CodeMonkey
LESSON PLANS

CODING ADVENTURE

FUNDAMENTALS

Lesson Plans
1-16

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

Table of Contents

Table of Contents Cont.

Introduction

Thank you for choosing CODING ADVENTURE to teach your students coding. With fun challenges, cute characters and a unique user experience, Coding Adventure is a great way to introduce your students to the basics of computer science. The following course does not require any prior coding courses or experience and with the following 16 lesson plans, you can jump right into teaching your students real code at any time. The following lesson plans will cover all the challenges featured in CODING ADVENTURE FUNDAMENTALS.

Each lesson is made up of 3 parts, the introduction, playtime, and debriefing, and is designed to be 45 minutes long. Each section is further divided into the amount of time it takes. At the end of this document you will find a Reference Card that will summarize each coding concept.

For information regarding setting up a class, please read A Beginner's Guide to CodeMonkey. The guide can be found here or in the Teacher's Resources Menu on your homepage. Please feel free to email us at info@codemonkey.com for any questions you may have along the way.

Good Luck!!
The CodeMonkey Team

Lesson 1 – Let's Get Started

This lesson introduces students to the fascinating world of computers and to the CodeMonkey platform. 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.

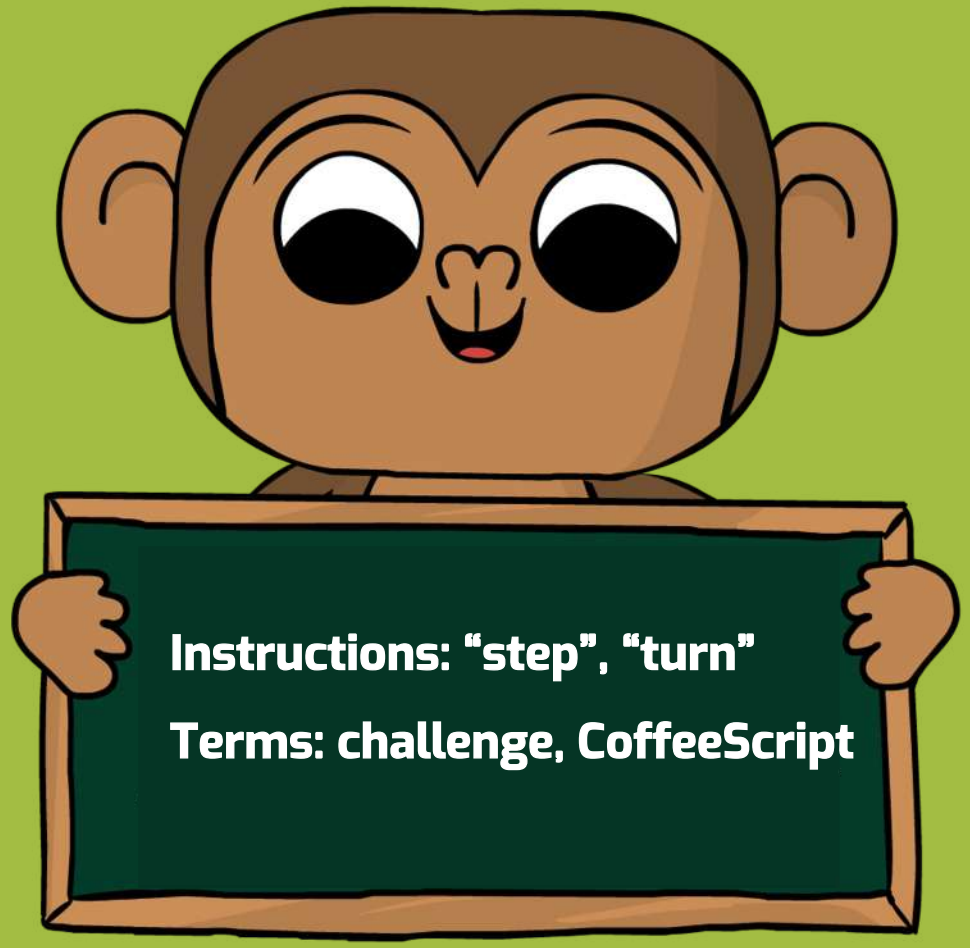
Good luck!

Objectives

In this lesson, students will:

- Define *coding* and *computer programming*
- Become familiar with the CodeMonkey platform
- Complete challenges 0-5

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 15 Minutes

Introduction

Discussion	**2 mins.**
1. How many of you have ever used a computer? 2. Have you ever created something on a computer, like a presentation, a drawing, or maybe even a game? 3. Let two or three students tell the class what they created.	
Explanation	**1 min.**
In order for all of our favorite applications and games to work on a computer, we have to give instructions to the computer. 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.	
Activity	**3 mins.**
Play a short game with your students to illustrate instructions. Place an object somewhere visible in the classroom. Ask your students to give you instructions to guide you from where you are standing to the object. What instructions did the students use? Step, turn right, turn left?	

Part 1: 15 Minutes

Introduction Cont.

Discussion	**2 mins.**
Do computers speak the same language as people? Computers have their own languages; they cannot understand human language as we understand it. Java, JavaScript, and Python are just a few of the languages computers speak. Each language is different, but they all have something in common: they require a certain way of thinking, clear instructions and structure. Basically, learning a coding language is just like learning a new language.	
Video	**7 mins.**
Today you will start learning basic coding principles through a game called Coding Adventure. The language we will learn is called CoffeeScript. Show the CodeMonkey Trailer to your class. Show your class the video of Getting Started with Coding Adventure – A Guide for Students.	

Part 2: 25 Minutes

Playtime

Log-in Information	**3 mins.**
Go to www.codemonkey.com. Instruct your class on how to log in to their CodeMonkey accounts. 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.	

Part 2: 25 Minutes

Playtime Cont.

Walk-through (1)	**4 min.**

Walk your students through the basic appearance of Coding Adventure:

- Click on the "play now" button on the homepage
- Watch the short introductory clip
- Read the instructions out loud
- Coding Adventure is built out of levels called challenges. This is what a challenge looks like.
- The editor on the right is where you will write your code. You can also use the buttons at the bottom of your screen for easy access.
- On the left is the stage. This is where you will see your code come to life. Your goal is to complete every challenge by helping the monkey catch the banana.
- The monkey on the lower left corner is called Gordo; he will give you instructions and sometimes even hints if you are stuck. At any time, you can click on Gordo to see the instructions again.

Part 2: 25 Minutes

Playtime Cont.

Walk-through (2)	**3 mins.**

- In every challenge, you will execute the code by clicking on the "run" button to see what the starting code will do.
- The code on the right says "step 15", so when we will click on "run" the monkey will step 15 steps forward.
- Click on run.
- We completed the first challenge. After every completed challenge, you will get a star-score rating your solution. 3 stars is the highest score and is rewarded for catching all the bananas, implementing newly-learned topics and writing short code. If you get less than 3 stars, a hint will help you get them all. 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.
- Edit the solution to change it from

  step 15

  to

  step 5

  step 10
- Click "run" again to execute your solution. Show the students that this solution only got 2 stars and draw their attention to the hint that tells them how to get the third star.
- Click replay again, fix the solution to get 3 stars and execute it again by clicking "run".

Part 2: 25 Minutes

Playtime Cont.

Walk-through (3)	**3 mins.**

- Let us move onto the next challenge, click on “next challenge”.
- Read the instructions out loud
- The code on the right says “step 10”, let us click on run and see what happens.
- The monkey did not walk far enough, and the hint told us to try “step 15”, so let us change the number 10 to 15, and click “run” again.
- It is always a good idea to use the pre-existing code, since it helps guide us to the correct code. Before we try to change the code, click “run” to see what happens, read the hint, and then try to solve the challenge. Since your score does not get affected by running a wrong solution, it is always a good idea to click on “run” before you begin a challenge. After all, the code is there for a reason, so there is no reason to delete it.
- Another good strategy for when you are stuck is to start again from the beginning, in cases like these you can reset your code by clicking on the reset button.
- Click the replay button and then click the reset button to show your students how to reset the code to what it was in the beginning.
- Solve again by editing the code and click “run” to execute the solution.
- Show your students how to go back to challenge 0 by clicking on the map in top right hand corner and clicking on challenge 0. Note that unlike you as a teacher, your students will not be able to skip forward beyond challenges they have not yet solved.

Part 2: 25 Minutes

Playtime Cont.

Playtime	**10 mins.**
All students should complete challenges 0-5 with at least two stars. (Students from the age of 12 and up should get three stars.) Use your classroom dashboard to keep track of students' achievements. If students are having trouble confusing right and left, draw their attention to the watch on the monkey's left wrist. Tell them that turning in the direction of the watch is left.	
Review	**2 mins.**
Open challenge 2 and show the ruler animation. Follow the instructions to measure the distance between the monkey and the banana, and then use that distance to fix the code. Make sure your students understand how to use the ruler.	

Part 3: 5 Minutes

Debriefing

Discussion	**3 mins.**

1. What instructions did you learn today?
2. What did you like most about Coding Adventure?
3. Besides instructions, what else did you learn today?
4. How do you get 3 stars in a Coding Adventure 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?
6. In a CodeMonkey challenge, how do you display the instructions again?
7. In a CodeMonkey challenge, how do you reset the code to what it was in the beginning?

Part 3: 5 Minutes

Debriefing Cont.

Review	**1 mins.**
Open challenge 6 and solve it with your class. They will solve it by themselves in the next lesson.	
Assignment	**1 mins.**
Due next lesson, create a map with your route to school by writing the directions as computer instructions, just like you learned today. You can also route the way from your room to other places in your house, or even from your homeroom at school to the playground. Be sure to use the basic instructions used in CodeMonkey. Show an example of such a sequence of instructions on the whiteboard.	

Lesson 2 – Turn Around

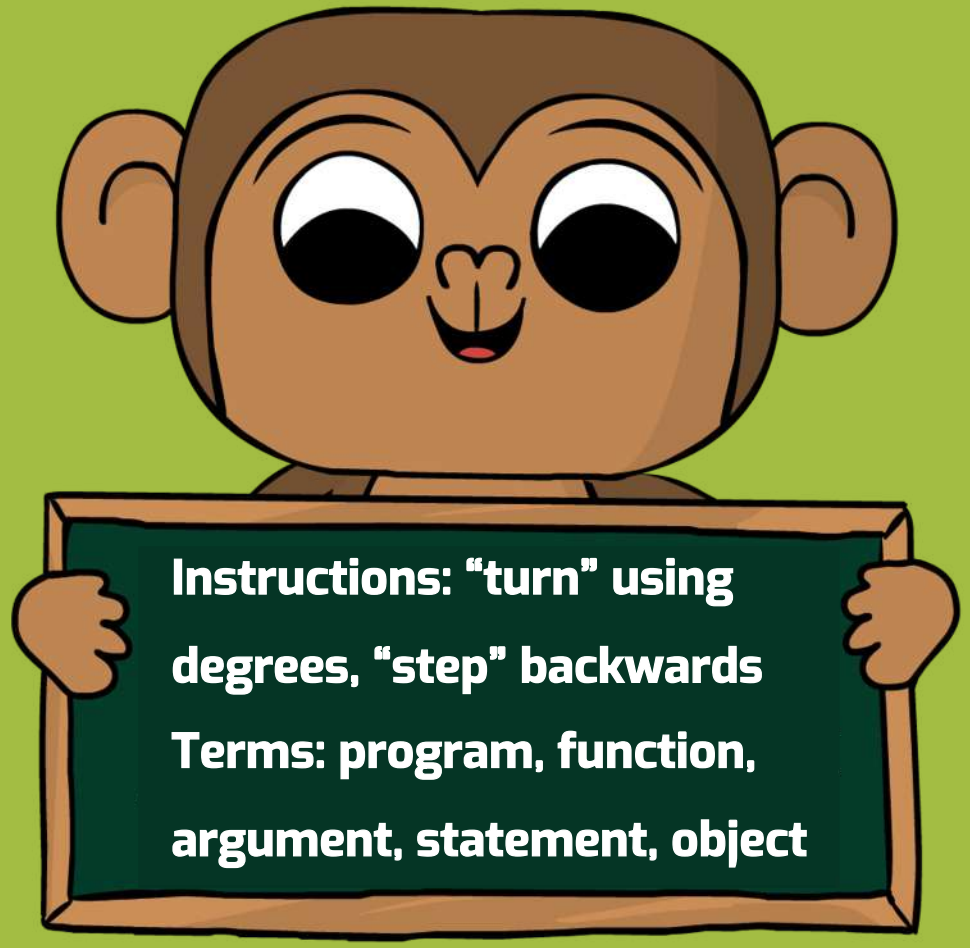
In this lesson, students will continue to explore the CodeMonkey platform by completing five more challenges. Prior to class, use the classroom dashboard to make sure all of your students have completed the first five challenges with three stars. It is important that all students are on the same level and no one is behind.

Objectives

In this lesson, students will:

- Review what they learned in the previous lesson
- Identify the different ways to use “turn” instructions
- Complete challenges 6-10

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 25 Minutes

Introduction

Review	**5 mins.**
Collect the homework. Start with a brief class discussion on the previous lesson. Ask them the following: • What is coding? • What instructions have we used so far? (step, turn) • What is a programming language and which one do we use in CodeMonkey? (CoffeeScript)	
Activity	**5 mins.**
Ask for three volunteers, giving each of them a role: one is the “Programmer,” one is the “Computer,” and the third is the “Character.” Now ask the “Programmer” to instruct the “Computer” to lead the “Character” to an object you placed in the classroom. Make sure the students use instructions properly (“step” with a number, “turn” left/right). As they go, write the instructions on the board to remind the other students of what they have learned. Repeat this activity with another group of three volunteers.	
Discussion	**3 mins.**
Ask the students, “Why do you need both a computer and a character? Why can’t one person be both?” If we compare programming to the human body, then the programmer is the brain that sends instructions to the different parts of the body. The computer is responsible for making sure that the different parts of the body (“characters”) execute the instructions exactly as instructed.	

Part 1: 25 Minutes

Introduction Cont.

Explanation (1)	**2 mins.**
Introduce your students to the term statement: an element that expresses some action to be carried out. A computer program is a set of instructions that are simple tasks provided to the computer. These instructions are called statements. The instructions the “Programmer” gave earlier to the “Computer” are statements. Statements can be anything from a simple line of code to a complex set of conditions and formulas.	
Explanation (2)	**3 mins.**
This lesson is about turning and walking backwards. There are three ways to make a character turn; the first is to use “turn right/left” like we learned in the first lesson. In this lesson, we are introducing another way to turn. Instead of turning right/left, we can turn by degrees. If your students have basic knowledge of degrees, such as a 360 degree turn or a 90 degree turn, then make a quick review of that knowledge. Otherwise, provide a short introduction to degrees. Optional: use a protractor.	

Part 1: 25 Minutes

Introduction Cont.

