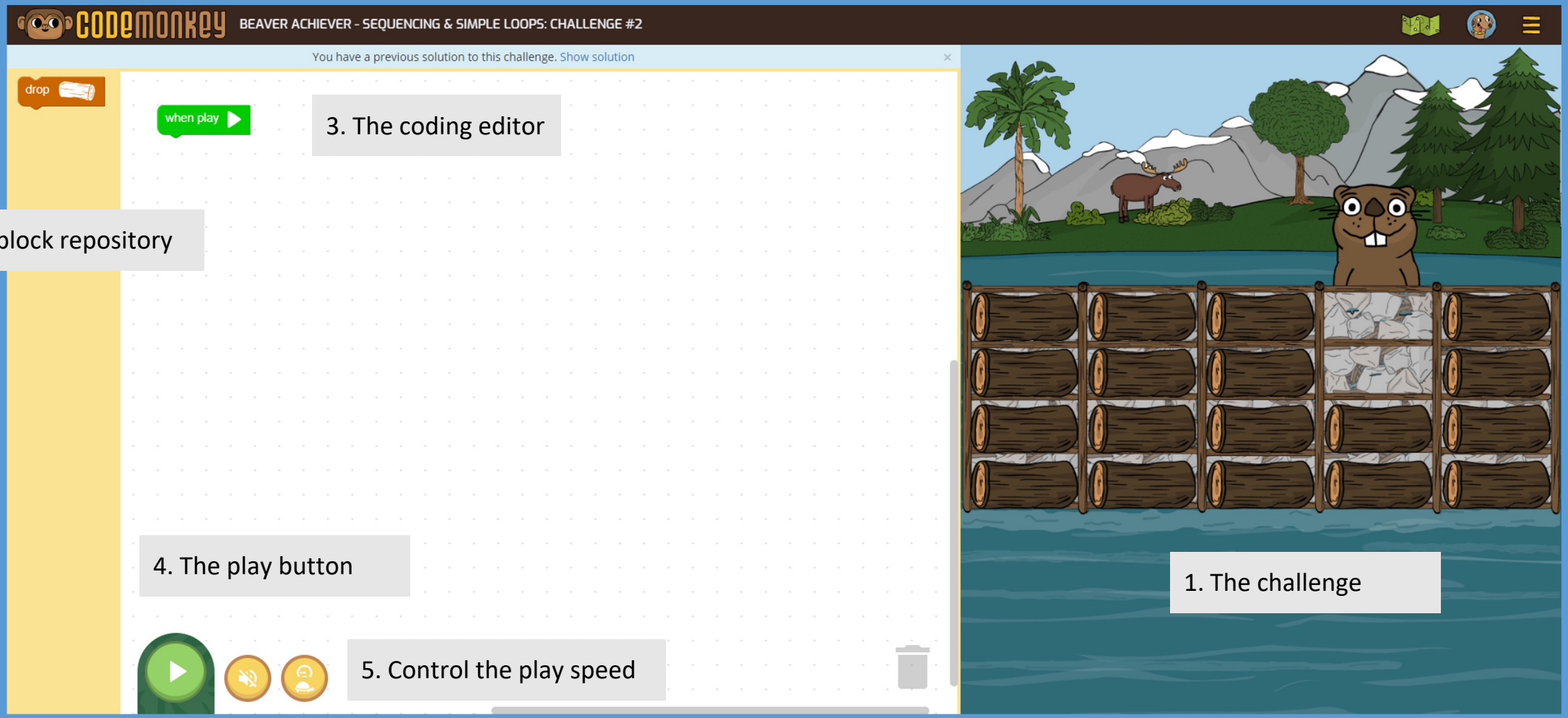
Part 1: 20 Minutes

Introduction cont.

