

Unit 4 - Variables, Conditionals, and Functions

Unit Overview

Students expand the types of apps they can create as they learn how to store information (variables), make decisions (conditionals), and better organize code (functions). Each programming topic is covered in a specific sequence of lessons that ask students to 'Explore' ideas through hands-on activities, 'Investigate' these ideas through guided code reading, 'Practice' with sample problems, and apply their understanding as they 'Make' a one-day scoped project. The entire unit concludes with a three-day open-ended project in which students must build an app that makes a recommendation about any topic they wish.

Unit Philosophy and Pedagogy

- **Intro to EIPM:** This unit is students' first experience with the Explore, Investigate, Practice, Make lesson sequence, or EIPM. This structured approach to teaching programming is covered in detail in the curriculum guide and we highly recommend that you watch the accompanying video series to better understand what EIPM should look like in the classroom. When used effectively, it supports deep learning of content and helps maintain a collaborative classroom culture, even as you move into more complex programming concepts.
- **Scaffolding Towards Independent Projects:** A major goal of this course is to empower students to design and build projects independently. The Create PT in Unit 8 offers students enormous freedoms to scope and build projects, and even this unit begins scaffolding towards that goal. Individual EIPM sequences of lessons gradually prepare students for scoped, independent Make projects. The unit project has a few requirements, but students are largely free to choose the design, topic, and implementation of their ideas. As you teach the unit, look for the opportunities to scaffold the skills and knowledge students will need to creatively and independently tackle the unit project.

Major Assessment and Projects

The unit project asks students to design an app that makes a recommendation based on input information from the user. Students are given a great deal of freedom to choose their topic, design their user interface, and decide how to actually program their app's behavior. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process they used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

AP Connections

This unit and unit project helps build towards the enduring understandings listed below. For a detailed mapping of units to Learning Objectives and EKs please see the "Standards" page for this unit.

- CRD-2: Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- AAP-1: To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow

programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

This unit includes content from the following topics from the AP CS Principles Framework. For more detailed information on topic coverage in the course review [Code.org CSP Topic Coverage](#).

- 1.4 Identifying and Correcting Errors
- 3.1 Variables and Assignment
- 3.3 Mathematical Expressions
- 3.5 Boolean Expressions
- 3.6 Conditionals
- 3.7 Nested Conditionals
- 3.15 Random Values

The College Board has supplied formative Create PT questions to help prepare students to complete the Create Task. We recommend that students complete the following prompts with the unit project. More information can be found in [Code.org CS Principles Topic Coverage](#).

- 3.a.i.
- 3.a.ii.
- 3.a.iii

Week 1

Lesson 1: Variables Explore

Develop a mental model for how information is stored and processed by programs.

Lesson 2: Variables Investigate

App Lab

Investigate and modify sample apps that use variables and learn common programming patterns with variables.

Lesson 3: Variables Practice

App Lab

Practice programming with variables through a set of programming puzzles.

Lesson 4: Variables Make

App Lab

Practice making an app that uses variables and programming patterns with variables.

Lesson 5: Conditionals Explore

Develop a mental model for how computers make decisions.

Week 2

Lesson 6: Conditionals Investigate

App Lab

Investigate and modify sample apps that use conditionals and learn common programming patterns with conditionals.

Lesson 7: Conditionals Practice

App Lab

Practice programming with conditionals through a set of programming puzzles.

Lesson 8: Conditionals Make

App Lab

Practice making an app that uses conditionals and programming patterns with conditionals.

Lesson 9: Functions Explore / Investigate

App Lab

Develop a mental model for using functions to replace repeated code. Investigate and modify sample apps that use functions.

Lesson 10: Functions Practice

App Lab

Practice programming with functions through a set of programming puzzles.

Week 3

Lesson 11: Functions Make

App Lab

Practice making an app that uses functions and programming patterns with functions.

Lesson 12: Project - Decision Maker App Part 1

Project

Create an app from scratch that uses variables, conditionals, and functions to help a user make a decision.

Lesson 13: Project - Decision Maker App Part 2

Project | App Lab

Create an app from scratch that uses variables, conditionals, and functions to help a user make a decision.