Explanation (3)	**3 mins.**
Objects are everything in the scene we can interact with, like the bush, bridge, banana, and turtle. Each object has a set of actions it can do, like “step”, “turn”, or “turnTo” (we’ll learn about turnTo in the next lesson) for the monkey. These actions are called functions, and the input we add to them is called an argument. For example in “turn 10”, the argument is 10.	
Discussion	**2 mins.**
• Ask the class to give you an example for a statement and write it on the whiteboard (possible results: step 10, step 15, turn right, turn left) • Ask what is the function in this statement (step or turn) • Ask what is the argument (10, 15, right or left)	

Part 1: 25 Minutes

Introduction Cont.

Explanation (4)	**2 mins.**
Understanding the concept of walking backwards is pretty easy. If we want to go forward 15 steps, we type “step 15”, and if we want to go backwards, we type “step -15”. -15 will be read by the computer in this context just like “15 steps backwards”. If your students are older (6th grade and above), this is a good opportunity to talk about negative numbers on the number line.	

Part 2: 15 Minutes

Playtime

Log-in	**1 min.**
Review log-in instructions here.	
Playtime	**2 mins.**
All students should complete challenges 6-10 with at least two stars. (Students from the age of 12 and up should get three stars.) Use the classroom dashboard to keep track of students' achievements. Keep in mind that students might find turning with degrees difficult. You may need to provide extra help in challenges 7 and 8. Use the walk-through below.	
Walk-through	**2 mins.**
Open challenge 7 and show the animation about angles. Use the ruler to measure the distance between the monkey and banana. Show how the ruler is also a protractor - it shows the number 45 which is the angle the monkey has to turn in order to face the banana. Show that this is the same number in the code. Make sure your students understand how to use the ruler as a protractor.	

Part 2: 15 Minutes

Playtime Cont.

Explanation	**2 mins.**
Open the challenge map and show your students the skill mode tab. Explain that in skill mode students can play through more challenges to perfect their coding skills. These extra challenges are great practice and they only unlock after students complete certain challenges. Hover over a locked challenge to show the unlocking tip. The first skill challenges will open for your students after they complete challenge 6. Let students know that if they finish early, they can go to skill mode and complete unlocked challenges.	
Playtime	**8 min.**
Students should continue working on levels 6-10. In challenges 8, yours students can use either “turn left” or “turn 90” to get three stars. Some of your students will probably use “turn left”. Make sure to emphasize that they can also use “turn 90” for the same result.	

Part 2: 15 Minutes

Playtime Cont.

Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 6-10, all of world 1 (First Steps) skill challenges are unlocked (1-1 – 1-11).	

Part 3: 5 Minutes

Debriefing

Review (1)	**2 mins.**
Check your students' understanding of turning with degrees. Ask your class to stand up and instruct them to "turn 90", "turn 120", and "turn 360". Repeat the explanation of turning by degrees – the code "turn" followed by a number of degrees turns the monkey by that number. For example, turn 90 turns the monkey the same as turn left.	

Part 3: 5 Minutes

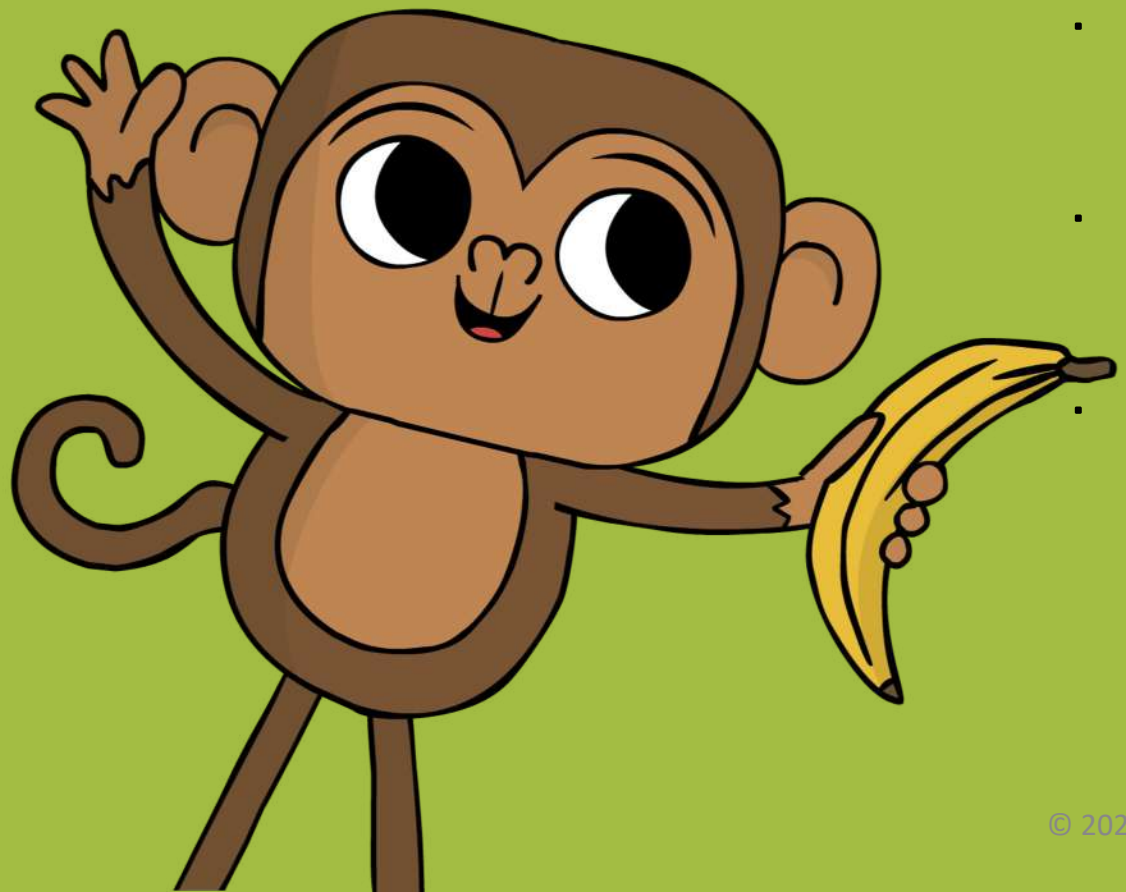
Debriefing Cont.

Review (2)	**2 mins.**
Check your students' understanding of walking backwards. Stand with your back to the door and ask, "If I were the monkey what would be the right instruction to get me to the door?". Emphasize that it should be one instruction and not involve turning. Make sure their answer includes "step negative X". Repeat the explanation of stepping backwards. To step backwards a number of steps, add the negative sign (-) before the number. For example: step -10. The computer reads -10 in this context just like "10 steps backwards".	
Assignment	**1 min.**
Due next lesson, ask your students to include degrees in the navigation instructions from their homes to the school.	

Lesson 3 – I Have a Plan!

This lesson revolves around planning. Everything we do in the physical world has to be planned, even if we sometimes do things automatically. We can cross the road without checking if it's clear, but that may result in a very dangerous outcome. Computers are the same; if we want to create a game or a program, we have to plan ahead and organize our instructions in the correct order.

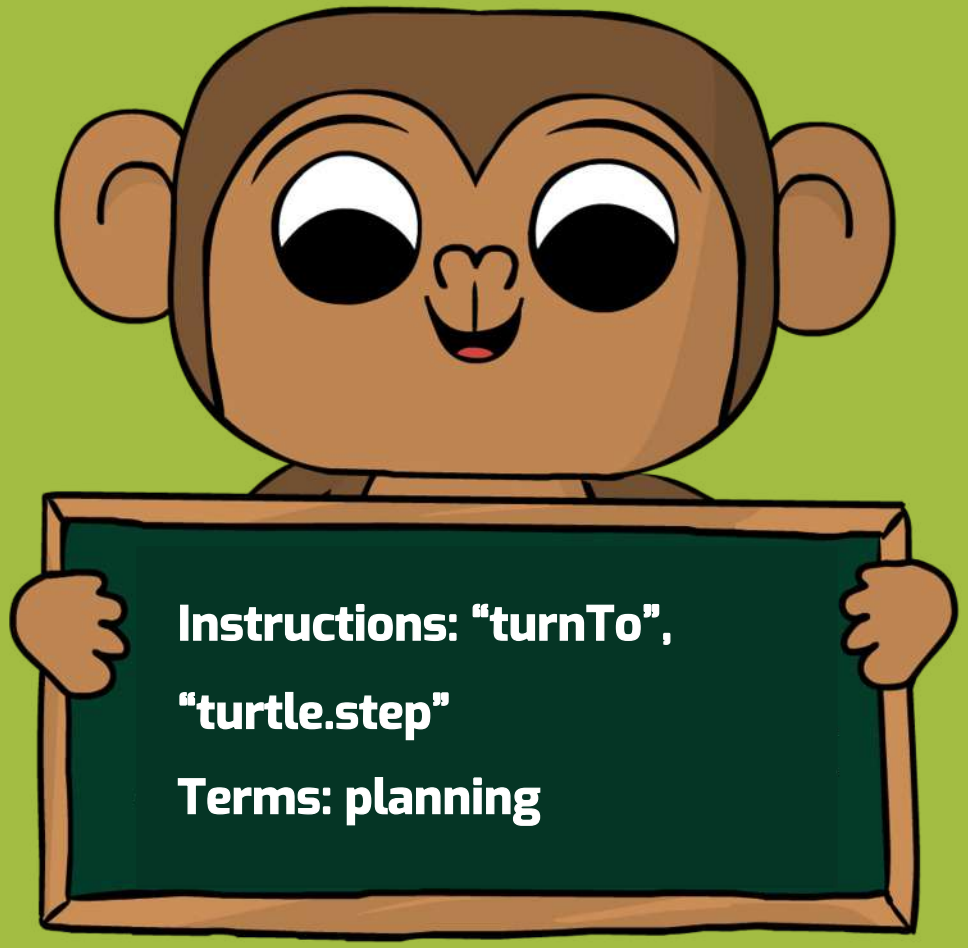
Objectives



In this lesson, students will:

- Review what they learned in the previous lesson
- Discuss the concept of planning and its importance in coding
- Complete challenges 11-15

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 20 Minutes

Introduction

Review	**5 mins.**
Collect last lesson's homework. Recall that we have learned two ways to turn, and ask your students to say some examples of both of them. The first way is turn right and turn left and the second way is turn 45, turn -120, etc. Introduce the third way to turn: by using "turnTo". When using "turnTo" the computer identifies that there is another object present, besides our beloved monkey, and by calling its name, it knows which way to turn. Review the first two ways to turn by asking for two volunteers and instruct each of them to explain and demonstrate one of the ways we learned in the previous lesson: 1. Direction (e.g. turn right) 2. Degree (e.g. turn 180) Write their answers with the examples on the board so the rest of the class remember as well.	
Activity (1)	**5 mins.**
To check your students' understanding of "turnTo", play a short game in the spirit of Simon Says. Give instructions to your students to "turnTo" a specific place or a specific student. They should only turn when you say "turnTo", and not when you say "turn".	

Part 1: 20 Minutes

Introduction Cont.

Activity (2)	**5 mins.**

In this lesson, your students will learn about planning.
Ask your students, “What do you do in the morning to get ready for school?”
Write their answers scattered on the board (not in a list).
Next, create a list out of the actions on the board, and put the tasks out of order, for example:

1. Get dressed
2. Take a shower
3. Wake up
4. Brush my teeth
5. Eat breakfast, etc.

Part 1: 20 Minutes

Introduction Cont.

Discussion	**5 mins.**

Ask your students, “Is this the order of actions you will take to get ready in the morning?” When they say no, ask them why not.

The point of this activity is to show the students the importance of planning. We plan our day and the order in which we do things; sometimes we do this without thinking and sometimes we plan every step.

Explain to your students that when we write code, we have to consider that computers read the code from TOP to BOTTOM, and we have to think ahead about the order of instructions. When we have just one object, this is not a big problem (in our case, the monkey is the object). But what happens when we want to control another object? How do we know who should be instructed to go first?

In this lesson’s challenges, your students will meet our trusty turtle and will have to use his help to get more bananas. In order to do so, they will have to think ahead and plan how to write the code.

Part 2: 20 Minutes

Playtime

Log-in	**1 min.**
For log-in instructions, go here.	
Playtime	**2 mins.**
All students should complete challenges 11-15 with at least two stars. Use the teacher dashboard to keep track of students' achievements. After 2 minutes, use the following walk-through:	

Part 2: 20 Minutes

Playtime Cont.

Walk-through	**6 mins.**

Open challenge #12 and show the animation. It explains how to use objects on the screen. After the animation, walk your students through the following steps:

- Hover over the bridge, show that the word “bridge” appears on the screen
- “bridge” is the name of that object.
- Highlight the word banana in the editor
- Click on the bridge and show how the word banana is replaced by bridge
- Move the cursor by clicking on row 3 after the word turnTo
- Click the banana and show how the word banana is entered into the code
- Move the cursor to line 4 and write “step 10”
- Run the solution
- Click replay to go back to your solution

Part 2: 20 Minutes

Playtime Cont.

Walk-through Cont.	**6 mins.**

- Now, delete all the code to start from scratch.
- You will demonstrate how to use even more clicking instead of typing.
- Hover over the block “step” at the bottom of the editor, show how a description shows up
- Show the descriptions that show up when hovering over every block
- By clicking step, turnTo, bridge, and banana, reach the following solution:

```
turnTo bridge
step 10
turnTo banana
step 10
```

- Be sure to only used the keyboard for typing the number and jumping to the next line.
- Be sure your students understand how to use clicking and hovering for object on the stage (banana, bridge) and for blocks at bottom (turnTo, step).

Part 2: 20 Minutes

Playtime Cont.

Playtime	**11 mins.**
Have your students continue working on challenges 11-15.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 11-15, skill challenges 2-1 – 2-5 are unlocked.	

Part 3: 5 Minutes

Debriefing

Discussion	**4 mins.**
Open challenge 14 and ask your students, “How did you plan what to write in your code?” Be sure to lead them to the correct answer, explaining the correct train of thought needed when planning the code. We should first think about which steps should be taken to achieve our goal (in this case, to get the banana) and then break the steps into separate statements while deciding what should come first (should the turtle or monkey step first?). If we tell the monkey to move before the turtle is in the right place, the monkey will fall in the water and monkeys do not like water.	
Review	**1 min.**
Use this opportunity to remind your students that a program is a set of instructions, or simple tasks provided to a computer. These instructions are called statements. Statements can be anything from a single line of code to a complex mathematical equation.	

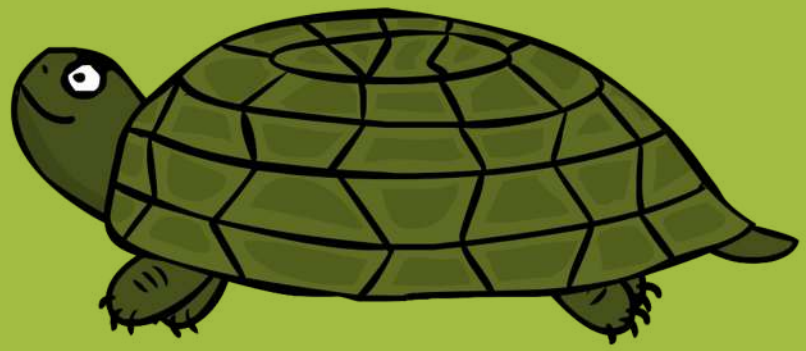
Lesson 4 – Turtle Lake

In the previous 3 lessons your students have learned how to move around using code. They have actually mastered the foundation to programming, as they are now able to write a block of code that will carry out the instructions they intend to give the computer. We will take the current lesson to practice and reinforce this knowledge, and to deepen their understanding of what is actually going on.