Walk through	**5 mins.**

Walk your students through the basic appearance of Beaver Achiever:

- Open the game on challenge #1.
- Your goal is to complete every challenge by helping the Beaver fix different sections of a dam using block-based coding
- Beaver Achiever is built out of challenges. Each challenge looks like:
  - The game is presented on the right
  - The block repository is on the left
  - The code editor is in the middle
  - The blocks from the repository are dragged to the editor and connected like pieces of puzzle
  - You can drag blocks to the trash on the lower section of the code area
  - On the lower middle section, you will find the play-button, once clicked you will see your code comes to life



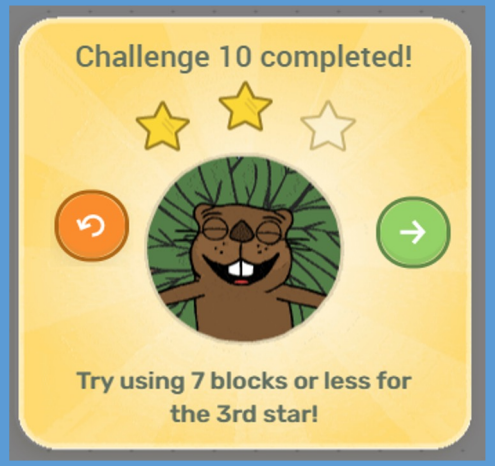
Part 1: 20 Minutes

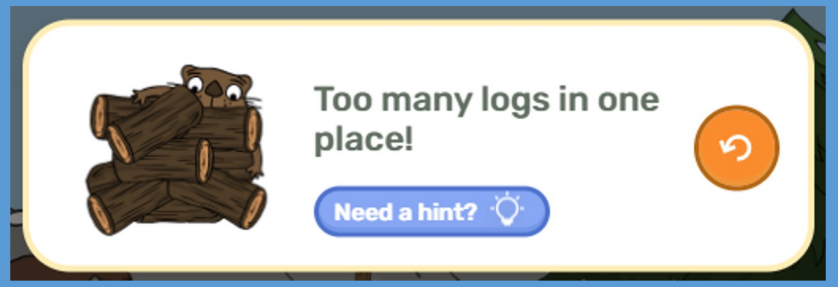
Introduction cont.

Walk through	10 mins.

Ask the students to complete challenges 1 and 2

- After every completed challenge, you will get a star-score rating your solution. 3 stars is the highest score and is rewarded for fixing the dam in the shortest code and using the concept taught in the challenge.
- If you get less than 3 stars, a hint will help you get them all (see below). Pay attention - the 2-star rating is taken from challenge #10 as an example)
- In case of a wrong solution a hint will appear on the top-right side
- You can try to solve a challenge as many times as you want, it will not affect your star score!
- Click on replay to see your solution again.





Part 2: 20 Minutes

Playtime

Guided Playtime	**20 mins.**

After completing the first two challenges, open challenge #3 and emphasize the following:

- New block - move
- Each tile is a single move, each drop is a single log
- Directions: Go over the difference between left and right

All students should complete challenges 3-5 with three stars.

- You could also ask students to try and program erred solutions and 2-star solution - this will improve their understanding of the platform and programming in general
- Use your classroom dashboard to keep track of students' achievements.

Tips & Tricks -

Before the end of the lesson, walk between students to see which level they are at and if they need help.

Learning from errors - Make sure you spend time presenting wrong solutions. This is very important in developing the ability to analyze the cause of the error and how to correct it.

Part 3: 5 Minutes
Debriefing

Discussion	**5 mins.**

Today we learned and wrote our first programs. We learned how to plan, and the importance of the order of instructions

1. What instructions did you learn today?
2. What did you like most about Beaver Achiever?
3. Besides instructions, what else did you learn today?
4. How do you get 3 stars in a Beaver Achiever challenge? Does it matter how many times you try to solve the challenge? (No, it does not!)
5. What do you do when you are stuck?

Lesson 3 – I Have a Plan

In this lesson, students will continue their progress by playing challenges 6-10 which focus on planning by coding advanced sequencing.

Sequencing is when students need to plan the order of instructions carefully.

This lesson introduces students to more complex problems which requires dividing a bigger problem to subproblems in order to solve it easily.

Objectives

In this lesson, students will:

- Review what they learned in the previous lesson
- Identify the different solutions to a single problem
- Complete challenges 6-10

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-11 • 1A-AP-12 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-11 • 1B-AP-13 • 1B-AP-16 • 1B-AP-17

Part 1: 10 Minutes

Introduction

Discussion	10 mins.