Lesson 14: Project - Decision Maker App Part 3

Project | App Lab

Create an app from scratch that uses variables, conditionals, and functions to help a user make a decision.

Lesson 15: Assessment Day

Project

Assessment day to conclude the unit.



This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 1: Variables Explore

Overview

To begin the lesson students explore sample apps similar to the ones they'll be able to build by the end of the unit. Then students complete an unplugged activity with plastic baggies and sticky notes to build a mental model of how variables are used to move and store information. The lesson ends with a synthesizing discussion and students adding key vocabulary to their journal.

Purpose

The warm up activity is designed to provide context for the coming unit and motivate the reasons students will want to learn the concepts covered in Unit 4. The subsequent activity provides students a physical mental model they will be able to use when they start programming with variables in the subsequent lessons.

Agenda

Warm Up (5 mins)

Explore Lessons:
Explore Sample Apps
Preview the Unit

Activity (30 mins)

Variables

Wrap Up (10 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe variables, expressions, and variable assignment.
- Evaluate expressions that include numbers, strings, and arithmetic operators.
- Trace simple programs that use variables, expressions, and variable assignment.

Preparation

- Collect for each pair of students:
 - 3 sandwich baggies
 - packs of red and yellow stickies
 - pens / pencils
 - 1 dry erase marker per four students (pairs can share)
- Review the sample apps shown in the warm up
- Review the rules and vocabulary used in the slides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation
- **EIPM: A Short Introduction** - Resource

Make a Copy ▾

Teaching Guide

Warm Up (5 mins)

Explore Sample Apps

📍 **Group:** Place students in pairs

📖 **Do This:** Give students 3-5 minutes to explore sample apps that they'll find on Code.org at the beginning of this lesson (in Levels 2-4). Students can explore one or more depending on time.

🗣️ **Prompt:** These are samples of the kinds of apps you'll be able to build by the end of this unit. As you go through them, write down at least two examples where the app seems to be keeping track of a piece of information or using it to make decisions.

Preview the Unit

🗣️ **Remarks**

Last unit we built apps that mostly focus on input and output. For example

- When button clicked → show picture
- When button clicked → change screen
- When button clicked → play a sound

But we all want our programs to do more than that! In this unit we're going to learn how to build apps that keep track of information and use to make decisions and perform calculations. This is going to let us build much more powerful apps! So let's get started by diving in deep on what's involved with programming with information.

Activity (30 mins)

Variables

Group: Group students in pairs.

📍 **Distribute:** Give each pair of students

- A small stack of red and yellow sticky notes
- A pen / pencil
- 3 plastic baggies
- A dry erase marker to share with another group

📖 **Display:** Use the activity slides for this lesson to guide the unplugged activity on Variables.

💡 Teaching Tip

This is the first official "Explore" lesson in the EIPM model. Review the EIPM model in the **EIPM: A Short Introduction - Resource**.

Explore Lessons:

Overview: Students explore the new concept through a teacher-led hands-on group activity.

- Typically uses physical manipulatives
- Teacher leads with support of slides and activity guides

Goal: Students begin to develop a shared mental model and understand the main ideas of the new concept.

Explore



🗣️ Discussion Goal

Goal: Have volunteers share what they noticed with the room. Some ideas include:

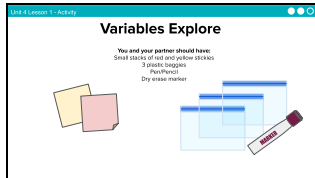
- The Pet Rock App keeps track of clicks and uses it to decide when the pet rock will "evolve"
- The Poem App keeps track of the poem as you write it
- The Thermostat App keeps track of temperature and uses it to change the color of the text

Slides

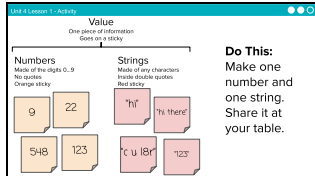
Speaker Notes

Slides

Speaker Notes

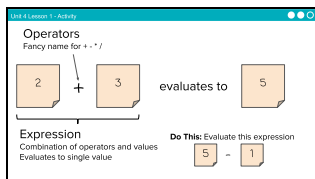


Say: Today we are going to explore Variables.