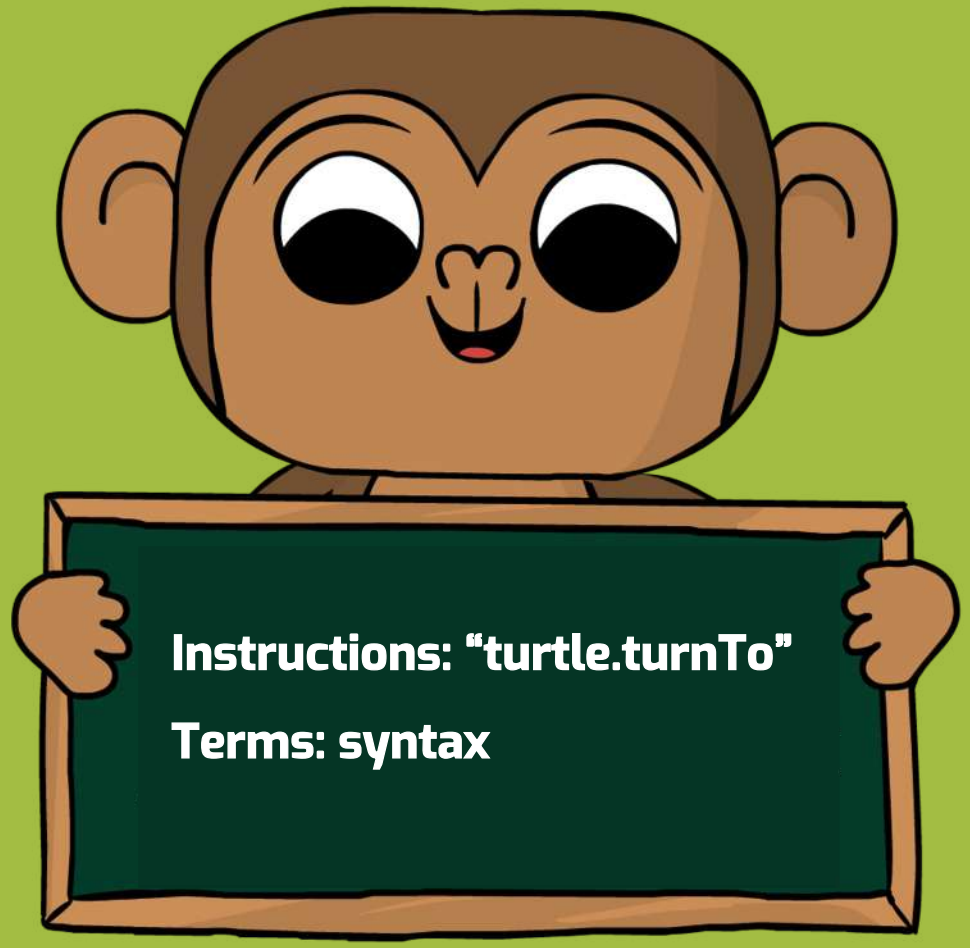
Objectives

In this lesson, students will:

- Practice using functions with different objects (monkey, turtle)
- Complete challenges 16-20



Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 10 Minutes

Introduction

Explanation	**5 mins.**

Recall with your students that in Coding Adventure we are writing code in a programming language called CoffeeScript.
As they experienced in the previous lessons, the code has to be written in a particular way in order for the computer to do what we are trying to achieve. Explain that this is because a programming language, just like any language, has its own rules on how things can be said or written.

In programming this is called the syntax of the language. There may be more than one correct way to say or write a certain statement in CoffeeScript, just like in English or any language. An important difference between programming languages and other languages is the following:

In a spoken language, sometimes we can say something incorrectly and still be understood. However, in a programming language even the slightest mistake will definitely cause our code to fail. So, we always have to pay attention to syntax and be very accurate. Note that we can still write code correctly syntax wise, but the challenge will still fail because our code was incorrect.

For example, if we forget a dot or a space in turtle.step 10 we will get turtle step 10 or turtle.step10, and the code will not work correctly.

Part 1: 10 Minutes

Introduction Cont.

Walk-through	**5 mins.**

Open challenge 15 and click “reset” to reset the code to what it was initially (blank). Use typing only, no clicking, and enter the following code:

```
turtle step 10
```

Click “run” to execute the code. Read out loud the error message that appears. Explain that the dot is important. In this example, the computer was able to guess what we meant, but this is not always the case.
Edit the code to the following:

```
turtle.step 10
step15
```

Execute it and read the error message with the students.
Repeat the same with the following modification to the 2nd line (capital S):

```
turtle.step 10
step 15
```

Part 1: 10 Minutes

Introduction Cont.

Walk-through Cont.	**5 mins.**

And with the following (without breaking between lines):

```
turtle.step 10 step 15
```

Conclude that spelling, punctuation, capitalization and entering lines are part of the syntax that is essential for our code to do what we want.
Finally, run a 3-star solution:

```
turtle.step 10
step 15
```

When it completes, click “replay” and change it to the following:

```
turtle.step 10
monkey.step 15
```

Conclude with your students that step and monkey.step can be used interchangeably because the computer assumes we are referring to the monkey. When we refer to the turtle or any other object, we must use its name.

Part 2: 30 Minutes

Playtime

Log-in	**1 min.**
If needed, review log-in instructions here.	
Playtime	**29 mins.**
All students should complete challenges 16-20 with at least two stars. Students 12 and up should get three stars. Use your classroom dashboard to keep track of student achievements. Note that challenge 16 is a tricky one to achieve three stars in. Make sure your students do not stay on this challenge for too long and encourage them to keep going and come back to it if they have time left. At the end of the lesson, you can open a discussion regarding this challenge and try to solve it together with your students in order to get those sneaky three stars.	

Part 2: 30 Minutes

Playtime Cont.

Playtime Cont.	**29 mins.**

In challenge 19, there are different ways to make the turtle turn the correct way after catching three bananas. One way is by using the island :

```
turtle.turnTo island
```

and another way is by using any of the bananas along that path:

```
turtle.turnTo bananas[3]
```

In both cases, hovering and/or clicking will do the trick. Remind your class that hovering over an object shows its name, and clicking enters that name into the editor.

If your students ask you about the meaning of something like bananas[3], tell them that it is the way to access a particular banana and we will get back to it later on.

Part 2: 30 Minutes

Playtime Cont.

Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 16-20, skill challenges 2-6 – 2-10 are unlocked.	

Part 3: 5 Minutes

Debriefing

Walk-through	**5 mins.**
Open challenge 2-7 in skill mode and solve it with your class. Ask them to explain how they plan the solution for this challenge. You can even invite a student to solve the challenge in front of the class. The trick in this challenge is similar to the one in challenge 16 - tell the monkey to walk backwards in order to have less lines of code, and to get the third star. If at first try your students cannot get the third star, ask them if this challenge seems similar to one they have solved before. Explain that it is fairly common to use references from old projects when programming, or even full blocks of code, and in Coding Adventure, they are encouraged to go back to old challenges to get inspiration or help.	

Lesson 5 – In the Loop

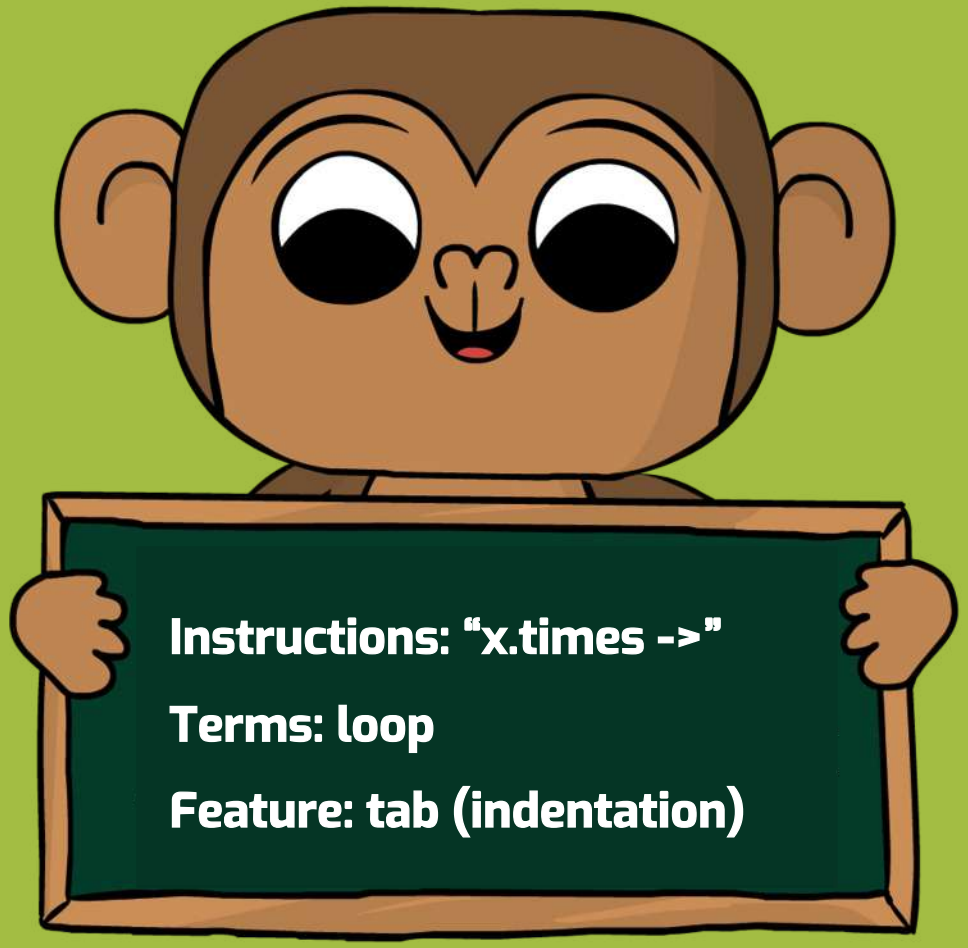
Congratulations! You have passed the introductory part of Coding Adventure. You and your students now hold basic programming skills. This lesson will focus on loops. There are different kinds of loops, like “for loops” and “until loops”, but first we will learn how to use a simple loop.

Objectives

In this lesson, students will:

- Define *loop* as a programming term
- Understand why using loops in programming is more efficient
- Complete challenges 21-25

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 25 Minutes

Introduction

Discussion	**10 mins.**

Programming is not only about writing the correct statements in the correct order, it is also about knowing how to write clear and short code.
Imagine that we have to write a simple program to make the monkey climb up a high staircase of 100 steps and we can only use the function “stepUp” to make the monkey climb up one step at a time.

Ask your students: “Do you think that the programmer wrote a line of code for every stair-step?” Just imagine how LONG this code would be ! It would be 100 lines of code!
So, instead of code that looks like this (X100):

```
stepUp
stepUp
stepUp
etc...
```

Wouldn’t it be great to write something shorter? Ask your students to suggest a shorter way.
How about something like this?

```
stepUp 100 times
```

Luckily, this is possible. Not exactly the way we just wrote it now, but quite similar. Code that is written in such a way is called a loop.

Part 1: 25 Minutes

Introduction Cont.

Explanation	**5 mins.**

Explain to your class that a “simple loop” is a sequence of instructions that repeats a specified number of times. There are also other kinds of loops (for loops, until loops) that last until a particular condition is met, but we will learn about those later on.
Back to the staircase example…the way to write that in Coding Adventure would be:

```
100.times ->
    stepUp
```

The number 100 represents the number of times that we want the code inside the loop to run for.

Note the special syntax: the dot between the number and the word times, the space before the ->, and the indentation of the code inside the loop (stepUp is the code inside the loop). Make sure your students know how to use the “Tab” key on their keyboards in order to get indentation into the code. Another alternative is to press the spacebar four times.

Remember that you can click the “times” button at the bottom in order to get a loop into the code without having to worry about the syntax.

Part 1: 25 Minutes

Introduction Cont.

Activity	**10 mins.**

Let's show another example to better clarify the use of a simple loop.
Write the following code on the left-hand side of the board:

```
step 10
turn left
step 10
turn left
step 10
turn left
step 10
turn left
```

Part 1: 25 Minutes

Introduction Cont.

Activity Cont.	**10 mins.**

Ask your students to identify a repeating pattern in the code. The pattern they identify should be:

```
step 10
turn left
```

Now, next to that code, on the right-hand side, write:

```
4.times->
    step 10
    turn left
```

Ask your students what they think each block of code does.

Explain that each code is the same; only the right-hand side of the code is written as a loop. Once we found the pattern on the left hand side, all we had to do was just write it once, and add 4.times->. The resulting code does the same, but is shorter and easier to understand.

The meaning of the code on the right is that “step 10, turn left” would repeat four times, and then the loop would be over. Once the loop is over, the computer moves onto the next statement.

Part 2: 15 Minutes

Playtime

Login	**1 min.**
If needed, review log-in instructions here.	
Playtime	**14 mins.**
All students should complete challenges 21-25 with at least two stars. Use your classroom dashboard to keep track of student achievements.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 21-25, skill challenges 3-1 – 3-5 are unlocked.	

Part 3: 5 Minutes

Debriefing

Walk-through	**3 mins.**

Open challenge 25 and click the reset button to reset the code. Go over the code with your students. Read the statements out loud, slowly and clearly. This walk-through is intended to show students how to read code correctly.

Walk your students through the process of identifying the pattern of bananas arranged in an L shape and translate that into the sequence of statements:

```
turn left
step 5
turn right
step 5
```

Then, fix the inside of the loop to match the sequence of statements and hit “run". This will not solve the challenge as the loop runs 3 times instead of 4.

Ask your students how many times the L pattern repeats itself? The solution is the last hint needed in order to solve this challenge correctly, i.e. replacing the 3 by 4.

Part 3: 5 Minutes

Debriefing Cont.

Explanation	**2 mins.**

Imagine you had to give instructions to somebody to find a place that is 5 blocks down the street.
Do you say: walk a block, then walk another block, then another, then another, and then one more. No. You simply say: walk 5 blocks down the street. That is because the same action has to be done more than once.

Remind your students that it is the same in coding. When there is a repeating pattern of things to do, then a loop is a good way to keep the program short and easy to understand. Just find the pattern, write it once and add the line of code that tells the computer how many times to repeat.

Lesson 6 – Loop on

Today your students will continue using simple loops and will deepen their understanding on why it is important to use loops.

Objectives

In this lesson, students will:

- Understand why using loops in programming is more efficient
- Understand the importance of using indentation correctly
- Complete challenges 26-30

Components

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 10 Minutes

Introduction

Review	**10 mins.**

We will begin this lesson with a review of loops and how we use them.
Ask your class:

1. What is a loop?
2. When do we use loops in our code?
3. What is a syntax of a loop that makes the monkey walk in the shape of a square?
4. How do we tell the difference between code that is inside the loop and code that is outside the loop?