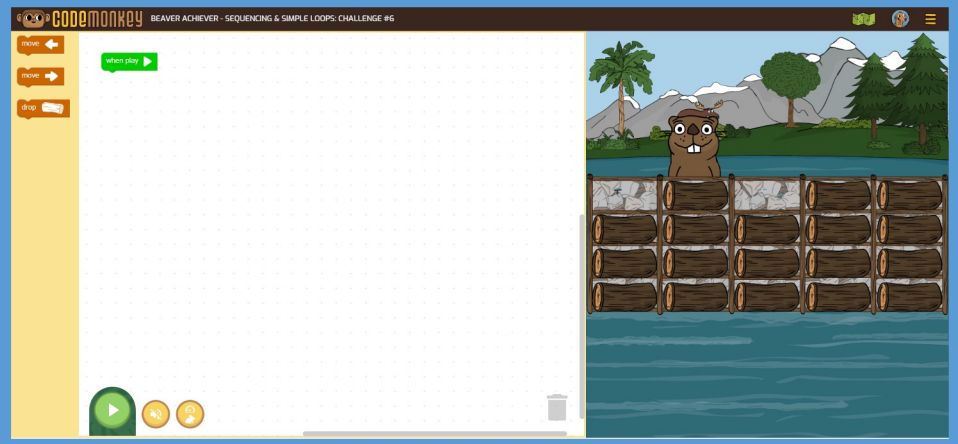
What happens when the missing logs are on your left and on your right?

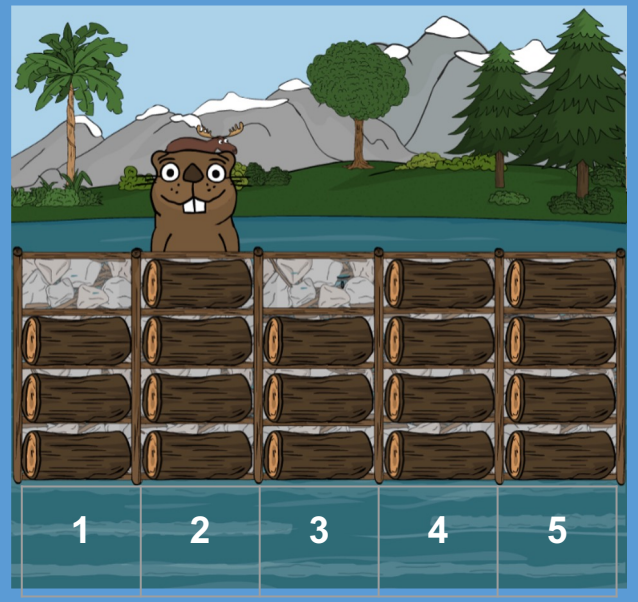
Open challenge #6 - How can you solve this problem?

1. Does it matter which side we move first?
2. If we move left first, how many steps do we have to move right? Where does the counting starts from?

Present the challenge on the board and mark the columns with numbers from left to right -

1. Starting point - log #2
2. Count from column #2 to column #1 - 1 step left
3. Then we need to count from column #1 to column #3 - 2 steps right





Part 2: 30 Minutes
Playtime

Log-in	**5 min.**
Review log-in instructions here.	
Playtime	**20 mins.**
All students should complete challenges 6-10 with at least two stars. Use the classroom dashboard to keep track of students' achievements.	
Analysis	**5 mins.**
Ask the students to present two different 3-star solutions to challenge #10. What is the difference between the two solutions? Can they do the same to challenge #9? Why?	

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**
There are many ways to solve a problem. Each way may differ in quality, effort and outcome. In this chapter, we saw that you can code different paths to fix the dam, and more than one solution can get 3-star rating. Programming requires planning and analyzing the meaning of performing instructions one by one in a sequential order. We learned that counting steps is always a matter of perspective of the starting point and since we need to plan two paths, one to the missing log on the left and then going from the left to the missing logs on the right, or vise versa, the counting will change and are not always related to the starting position of the challenge.	

Lesson 4 – On Repeat

You made it! You and your students now hold the very basic programming skills.

This lesson introduces a new and important programming concept: Loop - doing something over and over again.

Your students will learn the concept of loops in life and in programming. There are several different kinds of loops in programming, like “repeat loops”, “while loops”, and more.

In the scope of this course, we will start first with a simple repeat loop.