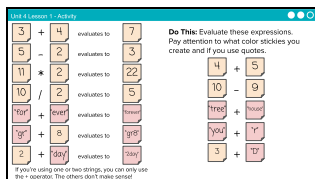
Say: We're going to be thinking about how computers work with information. We're going to call one "piece" of information a "value". Right now there are two different types of values: numbers and strings. There are a few ways to tell them apart. Numbers work the way you normally think of numbers. They are made of the digits 0 through 9. We are going to put numbers on yellow sticky notes. When we write numbers you don't need quotes. Strings are made of any characters you can see on the keyboard which means all these examples count as strings. Strings go inside double quotes. You can see how it would be important to tell the number 123 apart from the string "123".

Do This: Make one string and one number and hold it up at your table.



Say: Operator is just a fancy name for the plus, minus, multiply, and divide symbols. An expression is a combination of two sticky notes and an operator. When I ask you "what's 2 plus 3" you say "5"! You just "evaluated" the expression 2+3, or figure out what the value is. Since 5 is a number, we put it on a yellow sticky. When evaluating an expression, we follow order of operations - just like in math class.

Do This: Evaluate the expression 5-1 and make a sticky note. Make sure you pick the correct color and decide if you need quotes or not!



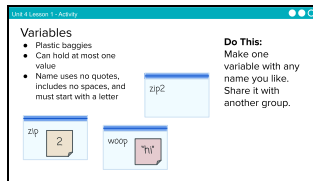
Say: This table shows the different ways we can create expressions. We can use all four operators with numbers the normal way. When you want to use strings, you can only use the + operator and it connects the two words together. If you're connecting a number and a string, the number first gets converted to a string.

Do This: With your partner make a sticky note for each of these 5 expressions and we'll share as a class.

 **Click for animation:** Click through to see the answers.

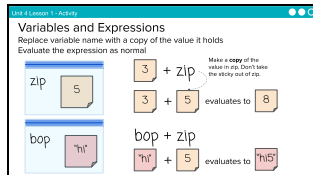
Slides

Speaker Notes




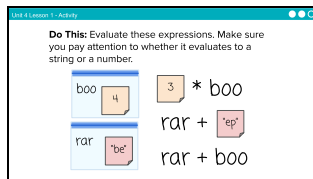
Say: We're going to call the plastic baggies on your table "variables". Variables can hold at most one value, or sticky note. They have names that use no quotes, include no spaces, and must start with a letter. For now just practice making a variable with your partner.

Do This: Make one variable with any name you like. Share it with another group. Make sure you use a whiteboard marker so we can reuse the baggies later.



Say: We can evaluate expressions that include variable names. To do that, first make a copy of the sticky note inside the variable. Then evaluate the expression the way you normally would. These two examples show you how.

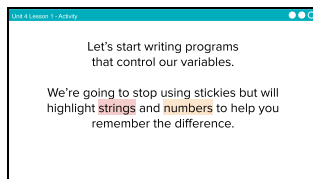
 **Click for animation:** Click through to see the answers.



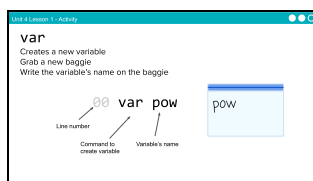
Do This: Evaluate these expressions. Make sure to pay attention to whether it evaluates to a string or a number.

Note: Have students share their answers by holding up the sticky notes for each solution.

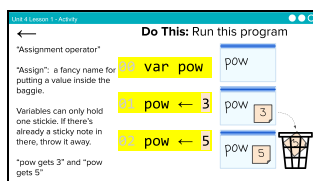
 **Click for animation:** Click through to see the answers.




Say: Now let's write programs. We're going to stop using sticky notes, but will highlight strings and numbers to help you remember the difference.



Say: Here's what a program looks like. The var command tells the computer to create a variable.



Say: The left arrow is called the "assignment operator". That's just a fancy word for "put this value in the baggie". If we wanted to read line 01 we would say "pow gets 3". We know that variables can only hold one sticky note or value. So if we try to assign a variable that already has a value in it, we just throw the old one away.