Part 2: 30 Minutes

Playtime

Log-in	**1 min.**
If needed, review log-in instructions here.	
Playtime	**4 mins.**
All students should complete challenges 26-30 with at least two stars. Students from the age of 12 and up should get three stars. Use your classroom dashboard to keep track of students' achievements. The main thing to watch in this playtime session is the use of loops. Most of the challenges that involve a repeating pattern can be solved by repeating the same or similar code. However, this misses the point, so you should make sure the students are actually using loops and getting at least 2 stars.	

Part 2: 30 Minutes

Playtime Cont.

Walk-through (1)	**5 mins.**

Open challenge 27.

1. Draw the challenge on the whiteboard. You can use a shape (smiley, etc.) to represent the monkey.
2. Ask one of your students to draw on the whiteboard and guess the path that the monkey has to go to get all the bananas. The answer should be a + shaped path.
3. Ask another student to translate the path into a sequence of steps and turns without loops. The answer is something similar to the following (the direction right or left can be different).

```
step 10
step -10
turn right
step 10
step -10
turn right
step 10
step -10
turn right
step 10
step -10
turn right
```

Part 2: 30 Minutes

Playtime Cont.

Walk-through (1) Cont.	**5 mins.**

The answer may be longer like in the following example:

```
step 10
turn 180
step 10
turn right
step 10
turn 180
step 10
turn right
step 10
turn 180
step 10
turn right
step 10
step 180
step 10
turn right
```

Part 2: 30 Minutes

Playtime Cont.

Walk-through (1) Cont.	**5 mins.**

If so, ask the student to improve the code by making it shorter. A hint on how to do this would be to walk backwards.
Ask another student to identify the recurring pattern.

The recurring pattern in the example above is:

```
step 10
step -10
turn right
```

Ask the student to make the code shorter by using a loop, expect an answer like this:

```
4.times ->
    step 10
    step -10
    turn right
```

Part 2: 30 Minutes

Playtime Cont.

Playtime	**4 mins.**
The students continue their work on challenges 26-30.	
Walk-through (2)	**5 mins.**
Open challenge 28 and click “reset” to reset the code. Observe with your students that the existing code will go through the half-circle and collect all the bananas except one. Conclude that the missing line of code to complete the challenge is “step 10” (use ruler if necessary). Try to type “step 10” and click “run” to see what happens. This does not solve the challenge (hit “stop” if this goes on too long). Observe that the problem in this solution is that the computer takes the “step 10” as if it should be executed 10 times in the loop, not 1 time after the loop. Go back to your code and remove the indentation before the step 10. Now run your code again. Show your students that this is how to run code after the loop. Remind them that this is called indentation.	

Part 2: 30 Minutes

Playtime Cont.

Playtime Cont.	**11 mins.**
The students continue their work on challenges 26-30.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 26-30, skill challenges 3-6 – 3-10 are unlocked.	
Quiz	
After completing challenges 0 – 30, you can assign your class the first quiz – Part 1: Sequencing, Objects & Times Loops. The quiz includes 5 challenges. You can assign quizzes from the Quizzes tab on your teacher dashboard.	

Part 3: 5 Minutes

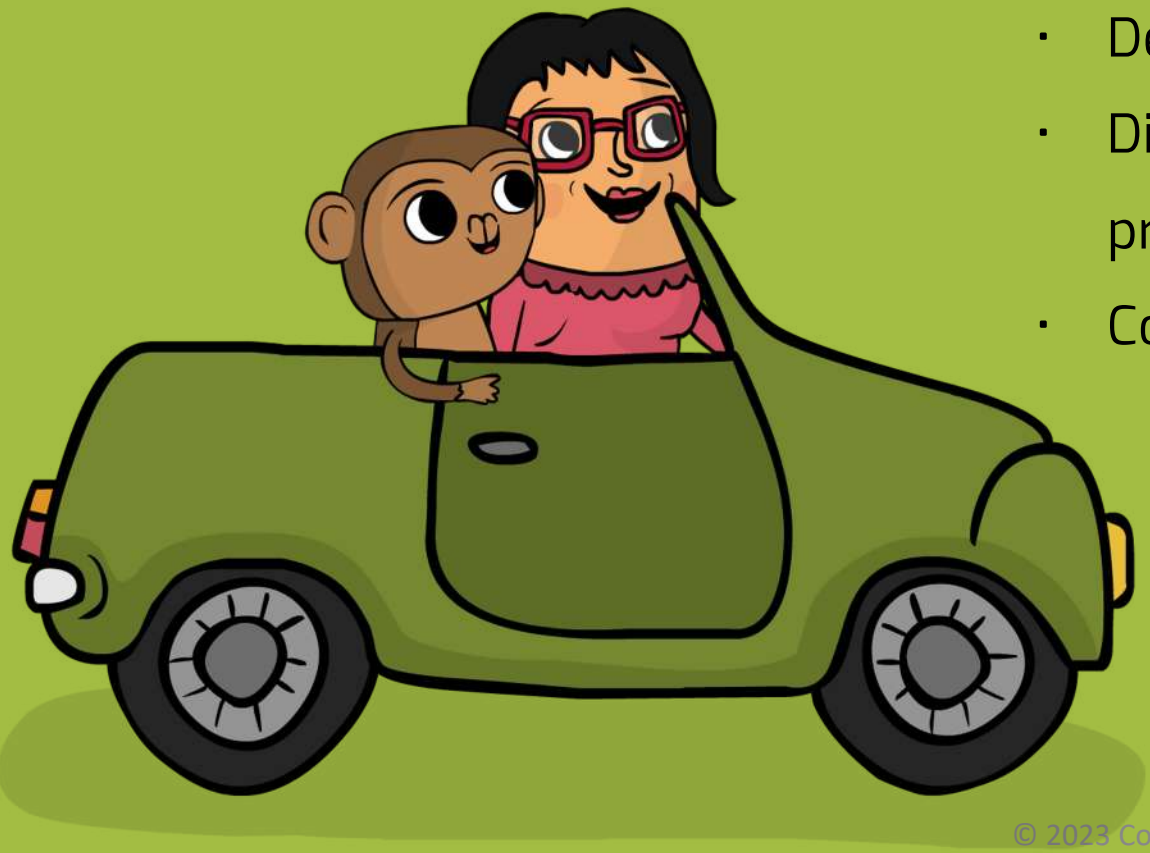
Debriefing

Walk-through	**5 mins.**
Open challenge 29 and click the reset button to reset the code. Go over the code with your students. Read the statements out loud, slowly and clearly. This walk-through is intended to show students how to read code correctly. Click “run” to run the code and direct your students’ attention to the orange highlighter that highlights the line of code that is currently being processed by the computer. Solve challenge #29 with your students. You can also open one of your student’s solutions anonymously using your classroom dashboard.	

Lesson 7 – Variable Valley

Welcome to the second chapter of CodeMonkey. You and your students will start covering advanced topics such as variables and “for” loops.

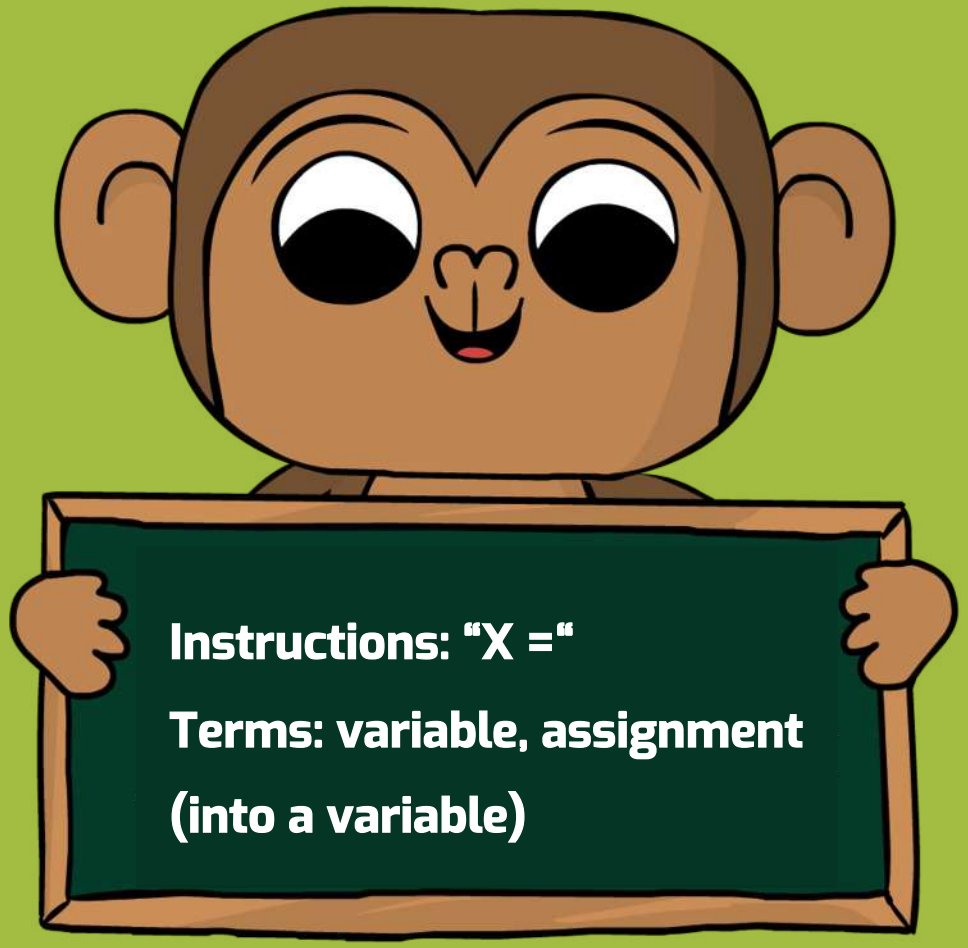
Objectives



In this lesson, students will:

- Define *variable* as a programming term
- Discuss how and why we use variables in programming
- Complete challenges 31-35

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 20 Minutes

Introduction

Activity (1)	**5 mins.**

Begin by asking for two student volunteers – Student 1 (who will act as the instruction given) and Student 2 (who will act as the variable). You, the teacher, will act as the programmer. Student 1 will be in charge of the instruction to 'step' the distance of the given variable and Student 2 will be given a folded paper with the value of the variable.

Let your class know that Student 1 is representing the instruction part of the code. She knows that her instruction is to 'step'. However, she does not know the distance she needs to step. Then, tell the class that you are handing Student 2 a folded piece of paper with a number between one and 10 on it. This paper will say how many steps Student 1 will need to take.

Instruct Student 2 to give the paper to Student 1 when you say start.

Explain to the class what is going to happen: First, Student 2 will give the note to Student 1. Then, Student 1 is going to step the number of steps written on the paper. Emphasize the fact that Student 1 only knows her instruction is to step, but she does not know how far yet. Now, instruct them to start.

After they are done, ask your class, "What do they think was the meaning of this activity?" Discuss the fact that when the instructions were given to Student 1, the number of steps was unknown to her, but it was still possible to give her the instructions. That is just how a variable works in computer programming.

Part 1: 20 Minutes

Introduction Cont.

Activity (2)	**5 mins.**
Repeat the previous activity with 2 other students, but with the following difference: This time Student 2 will write the number on the paper. Point out the fact that even though you, the programmer, does not know the number of steps Student 1 will need to take, you are still able to give the instructions. This is all an analogy for a variable – a programmer can give out commands or instructions without writing an exact value. They simply do this by using variables.	

Part 1: 20 Minutes

Introduction Cont.

Explanation	**10 mins.**

Explain that in previous lessons we learned how to call functions when they take a constant value as an argument, for example, “step10” or “step20”. Another way to call functions is by using a variable.
A variable is like a storage unit; we store data in it, and we use it only when we need it. It is similar to the piece of paper we had in the activity, in that it can hold a value for later use.
To store information in a variable, we write an assignment using an equals sign (=). This is like writing a number on the piece of paper. An assignment statement has two components: identifier and value, for example:

X = 20

X is the identifier; it can be any other letter or word. The identifier is the variable’s name. When we want to use the value of the variable, we write its name, for example, “Step X”. In this example, we want the monkey to step a distance equal to the value of the variable; in this case, 20. This separation of name and value allows the name to be used independently of the information it represents. We can use X when writing the program, without knowing what its value will be when the instructions will be carried out.

Optional: Return to the activity by asking Student 2 to write a new number on the paper, and go through this process a few more times to make the point clear. This activity is intended to represent how variables work in programming. In this example, Student 1 knew the action she needed to carry out, but she didn’t have the value until Student 2 gave her the paper.

Part 2: 15 Minutes

Playtime

Login	**1 min.**
If needed, review log-in instructions here.	
Playtime	**14 mins.**
All students should complete challenges 31-35 with at least two stars. Use your classroom dashboard to keep track of students' achievements. Use this time to walk around the class and help students who are struggling.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 31-35, skill challenges 4-1 – 4-3 are unlocked.	

Part 3: 10 Minutes

Debriefing

Walk-through	**7 mins.**

Open challenge 35. This is where we first had to define our own variable. Click the reset button to return the code to the initial version. Start by solving the challenge. Now, show a few more examples for 3-star solutions:
Change name of variable from x to something else. Use a few different options and run the solution each time. Emphasize that the name of the variable does not have to be 'x'. Change assignment into arithmetic such as x = 1+1. Emphasize that the computer is capable of computing arithmetic and other operations (that is why it was originally called a computer!).
Assign from one variable to another. For example:

```
y = 15
x = y
```

This does not give us 3 stars, but it shows how we can use different variables and assign one to the other, just like copying a number from one paper to another.

Activity	**3 mins.**

Repeat the activity from the beginning of the lesson, with the following difference:
Hand Student 2 the piece of paper with a number. Student 2 first copies it to another paper, then hands it to Student 1, then Student 1 walks. Explain how this is equivalent to:

```
y = 15
x = y
step x
```

Lesson 8 – Drop it!