Objectives

In this lesson, students will:

- Define *loop* as a programming concept
- Understand why using loops in programming is more efficient
- Complete challenges 11-13

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 25 Minutes

Introduction

Activity	**10 mins.**

This fun activity will lead to a discussion about the concept of repetition and the need for loops:

1. Ask for 2 volunteers - moderator & performer
    a. Brief the moderator in private - “think about an activity the second volunteer will perform - the activity should be simple. For example: raise your hands, jump on one leg, walk around the table
    b. Ask the moderator to decide how many times (between 4-8) the performer will perform the action
    c. Ask the moderator to write on a paper the activity
    d. The moderator will present the paper with the instructions to the performer
    e. Ask the volunteer to perform the action as written on the paper
    f. Then the moderator will present the paper again, and again according to the number of times he decided
2. You can play this game 2-4 rounds with different volunteers

Part 1: 25 Minutes

Introduction cont.

Discussion	**5 mins.**
Ask the students: ● Can you describe the process? ● How would it look if I the moderator would have chosen the number 100? ● Is there an easier way to manage this process? ● What would it be?	
If you need to repeat an action a number of times - it’s called "looping”. When the moderator knows in advance the number of times to perform an action, it is easier if the instruction was "Repeat <action> <X> times”. ● Can you think of other activities that we could loop? Think or your daily routines	

Part 1: 25 Minutes

Introduction cont.

Discussion	**10 mins.**

Let’s go back to our Beaver, what would the code look like if the Beaver needs to place 6 logs?
drop-drop-drop-drop-drop-drop-drop
Is there a shorter way?

Ask your students: “What will a programmer do if they need to write code for placing 100 logs?
Do you think that they will write one line of code for every log-drop?”
Just imagine how LONG this code would be! It would be 100 lines of code!

Ask your students to suggest a shorter way it could be done in the Beaver game.

Luckily, this is possible. In programming we have several mechanisms to manage repeated execution, or as we call it - loops .

Part 2: 15 Minutes

Playtime

Explain	**5 mins.**
Explain to your class that a "**repeat loop**" is a coding mechanism for repeating a sequence of instructions for a specified number of times. There are also other kinds of loops (for loops, until loops) that run until a particular condition is met, but we will learn about those later on.	
Open challenge #11 (guided challenge) - the Beaver needs to place 3 logs in a single column. Up until now we would have solved it with 3 drop blocks. In this challenge we learn how to use the repeat loop, how does it work? ● The programmer controls the number of iterations (how many times the repeat loop will run) ● The blocks nested within the loop block (one, like in challenge #11, or more) are repeated one-by-one on each repetition of the loop ● The repeat loop is part of the program we write, it can be used at every point in our code (beginning, middle, end) ● When there are additional instructions after the loop, the computer moves onto the next instructions once the loop is over	
Why loops are important? Programming is not only about writing the correct statements in the correct order to solve a problem, but also about knowing how to write clear and short code.	

Part 2: 15 Minutes

Playtime cont.

Guided Playtime	**10 mins.**
All students should complete challenges 11-13 with at least two stars. Use your classroom dashboard to keep track of student achievements. How to solve challenges with loop: ● Analyze the sub-process that requires repetition - how many times? which blocks? ● Are there blocks before or after the loop?	
Open challenge #12 - What is the difference from challenge #11? We can see that the repeating block can vary. The students can solve challenges 12-13.	

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**

Today we introduced one of the most fundamental topic of programming - Loops (repeated execution).
While we use repeated activities (loops) in our daily routines implicitly and spontaneously , in programming we must define this structure carefully.

- Ask your students what loops are useful for, why we use them and when?
- Explain the fact that we use loops in cases where we need to repeat an action many times.
- Ask the students to design (in their notebook or on the board) a Beaver challenge that include loop? what is the repeated task within the loop?

In the next lesson we will continue to practice loops and repeated processes. Stay tuned!

Lesson 5 – One More Time

This lesson complements the basic loops we presented in our previous lesson.

Most of the lesson will be focused on playtime and completing challenges 14-18 which include repeated processes that require multiple instructions within the loop.

Objectives

In this lesson, students will:

- Refresh their knowledge of loops that was presented in the previous lesson
- Complete challenges 14-18

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 5 Minutes

Introduction

Explain	**5 mins.**

1. Ask the students what was presented in the previous lesson and why is it so important?
   a. Why loops are important? Programming is not only about writing the correct statements in the correct order to solve a problem, but also about knowing how to write clear and short code.
   b. Remind your class that a “repeat loop” is a coding mechanism for repeating a sequence of instructions for a specified number of times.
   c. What can be repeated when the Beaver builds the dam? Moving right or left, dropping logs.