 **Click for animation:** Click through to run the program.

Say: In a computer you don't actually put the old number in a trash can, it's totally deleted! The numbers are stored as electrical charges somewhere in your computer's memory. When you assign a new number to a variable it actually erases the old number, and stores the new number in its place.


Slides

Speaker Notes

```
Unit 4 Lesson 1 - Activity
Do This:
Run this program. Compare your result with another group.
00 var pizza
01 pizza ← 3
02 var tacos
03 pizza ← "yum"
04 tacos ← "the best"
```

Do This: Run this program. Compare your results with another group.

Note: Walk around the room making sure students are following the rules correctly. No baggies should have more than one sticky note in them.

 **Click for animation:** Click through to see the answers.


```
Unit 4 Lesson 1 - Activity
Assign a Variable with Expression
Evaluate the expression first to get one value.
Assign the value as normal.
00 var pow
01 pow ← 1 + 2
02 pow ← 3 + 4
```

Say: Now let's combine what we've learned. You can use assignment with expressions. In order for this to work you need to evaluate first, then assign. This makes sense because we know we can only put one sticky note in the variable baggy.

```
Unit 4 Lesson 1 - Activity
Do This:
Run this program. Compare your result with another group.
00 var zow
01 var fly
02 fly ← "to" + "day"
03 zow ← 4 - 1
04 fly ← 3 * 3
05 zow ← 4 + "now"
```

Do This: Run the program. Compare your results with another group.

Note: Walk around the room making sure students are evaluating before they assign. Reinforce the language "gets" for the assignment operator. For example, line 02 would read "fly gets two plus day".

 **Click for animation:** Click through to see the answers.

We're not going to highlight our strings and numbers anymore. We can just use double quotes around the strings to tell the difference.

Say: We're not going to highlight our strings and numbers anymore. We can just use double quotes around the strings to tell the difference.

```
Unit 4 Lesson 1 - Activity
Assign a Variable: Expressions with Variables
Evaluate the expression on the right first to get one value.
Assign the value as normal.
var kit
kit ← 1
var boo
boo ← kit + 1
kit ← 5
```


Do This: Run through this program together as a class.

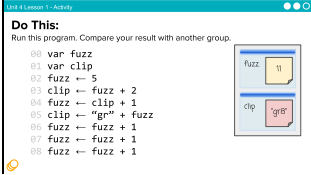

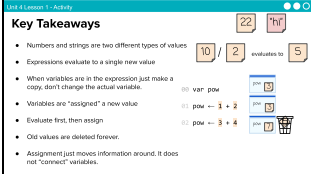
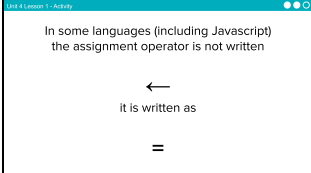
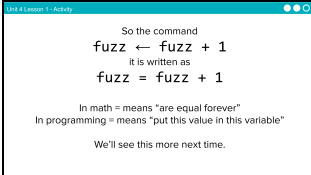
Note: Reinforce:

Evaluate, then assign


As shown on line 04, there's no special "connection" made between variables. All we're doing is moving information around.

Variables only hold one value, the old one is "thrown away" (again, what's happening is that assignment replaces the electric charges somewhere inside the computer so the old value is actually "erased" or "deleted")

 **Click for animation:** Click through to run the program.

Slides	Speaker Notes
	<p>Do This: Run this program. Compare your results with another group.</p> <p>Note: Have students hold up their two baggies when they're done. Make sure students are evaluating and then assigning. Call out that last three lines which can be a little nutty to think about. You're using a variable's value as part of the assignment. This lets it "count up by one".</p> <p> Click for animation: Click through to see the answers.</p>
	<p>Do This: Review the key takeaways with students.</p>
	<p>Say: In some programming languages the assignment operator is not written with an arrow but is written as an equal sign.</p>
	<p>Say: Here's how the assignment operator looks in Javascript. We'll see more of this in the next lesson!</p> <p>Note: In math = means "are equal forever". In programming = means "put this value in this variable".</p>

Wrap Up (10 mins)

 **Journal:** Have students add the following words to their journals: Expression, Variable, Assignment Operator.

- **Expression:** a combination of operators and values that evaluates to a single value.
- **Variable** an abstraction inside the program that can hold a value. Each variable has associated data storage that represents one value at a time.
- **Assignment operator:** allows a program to change the value represented by a variable.