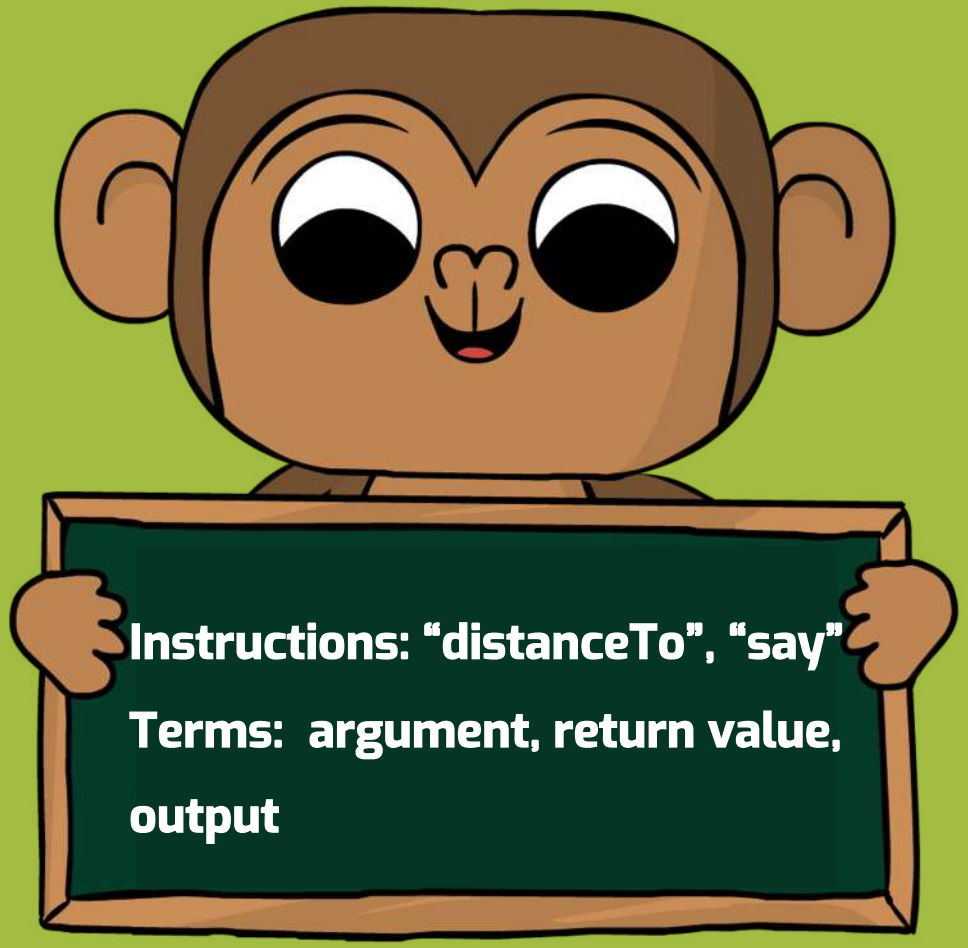
In this lesson, we will continue using variables. Variables are a very useful tool for programmers, and it is very important that your students understand how to use them. Alongside variables, we will learn how to use the “distanceTo” function.

Objectives

In this lesson, students will:

- Continue experimenting with variables
- Use the function “distanceTo”
- Complete challenges 36-40

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 15 Minutes

Introduction

Review	**5 mins.**

In the previous lesson, we learned about variables. Remind your students that variables are like paper that store numbers.

Assigning a variable is like writing on a number on a piece of paper. Assignments include an identifier, the name of the variable or a value, the number to store and an equal sign. We use variables when we do not have a constant value or when we do not know what the value will be until the action is carried out.

Ask your class what are possible options for the identifier, the variable's name, and make sure the answers are various words or single letters. The identifiers can not include spaces and cannot be the same as reserved keywords of the programming language (for, in, if, else, and, or, not, and a few others).

Part 1: 15 Minutes

Introduction Cont.

Activity	**5 mins.**
Ask for a student volunteer. On your mark he will need to step. This time, instead of giving him a value, give him an object to go to. Instruct the student to “step the distance to” an object in the classroom (a banana would be best!). Make sure he stands right in front of the object. Ask the class, “What do you think is the distance he walked?” Let them guess, but the number does not really matter. Explain that computers are more than just machines we play games on. Computers are really smart calculators. They can calculate very fast, and we can use it to our advantage when we need it. When we are using objects, we can ask the computer to calculate the distance between the objects for us. This will save us time and will also help us if we do not know in advance the objects’ locations.	

Part 1: 15 Minutes

Introduction Cont.

Explanation	**5 mins.**

The function we will use to make the computer calculate for us is “distanceTo”. This function is similar to the ones we used when we first started coding in Coding Adventure.

The function distanceTo is used with an object such as distanceTo banana, distanceTo bridge etc. This is similar to the way we use turnTo.

Using “distanceTo” is like asking the question “What is the distance to the banana?” The answer is a number, calculated by the computer, that represents the distance. This is called a “return value” because we are asking a question and receiving an answer (value) in return. When we use an assignment like this x = distanceTo banana then the return value is stored in the variable x.

We can also use “distanceTo” with another function like “step”. For example, “step distanceTo banana” will make the computer measure the distance between the monkey and the banana. Then, it will use this number to carry out as instructed using the measured value as the argument for “step”.

It is important to understand that the computer starts with the “distanceTo” function and evaluates the distance to the object (in this case the banana). Then, uses the number returned with the function “step”.

Part 2: 15 Minutes

Playtime

Log-in	**1 min.**
To review the log-in instructions, go here.	
Playtime	**14 mins.**
All students should complete challenges 36-40 with at least two stars. Use your classroom dashboard to keep track of student achievements. Use this time to walk around the class and help students who are struggling.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 36-40, skill challenges 4-4 – 4-11 are unlocked.	

Part 3: 15 Minutes

Debriefing

Walk-through	**10 mins.**

Open challenge 37. We will play around a little with the “say” function. This is a really cute function where your students can have fun and experiment.
Write something fun for the monkey to say. Ask your students, “Why do we have to use quotes (“ ”) around the phrase we want the monkey to say?”
Explain that in order for the computer to understand that the text we entered is not a variable, we have to use a symbol to tag it, which is why we use quotes.

Now, demonstrate return values.

Open challenge 38 and show a 3-star solution. Bring your students’ attention to the speech bubble with the number. Point out that this is the value stored in the variable x, which was put in there when we assigned x = 20.

Now open challenge 40 with a 3-star solution. In the second line of code, insert the statement say x. (Note: this will now be a two-star solution, but we use it for demonstration). Show your students how the value in the variable can be used as the argument for say, just like “boo” was an argument. Also, point out that we were able to solve the challenge without knowing what the distance was, but now by using say we were shown that the value of x is 20. Showing a value like this is called output.

Part 3: 15 Minutes

Debriefing Cont.

Activity	**5 mins.**
Optional (activity can take place outside): Let us play a guessing game. Ask your students to divide into pairs. Each student instructs the other to “step distance to” objects in the school, like the water fountain, stairs, etc. Before each student steps, he should guess how many steps it will take him, and while stepping he should count his steps to see if he was right.	

Lesson 9 – Walk the distanceTo

In this lesson, we will continue using the concepts we recently learned:

- Variables
- Functions and their return values
- The function distanceTo

Objectives

In this lesson, students will:

- Deepen their understanding of how to use variables and return values
- Complete challenges 41-45

Components

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-12 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 5 Minutes

Introduction

Review	**5 mins.**

Recall with your students what an assignment is, how an assignment statement is written, and what the different parts of an assignment statement is called.

An assignment places a value inside a variable. It is written in the form x = 10, where x is called the identifier, = is the equal sign, and 10 is the value.

Also, recall that sometimes calling a function is like asking a question and that the answer to that question is called a return value.

As we saw in the previous lesson, a return value can be used in an assignment to a variable. For example, x = distanceTo banana. It can also be used in an assignment to an argument, for example say distanceTo banana.
If we can say distanceTo, why can we not step distanceTo?

That is exactly what we will learn to do today!

Part 2: 30 Minutes

Playtime

Log-in	**1 min.**
To review the log-in instructions, go here.	
Playtime	**15 mins.**
All students should complete challenges 41-45 with at least two stars. Use your classroom dashboard to keep track of student achievements. Use this time to walk around the class and help students who are struggling.	

Part 2: 30 Minutes

Playtime Cont.

Walk-through (1)	**5 mins.**

A few minutes into playtime, ask for your students attention and open challenge #41 with a 3-star solution.

Discuss the line step -d. If your students have learned negative numbers before, explain that -d is like -10. Recall the analogy of variables to paper: if the d is the piece of paper, then -d is like writing the - sign in front of whatever number is written on the paper.

If your students have not learned negatives before, simply explain that step -d is like step -15 which was used a few lessons ago. It simply tells the turtle the same as step d, but backwards.

Replace turtle.distanceTo monkey by monkey.distanceTo turtle and run the solution again. Show that the result is the exact same, and explain that there is no difference whether you measure the distance from the turtle to the monkey or from the monkey to the turtle.

Part 2: 30 Minutes

Playtime Cont.

Walk-through (2)	**5 mins.**

Open challenge #42 with a 3-star solution.

Discuss step distanceTo banana.
Ask how many functions were used in this line of code?
The answer: 2 (step and distanceTo)

Ask which one was executed (ran) first? The answer is counter-intuitive: distanceTo. Explain that this is the order of operations:

1. The computer runs distanceTo banana
2. The distance is calculated
3. The computer runs step with the answer

Tell your students that this way of calling a function will be useful in challenges #44 and #45.

Part 2: 30 Minutes

Playtime Cont.

Playtime	**5 mins.**
All students should complete challenges 41-45 with at least two stars.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 41-45, skill challenges 4-12 – 4-15 are unlocked.	

Part 3: 10 Minutes

Debriefing

Walk-through	**5 mins.**

Open challenge #44 with a 3-star solution. Run the solution 2-3 times and show that the distance to the banana was different each time. The code did not change, but solved the challenge every time. The distance to the banana was unknown, but was calculated on the spot.

In order to emphasize this, let's experiment with this challenge.

Replace the step distanceTo banana with step 7

Run the solution a few times. Show that sometimes the distance was ok, but sometimes it was too short. So, this solution was not so great.

So what if we made the code a longer? This time, try replacing that line with step 10. The monkey will usually fall into the water.

The only way to get it right every time is to use step distanceTo banana

Part 3: 10 Minutes

Debriefing Cont.

Explanation	**5 mins.**

The values 10 and 7 are constant values. This means that once they are written into the code, they do not change.
When we write step 10 we are using the function “step” with a constant (10). Sometimes, circumstances are unknown when we are writing a computer program, like the distance we have to walk. The correct value will only be determined when the code runs, or in coding lingo - at runtime.

The solution in these cases is to use something else instead of the constant value.
Two such examples that we have done are:
1. step x- using the function step with a variable
2. step distanceTo banana - using the function step with a return value from another function

Lesson 10 – Change is All Around

In this lesson, we will continue using the concept we learned recently: variables. Specifically, the variable's value will be changed in the course of the game.

Objectives

In this lesson, students will:

- Deepen their understanding of how to use and change variables
- Complete challenges 46-50

Components

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 15 Minutes

Introduction

Review	**2 mins.**

Recall with your students the meaning of variables.

A variable is like a storage unit.
It has:

- An identifier (the name of the variable)
  - We use the name of the variable in the code
- A value (the content of the variable)
  - When our program runs, the computer uses the value of variable

When we want to use a variable, we need to give it a name and assign it an initial value. In our program we can change the value of the variable.

Part 1: 15 Minutes

Introduction Cont.

Activity	**8 mins.**

Ask for two volunteers. Let's call them Student 1 and Student 2.
Give Student 2 a small whiteboard, or you can also draw a square on the whiteboard. On it, write d = 2.
Tell Student 1 that she needs to repeat the following instructions four times:

```
step d
step -d
```

Before they start the activity ask your students what they think will happen?

- Student 1 will walk two steps forward and two steps backwards, and will repeat it four times.

Ask Student 1 to execute her code. Each time she reaches the variable d in her code, she needs to ask Student 2 "what is the value of d?"
In this activity we used a variable in the code but it did not change. Now let's change it. Tell Student 1 that her code is:

```
step d
step –d
d = d + 2
```

Tell her to repeat the above code three times.

Part 1: 15 Minutes

Introduction Cont.

Activity Cont.	**8 mins.**

Tell Student 2 that each time Student 1 gets to the line of code that changes d, he needs to delete the value of d and update it with the new value.
Before they start the activity, ask your students what they think will happen?

- Student 1 will walk:
  - two steps forward and two steps backward
  - four steps forward and four steps backward
  - six steps forward and six steps backward

Ask Student 1 to execute her code. Each time she reaches the variable d in her code, she needs to ask Student 2 what the value of d is.
When Student 1 gets to the line of code d = d + 2, Student 2 should say: 2 + 2 is 4, erase d = 2 and write instead d = 4. Then, the next time Student 2 should say 4 + 2 is 6, erase d = 4 and wrote d = 6 instead, and so on.
In this activity, the variable d is changed when the code runs, its initial value was 2 and after running the code, it is 8.

Repeat this activity, but change only the initial value of d to 5, ask your students how many steps Student 1 will walk each time? Their answer should be: 5, 7, 9.
What will be the value of d at the end? 11 (since d is changed after stepping)

Part 1: 15 Minutes

Introduction Cont.

Explanation	**5 mins.**

When we write the code

```
d = d + 2
```

each time that our program gets to this code, it takes the value of d, adds two to it and overrides the value of d with the new value.
When we use this code in a times loop, in each iteration, the value of d is different.
Write on the whiteboard:

```
d = 10
3.times ->
    say d
    d = d - 1
```

Ask your students what will the value of d each time? 10, 9, 8
Now change the code to:

```
d = 10
3.times ->
    d = d - 1
    say d
```

Part 1: 15 Minutes

Introduction Cont.

Explanation Cont.	**5 mins.**

Ask your students what will the value of d be each time? 9, 8, 7

Why is there a difference between the two examples?
The reason is, in the first example, we first say the value of d, then reduce its value by 1. In the second example, we first reduce the value of d by 1 and then say its value.

This will be the execution of the first example:

1. say 10, 10 - 1 is 9
2. say 9, 9 – 1 is 8
3. say 8, 8 – 1 is 7

This will be the execution of the second example:

4. 10 – 1 is 9 ; say 9
5. 9 – 1 is 8 ; say 8
6. 8 – 1 is 7 ; say 7

Part 2: 25 Minutes

Playtime

Log-in	**1 min.**
To review log-in instructions, go here.	
Playtime	**14 mins.**
All students should complete challenges 46-50 with at least two stars. Use your classroom dashboard to keep track of students' achievements. Use this time to walk around the class and help students who are struggling.	

Part 2: 25 Minutes

Playtime Cont.

Walk-through	**5 mins.**

Open challenge 48.
Ask your students if this challenge can be solved without a variable.

If we use a times loop then each time the monkey will step the same amount.
For example:

```
5.times ->
    step 8
    turn left
```

Run the code and show that the challenge is not solved.
You can also change the number of steps, for example to 12, or any other number and show that it does not solve the challenge.

Part 2: 25 Minutes

Playtime Cont.

Walk-through Cont.	**5 mins.**

We could solve this challenge without a loop:

```
step 8
turn left
step 12
turn left
step 16
turn left
step 20
turn left
step 24
turn left
```

This code is very long and only gets one star.
Ask the class how many more steps the monkey steps after each banana. Their answer should be four more steps.

Part 2: 25 Minutes

Playtime Cont.

Walk-through Cont.	**5 mins.**

Reset the code. Now, explain to your students that the monkey steps the value of d each time.
At first it is 8. Ask them how we can change the value of d to add 4 more each time?
We need to add to the code:

```
d = d + 4
```

There is also one more change in the code before we can solve it. The number of times the loop is repeated (need to be changed to 5).
The code should be:

```
d = 8
5 . times ->
    step d
    turn left
    d = d + 4
```

Run this code and solve the challenge.

Part 2: 25 Minutes

Playtime Cont.

Playtime	**5 mins.**
All students should complete challenges 46-50 with at least two stars.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 46-50, skill challenges 4-16 – 4-20 are unlocked.	