Part 2: 35 Minutes

Playtime

Playtime	**10 mins.**
All students should complete challenges 14-15 with at least two stars. Use your classroom dashboard to keep track of student achievements.	
Guided Playtime	**10 mins.**
Open challenge #16 - this is the first example where the loop includes 2 block. Ask the students to define the repeating process, use the column numbers as presented on the right: 1. move to column #3 2. drop log 3. move to column #4 4. drop log 5. move to column #5 6. drop log In this way it is much easier to identify the repeating process.	

Part 2: 35 Minutes

Playtime cont.

Playtime	**15 mins.**

All students should complete challenges 16-18 with at least two stars.

Remind the students that solving the problem will be easier if they:

- Analyze the sub-process that requires repetition - which blocks? how many times?
- Are there blocks before or after the loop?

In this chapter, we will see that some challenges can be solved in multiple ways, check with your class to see their creative solutions and make sure they get 3 stars with the most efficient and shortest solution.

Use your classroom dashboard to keep track of your student achievements.

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**

Repeating processes are part of our day-to-day activities, and hence are one of the most important computing structures.

- Ask your students what loops are useful for, why we use them and when?
  - Explain the fact that we use loops in cases where we need to repeat an action many times.
- The main challenge in this chapter is identifying what the repeated section within the loop is and how to build the code accordingly.
- It is important to understand that the entire code within the loop is performed at each loop repetition. The order of the instructions within the loop doesn't change.

Loops are a fundamental part of programming - it is an essential part of any programming language and as becoming a programmer you will master the concept of repetition in many ways.

Lesson 6 – Practice Makes Perfect

In this lesson, your students will continue using simple loops while the challenge complexity increases. The students will deepen their understanding on why it is important to use loops and see different combinations to use loops within their code.

Objectives

In this lesson, students will:

- Understand why code with loops in programming is more code efficient than without loops
- Understand the importance building the code in the right order
- Complete challenges 19-23

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 5 Minutes

Introduction

Explain	**5 mins.**

Start this lesson by asking the students the following:

- Do you remember our previous lesson? We learned a new concept; Do you remember what it was?
- If students do not remember loops, use leading questions such as "what happens when we need to perform the same action many times?" or "In real life, is there a limit to the number of times we need to repeat an action?"
- If students do remember loops, you can strengthen what they already know by going over loops again and explain why they are so important (for example, loops are helpful in executing one or more statements a certain number of times).

Ask your class:

1. Which loop did we use?
2. How can the computer know when to stop iterating the loop?
3. What is the programmer responsible for when writing a loop?
    - number of repetitions and the repeated instructions
4. When do we execute the code that is outside the loop?

Part 2: 35 Minutes

Playtime

Guided Playtime	**5 mins.**
Start by presenting challenge #19 - there are 2 processes that require repetition - moving and dropping. These processes are not connected, these are consecutive processes. The Beaver should move left 3 times before dropping 4 logs → we will solve this challenge with 2 repeat-loops, the first for moving 3 times and the second for dropping 4 times. **Emphasize** - we can have multiple disconnected loops within our code.	
Playtime	**15 mins.**
Ask the students to complete challenges 19-21	

Part 2: 35 Minutes

Playtime cont.

Guided Playtime	**5 mins.**
Open challenge #22 - define the main sub-processes and investigate which process is a loop candidate: 1. move-left 5 times (loop) 2. drop 1-log 3. move-left 4. drop 4-logs (loop)	
Playtime	10 **mins.**
Ask the students to complete challenges 22-23. Emphasize the planning process - define sub-processes, analyze repeating processes and the bottom-up approach.	

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**

This lesson was all about consecutive loops, we know how to write a program with several loops one after the other and with stand alone instructions in between.

Ask the students to describe daily routines that involves consecutive loops, for example, first you brush your teeth, then you comb your hair.
There are many situations in our day today activities that involves a loop followed by another loop, and in programming, like in life, there are many situation when multiple separated loops are required.
We will see and practice more next lesson.

Lesson 7 – And Again…

As challenge complexity increases, your students will practice different loop combinations and understand the importance of using loops for both - code readability and efficiency.

Objectives

In this lesson, students will:

- Understand why using loops in programming is more efficient
- Understand the importance building the code in the right order
- Complete challenges 24-28

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 5 Minutes

Introduction

Explain	**5 mins.**
Start this lesson by going back to Challenge #23. Analyze the problem: ● Divide into sub-problems (fill first column, moving, fill second column) ● What is repeating in each sub-problem?	