Remarks

This isn't the last time we'll look at these definitions and it's fine to update them as you get more experience. In fact, I'm sure you'll have a lot more to add once we see all of these concepts used inside of apps when we meet next time!

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Teaching Tip

Supplies Substitutions: There's no need to use stickie notes if you have other scraps of colored paper. Also consider cutting stickies in 4 to make them go further. If you don't have dry erase markers handy consider using pieces of masking tape on the baggies.

Question: What will the value of score be at the end of the program?

```
var score
score <- 3
score <- score + 1
score <- "The score is: " + score
```

Teaching Tip

Running the Activity: This activity asks students to follow along as a number of core concepts for programming are introduced. The model is typically that a term or concept is introduced and modeled and then afterwards students are encouraged to try it out on their own. Trying it out typically means they are writing information on a sticky note and sharing it with another group before discussing the results with the whole class.

Slides with animations have an icon in the bottom left corner to let you know you need to click to reveal more of the slide's content.

To help you more easily prepare the activity and keep track of your instructions, detailed instructions have been included as speaker notes in the presentation. Here are some tips to help you throughout the presentation.

- There are opportunities throughout the presentation for students to actively engage. At these moments students should be making things with their manipulatives or using them to answer questions. Use these opportunities to check progress.
- There is a fair amount of new vocabulary introduced but it is introduced gradually and with intentional repetition. Make a point of actively modeling the use of new terms.
- The most important goal here is building a mental model. It is ok if students have some open questions that will get resolved over the subsequent conditional lessons.
- Both you and students can use the "Key Takeaways" to check your understanding at the end.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways
- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **DAT-1** - The way that the computer represents data is different from the way that the data are interpreted and displayed for the user



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 2: Variables Investigate

Overview

In this lesson students work with partners to investigate several versions of the "Thermostat App" to understand how variables store and update information. To begin, students examine a version of the app where the temperature displayed changes each time a button is clicked. The next two versions of the app demonstrate how variables can store strings. Students learn about the patterns they are observing, specifically "Counter Pattern with Event" and "Variables with String Concatenation Pattern". To conclude the lesson, students review and discuss the programming patterns that they will make use of in the programs they write.

Purpose

After building a conceptual model for variables in the previous lesson, students investigate three working examples of apps that make use of variables. This lesson also introduces common programming patterns when using variables. Students will have some opportunities to modify working code in this lesson, but the most significant practice with variables will come in the following lesson.

Agenda

Warm Up (5 mins)

Investigate Lessons:
Preview the Lesson

Activity (35 mins)

Investigation #1: Thermostat App v.1 (Levels 2 - 3) - 10 mins

Investigation #2: Thermostat App v.2 (Level 4) - 10 mins

Investigation #3: Thermostat App v.3 (Levels 5 - 6) - 8 mins

Patterns - 7 mins

Wrap Up (5 mins)

Assessment: Check For Understanding

[View on Code Studio](#)

Objectives

Students will be able to:

- Identify common programming patterns when using variables as part of an app
- Explain the purpose of those programming patterns with variables both in terms of how they work and what they accomplish
- Modify apps that make use of common programming patterns with variables to adjust their functionality

Preparation

- Review the example apps and the prompts that students will be asked to respond to for each
- Review the information covered in the slides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **EIPM: A Short Introduction** - Resource [Make a Copy](#)
- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation


Teaching Guide

Warm Up (5 mins)


Preview the Lesson

Remarks


Yesterday we explored storing information like a computer. Computers store each piece of information in a variable. In Javascript, we name or declare a variable with the keyword `var`. Today we are going to look at a new app that stores information in variables.

 **Prompt:** Let's do a quick review. How does a baggy represent a variable?

Activity (35 mins)


 **Group:** Place students in pairs. One student per group should navigate to the lesson on Code Studio.


Investigation #1: Thermostat App v.1 (Levels 2 - 3) - 10 mins


 **Level 2: Thermostat App v.1:** This level introduces a new app for students to investigate. It represents a Thermostat App where the temperature can be changed up and down.