Part 3: 5 Minutes

Debriefing

Activity	**5 mins.**

Bring to class a few objects. For example, Legos, books, fruit, anything that students can collect. Bring a bag or basket to put all the objects in.
Place the objects around the class and ask for two volunteers.
Give one a small whiteboard, or you can also draw a square on the whiteboard. Write on it, c = 0.
Give the second volunteer the bag or basket and ask him to walk around the class and pick up the objects.
Each time he picks up an object, he should tell the first volunteer “one more”.
The first volunteer will erase the number written on the whiteboard and write the new number.
After picking the first object it will be c = 1, than c = 2, c = 3 and so on.
After all objects are picked up, ask your students how many objects are in the bag? Their answer should be the value of the variable c.
After completing the activity, ask your students how they would write the code that represents this activity?
Their answer should be something like:

```
c = 0
loop as long as there are more items
   go to an object
   put the object in the bag
   c = c + 1
say c
```

The most important part of this code is the first line (c = 0) and increasing the value of c by 1 (c = c + 1).

Lesson 11 – A for Array

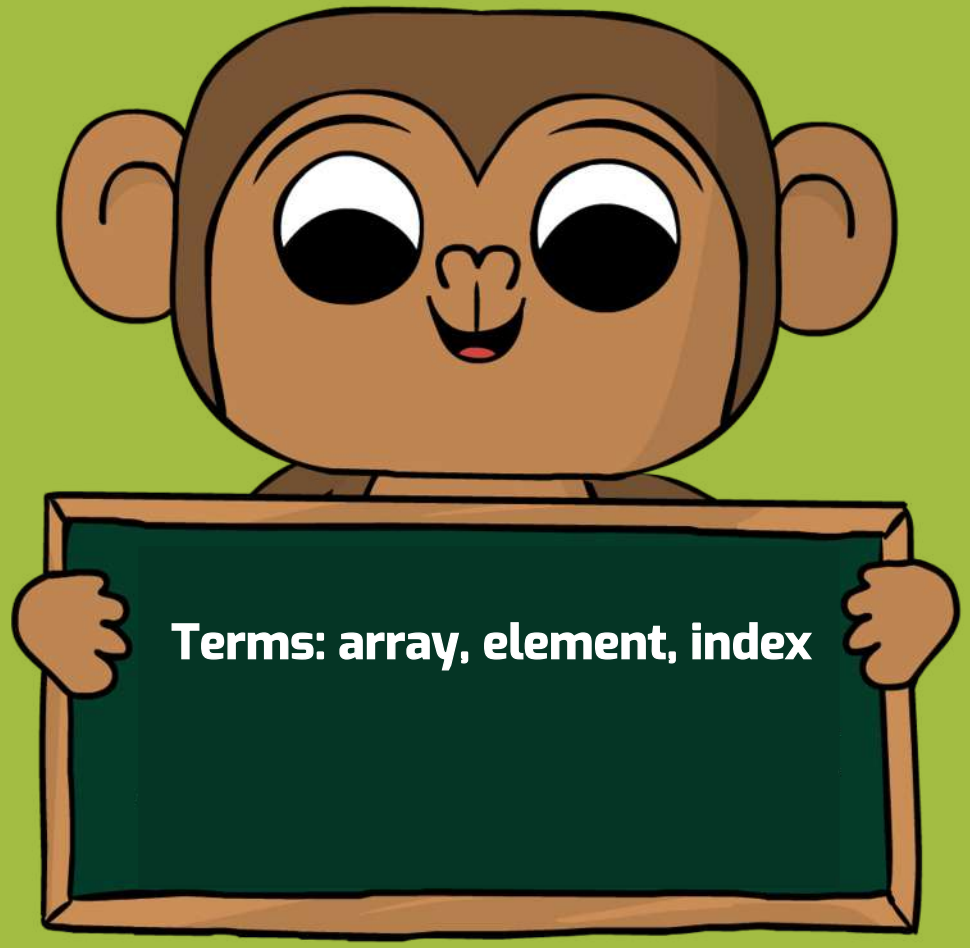
In this lesson we will start working with arrays. Arrays are a central and important part of programming and are used very often.

Objectives

In this lesson, students will:

- Understand the concept of an array and its elements
- Learn how to use indexes to access array elements
- Complete challenges 51-60

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 10 Minutes

Introduction

Activity	**5 mins.**

Optional: You can do the following activity outside.
Ask for 5 volunteers and assign a number to each of them.
Tell them that you will call them by their numbers e.g. “student 3”.
Have them stand in a location where each of them has some space to move around.
Give your students simple instructions such as:

student 1 step 2
student 2 turn right

Then make it more complex like this:

student 4 turn to student 3
student 2 step distance to student 1

Try saying just:

step

This probably confused the students a little because “step” did not specify which of the students should step.

Part 1: 10 Minutes

Introduction Cont.

Explanation	**5 mins.**

An array is a group of objects that share a common name. Usually, arrays contain objects of the same type, e.g. an array of bananas, an array of students. Each object in an array is called an element.

Every array has a name. It is common to name an array after the type of objects in it, e.g. bananas, students.

In the above activity, the students were an array. When we said “student 1” or “student 3” the number 1 or 3 was used to tell which element we were referring to. This number is called an index. In CoffeeScript, the programming language used in Coding Adventure, we start counting array elements from zero, so there would actually be a student 0, then student 1, etc. Starting indexes from 0 might seem annoying at first and take some getting used to, but it is the way most popular programming languages work today, for various reasons that were debated among mathematicians and computer scientists.

We will have to use indexes when we try to carry out an action that involves a particular array element. Whenever we want to involve all the elements at once, there will be easier ways that do not involve an index. We will learn that in the next lesson.

Part 2: 20 Minutes

Playtime

Log-in	**1 min.**
For log-in instructions, please click here.	
Playtime	**19 mins.**
All students should complete challenges 51-60 with at least two stars. Use your classroom dashboard to keep track of student achievements. Use this time to walk around the class and help students who are struggling.	

Part 2: 20 Minutes

Playtime Cont.

Practice	
Encourage students who finish early to open skill mode on the map and complete the unlocked challenges. After completing challenges 51-60, skill challenges 5-1 – 5-10 are unlocked.	
Quiz	
After completing challenges 31 – 60, you can assign your class the second quiz – Part 1: Variables & Indexes. The quiz includes 5 challenges. You can assign quizzes from the Quizzes tab on your teacher dashboard.	

Part 3: 15 Minutes

Debriefing

Walk-through (1)	**7 mins.**

Open challenge 55 with the following solution and run it.

```
beavers[0].step 10
beavers[1].step 10
monkey.step distanceTo banana
```

Ask your class how come the computer knows which object has to step each time?
The answer is because of the name of the object before the word step. That name tells the computer which object has to step. Point your students' attention to the fact that the first beaver is number 0, and the second is number 1. It is very important to remember that the we start counting from zero in CoffeeScript.

Can we use step without the word monkey or without the word beaver?
Experiment with the following variations:

```
beavers[0].step 10
beavers[1].step 10
step distanceTo banana
```

The above will work fine.

Part 3: 15 Minutes

Debriefing Cont.

Walk-through (1) Cont.	**7 mins.**

The previous code worked, but the following will not:

```
beavers[0].step 10
step 10
monkey.step distanceTo banana
```

The answer is that when the monkey has to turn or step, we can always use either:

```
monkey.step 10
```

Or instead simply:

```
step 10
```

The computer will assume that we are referring to the monkey, because the monkey is the main character in this challenge.
When it comes to other characters, we need to explicitly write which one we want to use, otherwise, the computer will not know. This is just like in the beginning of the lesson when we said “step” without saying which student has to step.

Part 3: 15 Minutes

Debriefing Cont.

Walk-through (2)	**7 mins.**

Open challenge 59 with a 3-star solution, and make sure all the beavers are moved before the monkey steps, like so:

```
beavers[0].step 5
beavers[1].step 5
beavers[2].step 5
beavers[3].step 5
monkey.step distanceTo bananas[0]
monkey.turnTo bananas[1]
monkey.step distanceTo bananas[1]
```

Now, change the code in the following way by cutting and pasting the last 2 statements with the beavers like this:

```
beavers[0].step 5
beavers[1].step 5
monkey.step distanceTo bananas[0]
monkey.turnTo bananas[1]
beavers[2].step 5
beavers[3].step 5
monkey.step distanceTo bananas[1]
```

Part 3: 15 Minutes

Debriefing Cont.

Walk-through (2)	**7 mins.**

Before running this solution, ask your students what they think will happen now. Then, run the solution and see if they got it right.

Repeat this activity with challenge 60.
Start with this code:

```
beavers[0].step 5
beavers[1].step 5
monkey.step distanceTo bananas[0]
monkey.turnTo bananas[1]
monkey.step distanceTo bananas[1]
beavers[0].step 10
beavers[1].step 10
monkey.turnTo bananas[2]
monkey.step distanceTo bananas[2]
```

Ask your students if there is a way to put all the beaver statements together. The answer is that it is not possible. The monkey has to move after the beavers have moved once, and then they must move again before the monkey moves again.

Part 3: 15 Minutes

Debriefing Cont.

Explanation	1 min.
Just like in Lesson 3 when we learned about planning, these examples were about writing code in the correct order of operations. What we saw in these examples is that if we do not use a correct order, the code will not run as we want it to. However, there might be more than one correct order to write a certain program.	

Lesson 12 – For Loop to the Rescue

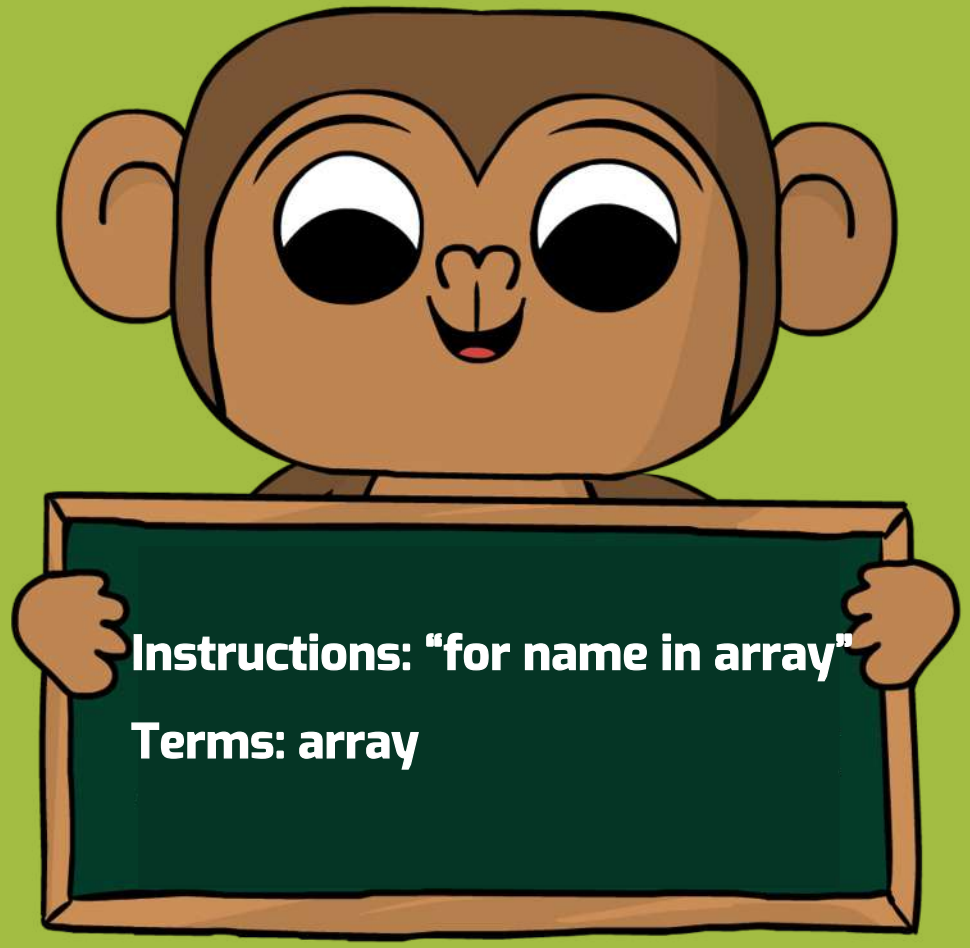
In this lesson, we will introduce a new kind of loop. By now, you have already mastered basic functions like “step” and “turn”, simple loops, and variables. You will apply your new knowledge as you use the “for” loop. The “for” loop might look intimidating, but it is actually quite simple.

Objectives

In this lesson, students will:

- Identify the difference between a simple loop and a “for” loop
- Discuss how and when to use “for” loops
- Complete challenges 61-65

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 15 Minutes

Introduction

Activity (1)	**5 mins.**

Choose a student volunteer and ask him or her to stand next to you. Choose six more students and ask them to stand next to each other in a line, facing forward, and raising their hands. The first student should stand at the beginning of the line and on your mark, should follow this code:

```
6.times - >
    step 1
    give high-five
```

The result should be that the first student gives a high-five to all six students standing in the line.
Next, ask the same student to “give high-fives to all the students in the classroom”. After he is done, ask your students, “What is the difference between the two instructions I gave?”

Explain that the first instruction you gave is a simple loop, the same as we learned in Lesson 4: In the Loop. It repeats the statement six times carrying out the exact same action each time.

Part 1: 15 Minutes

Introduction Cont.

Activity (1) Cont.	**5 mins.**

The second instruction was also a loop, but a different kind. The student walked different distances in different directions to reach each fellow student. The action carried out each time depended on that particular student. In programming, we use a “for” loop to complete tasks like this.

This is where the “for” loop is useful - when we have a collection of objects and we want to repeat an action that relates to each one of them specifically. This is a loop that will keep going until all the actions are done on all the objects in our collection.

In the previous exercise, the collection contained all the students who are in class today, and your student gave a high-five to each of them, by turning to face each particular student and walking the distance to that particular student.

An important difference to note here is that a simple loop is usually used to do the same thing a fixed number of times, whereas the “for” loop matches the number of items in a collection.

Part 1: 15 Minutes

Introduction Cont.

Activity (2)	**5 mins.**

Let's imagine we have a birthday cake with 6 candles on it, and the programmer has to write a program that blows out the candles. Ask one of your students to use a simple loop to blow out all the candles, just like in the stairs example from lesson 5.
You can expect code like this:

```
6.times - >
    blow candles
```

Ask your students, "Now let's make this a little bit more complicated: every time, we have to face one particular candle in order to blow it out. How do we write that?" Explain that because it is a simple loop, there is no way to turn to face a particular candle.

If we want the program to adjust to the proposed change in circumstance, we can use a "for" loop. For this, we will write:

```
for c in candles
  turnTo c
  blow c
```

Part 1: 15 Minutes

Introduction Cont.

Explanation	**5 mins.**

The “for” loop has two components: loop variable and array. The loop variable is a name we assign; it can be any name we want, like in other variables. It is common to name it after the first letter of the array.

Recall that an array is a collection containing objects. For example, we have an array containing six bananas. Each banana gets its own name, and all the bananas together have one name, which is the name of the array.