Part 2: 25 Minutes

Playtime

Guided Playtime	**25 mins.**
Ask the students to complete challenges 24-28. Emphasize the planning process - define sub-processes, analyze repeating processes and the bottom-up approach.	

Part 3: 15 Minutes

Debriefing

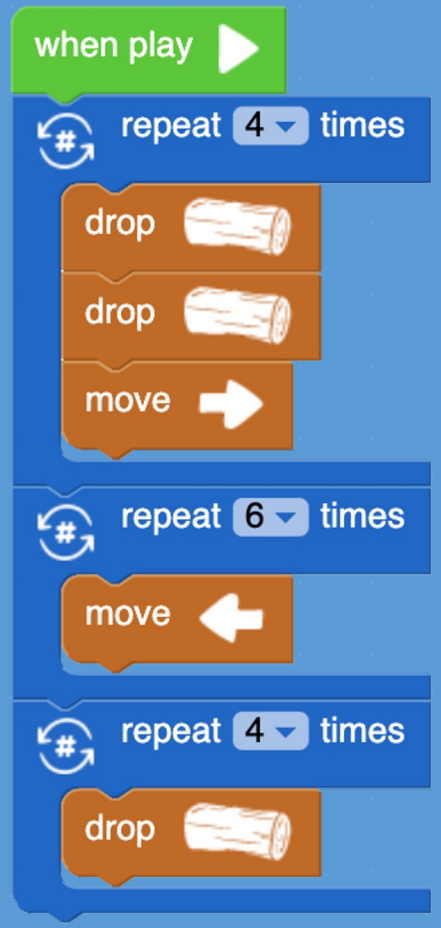
Walk through	10 mins.

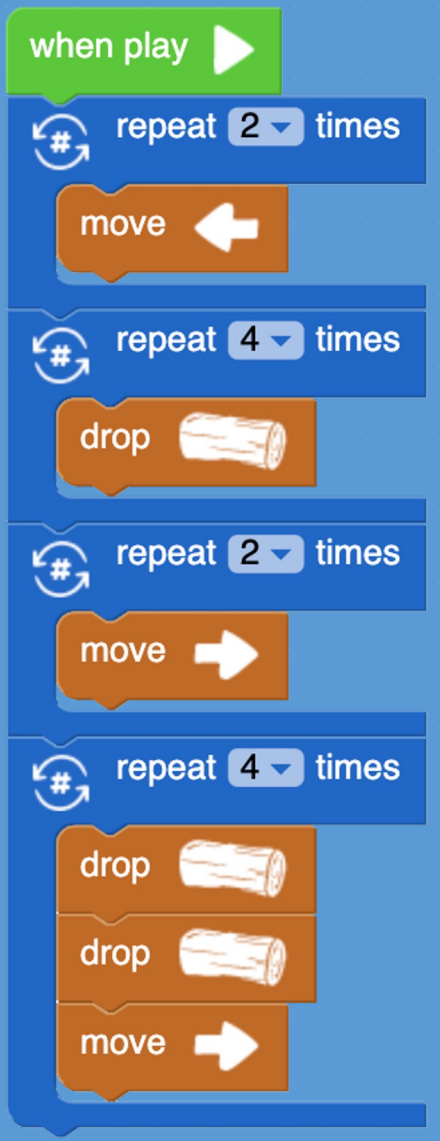
Open challenge #28 and analyze the challenge with your students.

This challenge is a summary of everything we learned so far:

- Planning
- Dividing into sub-processes
- Testing and comparing solutions

We present a few solutions that can solve the challenge, the one on the left is a 3-star solution - with less lines of code, the one on the right is a 2-star solution. Analyze the difference with your class and ask them how would they rate the solutions and if they can find additional solutions.





Part 3: 15 Minutes

Debriefing cont.

Discussion	**5 mins.**
This lesson was all about consecutive loops, we know how to write a program with several loops one after the other and with stand alone instructions in between.	
Now let's see if you are ready for the real challenge :) (and for the next lesson). Open challenge #29 - ask your students to identify the repeating process, what is going on here? Stay tuned for next lesson.	

Lesson 8 – Hula Loop

In this lesson, your students will continue to explore and practice their loop writing skills and learn how loops can be more powerful when nested together.