Open challenge 61 to demonstrate this point. Click on one of the bananas and two buttons will appear next to it. The upper one is the name of this specific banana. You can click a different one and you will see that each banana has a different number in its name. The lower button is the name of the array. These six bananas belong to an array called “bananas,” so no matter what banana we click, for all of them the lower button will say “bananas.”

Part 1: 15 Minutes

Introduction Cont.

Explanation	**5 mins.**

Now read the code on the right of the screen:

for (this tells the computer that there is a “for” loop here)
b (definition of the loop variable)
in (part of the “for” loop)
bananas (the name of the array)
turnTo b (b is the loop variable that refers to an object in the array)
step distanceTo b (b is the loop variable that refers to an object in the array)

Instead of writing the full name of the list next to every function, we must use the name of the loop variable which we defined in the “for” loop.

Part 2: 25 Minutes

Playtime

Log-in	**1 min.**
If needed, review log-in instructions here.	
Playtime	**24 mins.**
All students should complete challenges 61-65 with at least two stars. Use your classroom dashboard to keep track of students' achievements. Use this time to walk around the class and help students who are struggling.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 61-65, skill challenges 6-1 – 6-3 are unlocked.	

Part 3: 5 Minutes

Debriefing

Walk-through	**3 mins.**

Open challenge 6-2 from skill mode.
Ask your students how they think this challenge should be solved. Let them debate and offer a few suggestions.

The trick in this challenge is to pay attention to the way the bananas are indexed. Click on the top left banana and show that its name is “banana[0]”.

Click on the one next to it, and show that its name is “banana[2]”, meaning that “banana[1]” is placed somewhere else, probably under the bridge. Keep going until the path the monkey should make is clear, and solve this challenge with your students:

```
for b in bananas
    turnTo bridge
    step distanceTo bridge
    turnTo b
    step distanceTo b
```

Part 3: 5 Minutes

Debriefing Cont.

Explanation	**2 mins.**
Remind your students what they learned in the previous lesson – the computer knows which object is first because of the index which is visible in the object's name. We start at 0, then 1, 2 etc. Whenever we want to involve all the elements of an array at once, like the six bananas in challenge 6-2, we have to take into consideration the indexes of the array elements.	

Lesson 13 – Iterate Mate

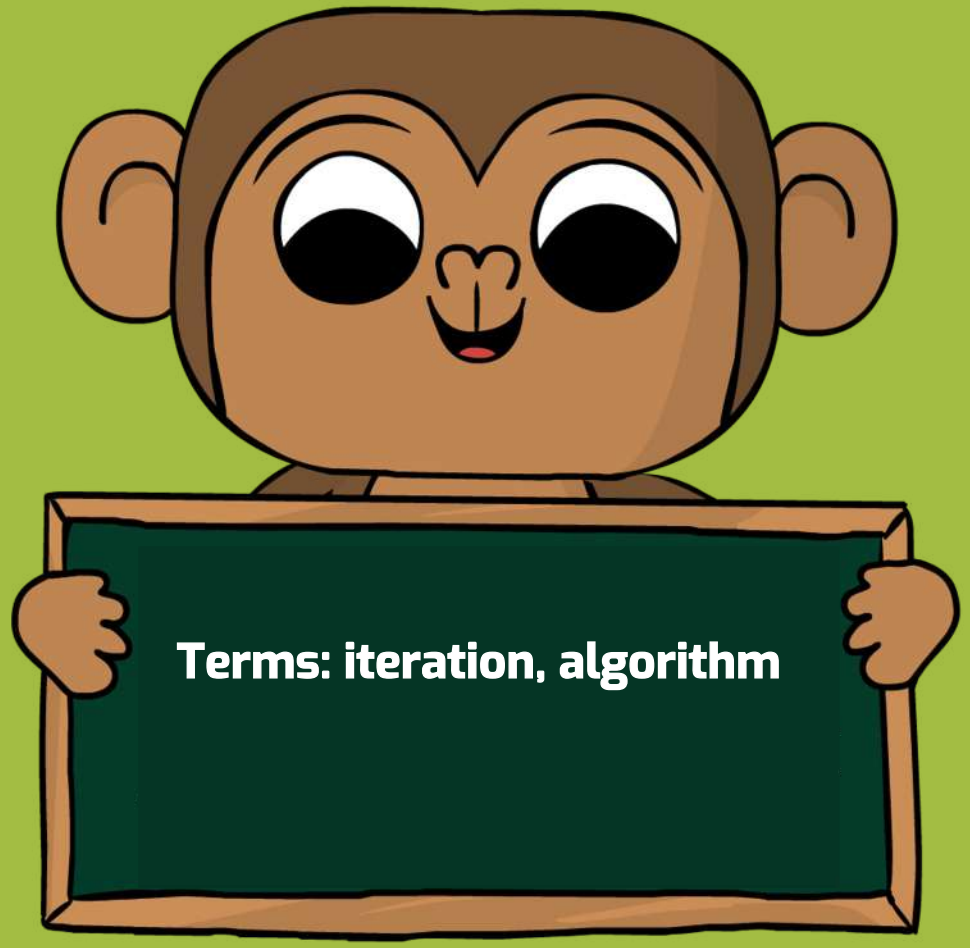
In the previous lesson, your students were introduced to “for loops”, and through them to the concept of iteration over a collection of objects. This lesson continues the work with for loops. Iterating over collections of objects is such a fundamental concept in computer science and is used so frequently in writing code, that it is worthwhile to spend some more time to practice using it.

Objectives

In this lesson, students will:

- Learn how to use the loop variable in other ways than passing it as an argument
- Practice for loops
- Complete challenges 66-70

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 10 Minutes

Introduction

Activity (1)	**6 mins.**

Recall with your students that a “for loop” repeats an action for every element of a collection, for example turning to and stepping to every banana on the screen. This process is called iteration. We say that we iterate over the collection of bananas. We can also iterate over a collection of other types of objects and do other types of operations with them. For instance, we can iterate over a collection of students and tell each one of them to step.

Ask for three volunteers and tell them they belong to the collection “students”. Instruct them to stand shoulder to shoulder with their backs to the wall. Ask a fourth student to write on the whiteboard a block of code that makes all of them take one step forward. Ask for a solution without a loop, first.

Part 1: 10 Minutes

Introduction Cont.

Activity (1) Cont.	**6 mins.**

Note: an answer such as students.step 1 is not acceptable because the array does not have a function that causes all its members to step. The correct answer would be:

```
students[0].step 1
students[1].step 1
students[2].step 1
```

Remember that the counting starts from 0!

Now, ask a fifth student to write a solution with a loop. The answer should be something like the following. (Pay attention to indentation.)

```
for s in students
    s.step 1
```

Ask your students to identify the array and loop variable.

Answer: the array is students and the loop variable is s.

Explain that this is a new way to use loops: the loop variable is before the dot and the function name because rather than carrying out an action on the array element (e.g. turning to a banana) we are instructing the array element to carry out an action (tell the student to step).

Part 1: 10 Minutes

Introduction Cont.

Activity (2)	**4 mins.**

Ask one of the students to stand with their back to the wall, at a distance that is more than one step, but that is not specified.

Ask other students if they can guess how to use the minus sign (-) to write one line of code that would make that student step backwards to the wall without turning. They can use the words “wall”, “step”, “distanceTo”, and “student”. Point out to the students that we do not know the distance to the wall and are not supposed to guess.
The correct answer is
student.step -distanceTo wall

Why is the - in front of the word distanceTo?
Explain the process that happens in the computer when this line of code is executed:

The computer calculates “distanceTo wall”, and replaces the words “distanceTo wall” in the code by the answer. For example, if the distance is 2 the result is student.step -2. Then the computer executes the function step with the argument -2 which is translated to “2 steps backwards”. Altogether, it is as if “step -distanceTo wall” is translated to “step backwards distanceTo wall”.

Part 2: 25 Minutes

Playtime

Log-in	**1 min.**
If needed, review log-in instructions here.	
Playtime	**5 mins.**
All students should complete challenges 66-70 with at least two stars. Use your classroom dashboard to keep track of students' achievements. Use this time to walk around the class and help students who are struggling.	

Part 2: 25 Minutes

Playtime Cont.

Walk-through (1)	**5 mins.**

When your students are at or near challenge 67, call for their attention. Open challenge 67 and click reset. Show the students that the initial code has “turnTo raft” in the inside of the loop. Tell them that in order to solve this challenge with 3 stars, it is important not to delete this code, but instead to base your solution on it.

Solve the challenge together with your students.

Copy the layout of this challenge onto the whiteboard and ask a student to draw the route of the monkey and use a different color each time the inside of the loop is executed.
Each one of the colors will show a path that starts at an end, goes to the raft at the center and goes on to the next end. Remind your students that this process is called iterating over the array of bananas.
Tell them that each time the inside of the loop is executed, i.e. each color, is called a single iteration.

Part 2: 25 Minutes

Playtime Cont.

Playtime	**14 mins.**
Students continue working on challenges 66-70.	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 66-70, skill challenges 6-4 – 6-7 are unlocked.	

Part 3: 10 Minutes

Debriefing

Walk-through (1)	**2 mins.**
Open challenge #68 and reset the code. Ask your students how they solved this challenge. You can use your classroom dashboard to anonymously show different solutions your students wrote. Explain that the trick in this challenge is understanding that the array we are dealing with contains the islands and not the bananas. Show the correct solution.	

Part 3: 10 Minutes

Debriefing Cont.

Walk-through (2)	**2 mins.**

Open challenge #70 and reset the code.

Solve the challenge with your students and use the negative sign (-) in order to get the monkey back to the turtle every time without having to turn. This way you will get a 3-star score.

Optional: In order to further demonstrate the point about the negative sign, especially if your students have learned negative numbers and basic algebra, you can show them the following solution for challenge 70. It is also a 3-star solution, and it shows how to use a negative sign in front of a variable in order to negate the value of the variable:

```
for b in bananas
    x = distanceTo b
    step x
    step –x
    turtle.step 8
```

Part 3: 10 Minutes

Debriefing Cont.

Explanation	**6 mins.**
Explain to your students that an algorithm is a step-by-step set of operations that are performed in order to solve a problem or to get a certain task done. Explain that in computer programming, there are often many ways (algorithms) to accomplish any given task. An algorithm may have advantages and disadvantages in different situations compared to other algorithms that accomplish the same or similar tasks. Advantages may be related to time and resource consumption, readability, and differences in behavior or functionality. The solutions we write for each challenge are representations of algorithms. In other words: a sequence of statements describes the algorithm we want the computer to carry out, in a programming language. There are many ways of representing an algorithm, such as flow diagrams, pseudo-code, visual block-coding, and sequences of statements in various programming languages.	

Lesson 14 – Crocodile Rock

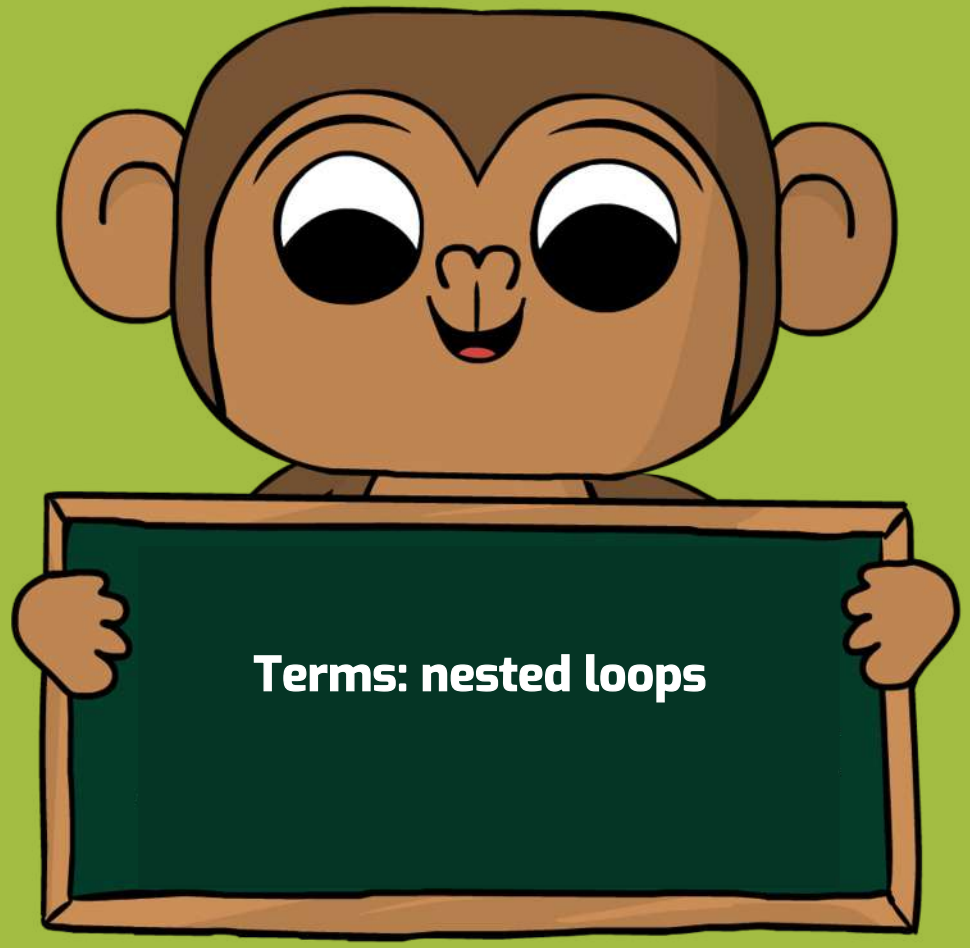
In this lesson, your students will continue to explore and practice their “for” loop writing skills. Challenge 75 marks the end of the second chapter of Coding Adventure.

Objectives

In this lesson, students will:

- Review the concept of *array indexing*
- Continue practicing with loops
- Complete the second part of CodeMonkey
- Use nested loops
- Complete challenges 71-75

Components



U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 10 Minutes

Introduction

Explanation (1)	**6 mins.**

Ask your students, “Do you think the order in which the monkey catches the bananas in the “for” loops is random or set in a particular way?”

The answer is that the order is set. As we have learned earlier in the course, each banana has its own number, called an index.

The index represents the banana’s location in the collection. The first banana in the collection is banana[0], the second banana is banana[1], and so on.

When the computer executes this loop, it essentially replaces the variable name with the first item in the collection. After it is done with the first item, it moves on to the second, and so on.

Part 1: 10 Minutes

Introduction Cont.

Explanation (1) Cont.	**6 mins.**

For example, we have this “for” loop:

```
for b in bananas
    turnTo b
    step distanceTo b
```

The first time this loop will run, the computer will “read” it like this:

```
turnTo banana[0]
step distanceTo banana[0]
```

Next, it would start over with the next banana:

```
turnTo banana[1]
step distanceTo banana[1]
```

And so on, until all bananas are caught.

Part 1: 10 Minutes

Introduction Cont.

Explanation (2)	**4 mins.**

Today your students will meet a new character - the Crocodile.
Crocodiles are used to form a bridge on water in order to help the monkey reach the bananas. Crocodiles can only “turn” or “turnTo”. We can create “for” loops for crocodiles as well. Instead of telling each crocodile to turn separately, we can write:

```
for c in crocodiles
    c.turnTo banana
```

This “for” loop is a little different. Remember when we had to tell the turtle to turtle.step? Well, crocodiles are the same. We have to tell them to crocodile.turnTo, which is why the loop’s variable is before the name of the function (before turnTo). Ask your students to guess how the computer will read this.

Answer: ...crocodile[0].turnTo banana, ...crocodile[1].turnTo banana etc.

Part 2: 30 Minutes

Playtime

Log-in	**1 min.**
For log-in instructions, click here.	
Playtime	**14 mins.**
All students should complete challenges 71-75 with at least two stars. Students 12 and up should complete all challenges with 3 stars. Use your classroom dashboard to keep track of student achievements. Use this time to walk around the class and help students who are struggling.	

Part 2: 30 Minutes

Playtime Cont.

Explanation	**5 mins.**

When your students are approaching challenge 75, ask for their attention to discuss one other topic.

A “nested loop” is a loop inside of a loop. When using a nested “for” loop, the inner loop is executed fully over and over again. For every element of the outer loop, the inner loop is executed from start to end, and then that whole process repeats for the next element in the outer loop, and so on.

Imagine we are looking for a person in a five-story building. Each level has a different number of rooms. How do we make sure we search every room on every floor? Show the following code. Note that everything after a pound symbol (#) is not executed, we will learn about this later. Pay attention to indentation.

```
for f in floors
    for r in roomsOf f
        search r # I go first
    goUp # I go only after “search” is done
```

This outer loop executes the inner loop for every floor. First, we will check all the rooms on the first floor. After we are done with all the rooms, we will move one floor up. The inner loop will then start over for the second floor, and so on.

Part 2: 30 Minutes

Playtime Cont.

Playtime	**10 mins.**
Students continue working on challenges 71-75	
Practice	
Encourage students who finish early to open skill mode on the map and complete unlocked challenges. After completing challenges 71-75, skill challenges 6-8 – 6-15 are unlocked.	
Quiz	
After completing challenges 61 – 75, you can assign your class the third quiz – Part 1: For Loops. The quiz includes 4 challenges. You can assign quizzes from the Quizzes tab on your teacher dashboard.	

Part 3: 5 Minutes

Debriefing

Explanation	**2 mins.**
Recall with your students that an algorithm is a step-by-step set of operations to be performed in order to solve a problem or to get a certain thing done. Remind your students that in computer programming, there are often many different ways (algorithms) to accomplish any given task. An algorithm may have advantages and disadvantages in different situations compared to other algorithms that accomplish the same or similar task. Advantages may be related to time and resource consumption, readability, and differences in behavior or functionality.	
Demonstration	**3 mins.**
Open challenge #74 and ask your students how they solved this challenge. There are two ways to solve this challenge; both will give three stars. Can your students guess the two ways? Answers: c.turnTo raft OR c.turnTo banana	

Lesson 15 – Let's Build Something!

This lesson introduces students to Challenge Builder. Using this platform, your students will be able to create their own Coding Adventure challenges. They will also be able to share and solve each other's challenges.

Objectives

In this lesson, students will:

- Become familiar with Challenge Builder
- Create their own Coding Adventure Challenge
- Solve challenges created by their classmates

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 20 Minutes

Introduction

Discussion	**10 mins.**
Ask your students: “What Computer Science concepts have we learned so far?” Here are a few: Objects, loops, variables. Ask students to give short explanations for each topic.	

Part 1: 20 Minutes

Introduction Cont.

Walk-through	**10 mins.**

Today you will get the opportunity to create your very own Coding Adventure challenges.

You can also review the Challenge Builder User Guide.

Complete the following steps to introduce your students to Challenge Builder:

1. Log into your CodeMonkey account and click on My Creations (https://app.codemonkey.com/creations)

2. Scroll down to “My Challenges” and click on “CREATE NEW”

3. A pop-up message will appear with Gordo. This is where you will write the instructions for your challenge. You can also edit this later on by clicking on Gordo. Click “OK”.

4. Explain that similarly to the game, in Challenge Builder, the left-side is the stage and the right-side is the editor (where we write the code).

Part 1: 20 Minutes

Introduction Cont.

Walk-through Cont.	**10 mins.**

5. The code in the right tab is important so we should not delete it. Click on the code with the mouse and you will see that it is locked. If we want to unlock, it we can click on the little lock next to the word “configurations”. But for now, keep it locked.

6. Look at what is written in the configuration tab. What do you think the meaning of this is? This code lets the computer know where to put the monkey and the banana on the stage. The “x” is responsible for the horizontal location, and the “y” is responsible for the vertical location. Click and drag the monkey and show how the numbers change.

7. “rotation” indicates the direction the monkey is facing. Hover over the monkey and you will see a small rotation icon, click and drag it and show how the numbers change on the left.

8. Click on the plus sign to add characters and objects to your challenge. Choose a few animals/objects and add them to your challenge. Show how their names and location were added to the code on the left.

8. The 3 colorful squares lets us choose the theme of our challenge. Your students probably know the themes from the game. They can choose between the green forest, the lake, the desert, snow and more. Choose a theme and click on “change”. Show how the name of the theme was added to the editor on the left.

Part 1: 20 Minutes

Introduction Cont.

Walk-through Cont.	**10 mins.**

10. Drag one of the objects on the screen to the trash can to see what happens. Show that if you try to drag the monkey to the trash, it will not be deleted.
11. Hover over one of the objects and click on the duplicate button. We can add as many objects as we want, but we can only have one monkey.
12. Hover over the buttons below to show that a negative sign appears. If you click on them, the button will not be visible to the person who will be solving this challenge. Accidentally deleted something that you need? no worries, just click on the plus sign and choose which buttons to show.
13. Click on the "Challenge Solution" tab to show your students that they can solve their own challenge inside the builder. If they want to share their challenge, they must solve it first.
14. Click on save. After the pop-up indicates that the challenge is saved, they students can continue editing the challenge. Note that the solution is saved by clicking on "SUBMIT" and not when you save the challenge.
14. If your students want to share their challenge (after solving it), they can click on "share", the entire classroom could play the challenge they build.
15. Show that you can see the challenge you created under "My Creations" tab (scroll down to "My Challenges") and that you can edit, delete or share them by clicking on the arrow in the top right corner of each challenge.

Part 2: 20 Minutes

Playtime

Log-in	**1 min.**
To review the log-in instructions, please click here.	
Building time	**19 mins.**
Students build challenges using Challenge Builder. They can build as many challenges as they want, and they can share them with the classroom. Encourage them to play challenges built by other students if they are done.	

Part 3: 5 Minutes

Debriefing

Discussion	**5 mins.**
• Did you enjoy building your own challenges? • How many challenges did you build? • Did you play a challenge created by another student? • Did you solve a challenge created by another student? • Now that you built your first challenge, what more do you want to build?	
Optional	
Show a challenge created by a student in your classroom and try to solve it with everyone.	

Lesson 16 – Fundamental Stars Party!

In this lesson, your students will revisit challenges they have already solved but received only one or two stars for their solution. By the end of this lesson, all of your students should have perfect three-star scores on every challenge in Coding Adventure Part 1: Fundamentals.

Objectives

In this lesson, students will:

- Revisit challenges where they received one or two stars
- Solve all challenges with three stars

U.S. Standards Addressed

CSTA-K12 Computer Science Standards	
• 1B-AP-9 • 1B-AP-10 • 1B-AP-11 • 1B-AP-12 • 1B-AP-13 • 1B-AP-15	• 2-AP-10 • 2-AP-11 • 2-AP-12 • 2-AP-13 • 2-AP-14 • 2-AP-16 • 2-AP-17

Part 1: 20 Minutes

Introduction

Prior to class: Check your teacher dashboard for challenges where a high number of students struggled to get three stars.

Walk-through	**15 mins.**
Choose two or three challenges that got a relatively high number of blue or red stars, and solve them together in class with your students.	

Part 2: 25 Minutes

Playtime

Playtime	**25 mins**
Once they are logged in, instruct them to click the map (upper-right corner) and find challenges where they got one or two stars. At the end of class, all students should have three stars in all the first 75 CodeMonkey challenges. Students who completed all the first 75 challenges with 3 stars should proceed to complete unlocked skill challenges, or get 3 stars in skill challenges they already solved. If any of your students finished all 75 challenges and all skill challenges with 3 stars, ask them to help their fellow classmates.	

Part 2
Quizzes

Quizzes	
There are three quizzes in this part: 1. Part 1: Sequencing, Objects & Times Loops – 5 challenges 2. Part 1: Variables & Indexes – 5 challenges 3. Part 1: For Loops – 4 challenges You can assign quizzes to your class from the Quizzes tab on your teacher dashboard.	

Part 3: 5 Minutes

Debriefing

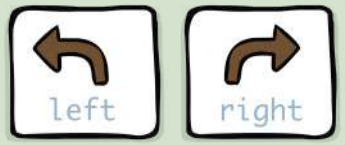
Explanation	**5 mins.**

Discuss briefly with your students the importance of writing short code.

In Coding Adventure, when we get two stars, it means there is a shorter way to reach the same end result. Either we have lines of code that are unnecessary for reaching the end result, or there is a shorter solution, like using a loop.

Imagine that every time you wanted to go to your next class, you had to first go home and then come back and go to the classroom. It does not make any sense to do that. Writing long code is the same. If there is a shorter way to do the same thing, it does not make any sense to do it the long way.

Reference Card

Keyword/Button	Description
	To make the monkey “step” a certain distance, we have to write “step X” using the number of steps we want the monkey to take. For example, if we want the monkey to walk 10 steps, we will write “step 10”. Pressing the step button at the bottom of the editor will enter “step” into your code.
	“Turn” should be accompanied by a direction (left/right) or degrees (45°, 90°, 180°). Examples: “turn right”, “turn 90” Pressing the turn button will write the word “turn” in your code.
	“Left” and “right” are used after the statement “turn” to make the monkey turn in the desired direction. Pressing the left or right buttons will write the word “left” or “right” in your code accordingly.
	“turnTo” is another way of turning. Instead of using direction or degrees, we are asking the monkey to turn to a specific object, for example, “turnTo banana”. Pressing the turnTo button will write the word “turnTo” in your code.

Reference Card Cont.

Keyword/Button	Description
	A simple loop is a sequence of instructions that repeats a specified number of times. For example: 3.times -> step 5 turn left In this example, the monkey will repeat “step 5, turn left” three times. The instructions we write in the loop should be written underneath it with an indentation (....). You can do that by pressing the Tab key on the keyboard. Pressing the times button will write the beginning of a simple loop in your code: “3.times - >”.
x = 10 step x	Assignments to variables. A variable is like a storage unit: we store data in it, and we use it only when we need it. An assignment to a variable is constructed out of an identifier and a value. This separation of name and value allows the name to be used independently of the information it represents. We can use X when writing the program, without knowing what its value will be when the instructions will be carried out.

Reference Card Cont.

Keyword/Button	Description
	"say" will make a speech bubble appear next to the monkey with the text we entered, for example: will make the monkey say "Boo!" We use quotation marks (" ") around the phrase we want the monkey to say in order for the computer to understand that the text we entered is not a variable. Try using "say" when there is a rat around. Pressing the say button will write the word "say" in your code.
	"distanceTo" is used with another statement like "step" or "say" and an object. Using "distanceTo" is like asking a question, for example, "What is the distance to the banana?" The answer is a number, calculated by the computer, that represents the distance. For example: step distanceTo banana The computer will measure the distance between the monkey and the object (banana). Then it will use the resulting number to carry out as instructed, using the measured value as the argument for "step". Pressing the distanceTo button will write the word "distanceTo" in your code.

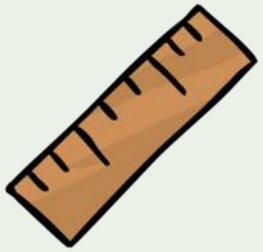
Reference Card Cont.

Keyword/Button	Description

	This is a “for” loop. A “for” loop is used when we have a collection of objects and we want to repeat an action that relates to each one of them specifically. The “for” loop will keep going until all the actions are done on all the objects in our collection (array). When the computer executes this loop, it replaces the variable name with the first item in the collection. After it is done with the first item, it moves on to the second, and so on. We can also use a “for” loop inside of a “for” loop; the example on the below is taken from challenge #75. Pressing the for button will write the following text in your code.
	Note the comment line –
	`for name in array`
	`  # Your code here`
	Example:
	`for b in bananas`
	`  for c in crocodiles`
	`  c.turnTo b`
	`  turnTo b`
	`  step distanceTo b`

	Pressing the run button will make the code on the right run. You can see the outcome by looking at the scene on the left.

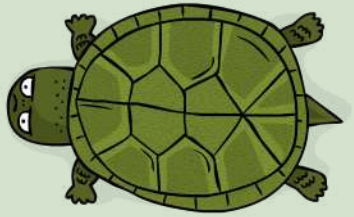
Reference Card Cont.

Keyword/Button	Description
	The reset button will erase everything you wrote in the code on the right and will reset the code to how it was at the beginning of the challenge.
	The ruler is a tool to help you measure the distance between different objects in the game, for example, the distance between the monkey and the banana. The ruler can also help you measure angles that the monkey or turtle has to turn in order to face another object on the screen, like a banana. To use the ruler, click it once, and then use your mouse to move the ruler to the point you want it to start measuring from. Click the mouse again, and then drag it to the end point. A number will appear at the end point to indicate the distance. Use this number with the “step” statement. Another number will appear near the starting point, this number indicates the angle from the first object to the second one, use it with the “turn” statement.

Reference Card Cont.

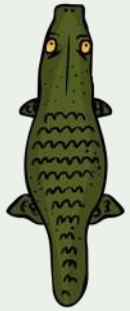
Keyword/Button	Description
	After each challenge, you'll receive a star-rating for your solution. The stars are distributed as so: ● First star is given if you got all bananas ● Second star is given if you used what you learned ● Third star is given if your code is short and to the point

Character Review

Character	Description
	Gordo, named after the first ape in space, is the guide who will help you and give you instructions along the way. His remarks are both funny and helpful. You can always click him to re-read the instructions.
	The monkey is the main character. You will have to help him collect bananas by writing lines of code. Just so you know, monkeys don't like to get wet, and they are very friendly.
	In challenge #13, you will meet our trusty turtle. The turtle will help you get those sneaky bananas. In order to instruct the turtle to "turn" or "step", we have to click it first. This will write its name in the code, and then separate it from the action we want it to take using a dot (.). For example: `turtle.step 10`

Character Review Cont.

Character	Description
	In challenge 55 you will meet the beavers. The Beavers like wood very much, and they have agreed to help you cross the water and get more bananas by stepping on their wood. They can only “step” forward and backwards. To use them we need to use a dot (.) between their name and the action we want them to take. For example: `beavers[0].step 10`

	Crocodiles are introduced in challenge #71. They are used to form a bridge on the water, to help the CodeMonkey reach bananas. They can only “turn” or “turnTo”. We usually use crocodiles with “for” loops.
	For example:
	`for c in crocodiles`
	`  c.turn right`

Great Job!

You have Completed

CODING ADVENTURE

FUNDAMENTALS