Objectives

In this lesson, students will:

- Use nested loops
- Continue practicing with loops
- Complete challenges 29-34

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 20 Minutes

Introduction

PREPARE IN ADVANCE:

Dice for the class - each pair of students will need 2 dice

Instructions - The Dice Game	**7 min**

Split the class into groups of 2-4 students and give each group a set of 2 dice.
The game is simple - on each turn one player throws the dice, the first one who gets a double wins the round.
Ask students to document the game by writing how many rounds they needed to get a double number.

Discussion	**3 mins.**

This games involves two nested repeating processes - the turns within a round, and the rounds until reaching the winning state.
Ask the students to describe the game - encourage them to describe the process:

- How many times they played?
- How many rounds they played in the shortest game? the longest game?
- How many times they threw the dice all together?
- What are the steps in a single round? What are the steps in the entire game?
- Can they split the game into small processes?
- Encourage the students to use terms like - repeating, iteration

Part 1: 20 Minutes

Introduction cont.

Introduction	**5 mins.**

Start by presenting challenge #29 - there are 2 processes that require repetition - moving and dropping.
These processes are connected, the Beaver needs to:

- move-left

Then the Beaver needs to drop 4-logs

- drop
- drop
- drop
- drop

This entire process should be repeated 3 times.

Ask your students, “What do you see within this challenge?”

The answer is that the challenge require repeating a process 3 times.
The process itself has a repeated process within - the 4-drops, like the game we played at the beginning of the lesson.
How many drops will be in total? 4 in each column multiply by 3 columns = total of 12 drops.

Part 1: 20 Minutes

Introduction cont.

Introduction	5 mins.

Open challenge #30 - this challenge introduces the nested loops mechanism - Ask the student to follow the guided instructions.
This challenge is the same as challenge #29 - there are 2 processes that require repetition - moving and dropping.
These processes are connected, the Beaver needs to:

- move-left

Then the Beaver needs to **drop 4-logs**

- drop
- drop
- drop
- drop

This entire process should be repeated 3 times

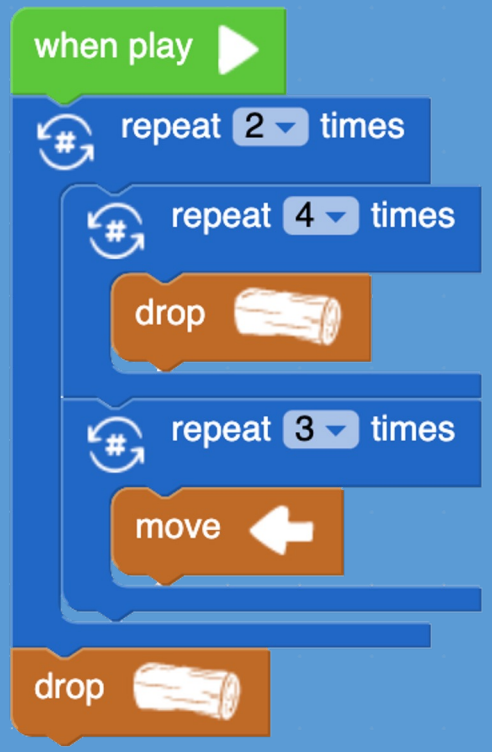
Analyze -

1. the inner loop is responsible for dropping logs - 4 logs on each column.
2. The outer loop is responsible for moving between the columns.

Part 2: 20 Minutes

Playtime

Playtime	**15 mins.**
Ask the students to complete challenges 29-33.	
Guided Playtime	**5 mins.**
Open challenge #34 - this challenge covers nested consecutive loops. Analyze with your students the sub-processes that should be managed with a loop, can we see a repeating pattern? 1. repeat 2 times a. 4-drop b. 3-move left 2. drop 1-log See the solution on the right.	



Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**

A “nested loop” is a loop inside of a loop. When using a nested repeat loop, the inner loop is fully executed on every iteration of the outer loop.

Let’s take school for example, assume that we have 6 grades, and each grade has 3 classes.
If we wish to check how many kids are missing from each class - we need to go grade by grade, and then in each grade go into 3 classes.

- The grades are the outer loop - repeated 6 times
- The classes are the inner loop - repeated 3 times

All in all, the inner loop is executing 6 times 3 classes - 18 classes.

Lesson 9 – Can You Do it Like This?

WOW! You made it, this is the last lesson in this chapter, and there's no doubt your students can code.

This is a good opportunity to teach a new type of loop - the counter loop.

This loop is like the repeat loop - the number of times is predefined, but there is a bonus, we have a counter we can use during the code execution.

We will explain everything shortly.

Objectives

In this lesson, students will:

- Learn how to use counter loop
- The advantage of using a counter loop
- Complete challenges 35-40

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1A-NI-04 • 1A-AP-08 • 1A-AP-10 • 1A-AP-11 • 1A-AP-12 • 1A-AP-14 • 1A-AP-15 • 1A-IC-18	• 1B-AP-08 • 1B-AP-10 • 1B-AP-11 • 1B-AP-13 • 1B-AP-15 • 1B-AP-16 • 1B-AP-17

Part 1: 20 Minutes

Introduction

Game	**7 min**
Arrange the class in a row and instruct the students to perform a specific operation. For example - clap your hands/jump/say “Hi”. Each student will perform the action according to their position in the row - the first will perform once, the second twice and so on. If the class is too big, ask for 5-7 volunteers.	
Discussion	**3 mins.**
This games involves two nested repeating processes? Ask the students to define: ● What is the outer loop? ● What is the inner loop? ● How many times each loop iterates?	

Part 1: 20 Minutes

Introduction cont.

Introduction	**10 mins.**
Sometimes we want to repeat things a certain number of times, but each time we want to perform a task for a different number of times. This is where the counter loop comes in handy. A counter loop uses a variable named **i**. You can think of variable **i** as a container that stores a value. Variable values can be changed and read. The counter loop initiates variable **i** with a starting value, and the value increases every iteration until it reaches the upper bound.	
In the game we just played - each student knew their number in the row and performed the action according to this number.	

Part 2: 20 Minutes

Playtime

Guided Playtime	5 mins.

Ask the students to complete challenge #35 - this is a guided challenge that introduces the counter loop.
There are 3 columns -

- column #1 - is missing one log
- column #2 - is missing two logs
- column #3 - is missing three logs

Additional task - We need to move between columns.

The inner loop will take care of dropping the logs. The number of repetitions will be dynamically set by the number of the column.
The outer loop will take care of increasing the counter value and moving to the next column.

The number of missing logs changes between the columns: starting from 1 and increasing by 1 as we move to the right. This is the exact place where we need to use a counter loop.
Ask your students where should be the move block - within the inner loop? within the outer loop?

Part 2: 20 Minutes

Playtime cont.

Playtime	**15 mins.**
Ask the students to complete challenges 36-40. Encourage the students to solve the challenges with 3-star rating.	

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**

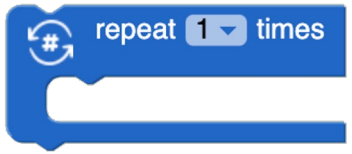
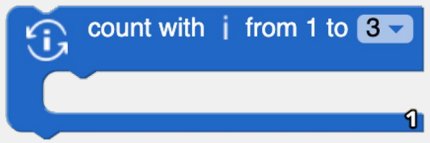
Can you think of other scenarios where you could use a counter loop?
Can you think of other uses for a variable?
Can a counter loop replace the regular repeat loop?

In programming there are many kinds of loops, in the scope of this chapter we learned two. As we progress with the Beaver Achiever courses will learn about other loops, each has its own advantages.

Quiz

Quiz	
Now that your students completed the course, you can assign a quiz for them to take. The quiz includes 5 challenges that will test their knowledge on: • Sequencing • Loops • Nested loops • Loops with counters You can assign quizzes to your class from the Quizzes tab on your teacher dashboard.	

Reference Card

Block	Description
	Move Right - makes the beaver move one column to the right. Causes an error if the beaver is already on the right edge of the dam.
	Move Left - makes the beaver move one column to the left. Causes an error if the beaver is already on the left edge of the dam.
	Drop - makes the beaver drop a log to fill in an empty space in the current column of the dam. Causes an error if the column is already full.
	Repeat Loop - makes the beaver repeat all the blocks placed inside the loop block for the number of times chosen in the dropdown.
	Counter Loop - makes the beaver repeat all the blocks placed inside the loop block the starting from the initial value (1) and up to the upper bound and increasing the value of i by 1 in each iteration.

Great Job!