- Run the app: Let students run the app for a few minutes.
- Predict: Students predict the information that is being stored in variables.


 **Level 3: Thermostat App v.1 Code:**

- Assign Code Sections: Assign half the pairs to investigate the first section on lines 1-12. The other half investigates the second half on lines 14-21.
- Read Code: Groups should carefully read the code for their section making sure they understand how it works. Give them 3 minutes to do so.
- Explain Your Section: Have partners make a group with members of the other section and carefully explain how their section works line by line.
-  Class Discussion: Ask a few members of each section to quickly share out how their section works. Display the code at the front so you can talk through it together.

 **Display:** Display the slide showing students how to add a watcher in the debugging panel to track the value a variable stores.

 **Modify:** have groups return to their original seats. Give them a couple of minutes to work on modifying the app to change the degrees by two when the up and down arrows are clicked.

Investigation #2: Thermostat App v.2 (Level 4) - 10 mins

 **Level 4:** This program is an updated version of the Thermostat app. This time students should continue to work in partners but do not need to work with other groups. They will need to:

Teaching Tip

This is the first official "Investigate" lesson in the EIPM model. Review the EIPM model in the **EIPM: A Short Introduction - Resource**.

Investigate Lessons:

Overview: Students investigate two or three sample programs that use the new concept.

- Close-reading of working programs
- Teacher-led discussions
- Tasks to modify apps

Goal: Students become comfortable reading and modifying programs that use the new concept.

Investigate



Discussion Goal

The baggy represents a variable by storing one item: a value on a sticky note placed in a named baggy. The value in the baggy can be changed at any time, just like a variable's value can change.

- Run the app: Let students run the app for a few minutes.
- Discuss Changes: Talk through how the app is different than the first version.
- Find the `Math.round` command: Discuss with a partner how this command might work. Try deleting `Math.round` in lines 3, 15, and 28. What happens? Add it back in.
- 🗣️ Class Discussion: Ask a few students to explain what's happening on line 3 with `Math.round`.
- Modify the Code: Change the code so that no space displays between the temperature and the unit description ("F" or "C").

Investigation #3: Thermostat App v.3 (Levels 5 - 6) - 8 mins

📖 **Level 5:** This program is again an updated version of the Thermostat app with a login screen.

- Run the app: Let students run the app for a few minutes.
- Predict: Students predict the information that is being stored in variables.

📖 **Level 6:** The code for Thermostat App v.3 is displayed.

- Read Code: Give students a few minutes to read and discuss the code with their partners. What has changed? What has stayed the same?
- 🗣️ Class Discussion: Look at line 40. How does `getText()` work? What is it doing?
- Modify the Code: Add an explanation point "!" to the end of the string stored in `userName`.

Patterns - 7 mins

🗣️ **Remarks**

We are learning two patterns here:

- Counter Pattern with Event
- Variables with String Concatenation Pattern

The Counter Pattern with Event is a common pattern for updating variables that you will use in making many different apps. We call it the **Counter Pattern with Event**.

📖 **Display:** Talk through the pattern with students.

🗣️ **Prompt:** With a partner, discuss the following:

- The "Counter Pattern with Event" should look familiar! How would you explain this pattern to another person?
- When might you want to use the "Counter Pattern with Event"?

🗣️ **Remarks**

Variables can store many different types of information including numbers and Strings. Anything placed inside of the quotation marks becomes a String.

🗣️ Discussion Goal

Here are some points that students are likely to bring up while discussing their code:

Lines 1-12:

- Two variables are created: `temp` and `tempF`. `temp` gets the value 70 and `TempF` is set to an empty string, which is represented with empty quotation marks.
- On line 7, an `onEvent` is created for when the `downButton` is clicked.
- When the down button is clicked, the `temp` variable decreases by one.
- Then the `tempF` variable is updated with the current value of `temp` joined with the letter `F`.
- The text on the screen is set to display the value stored in the variable `tempF`
- A sound is played.

Lines 14-21

- On line 16, an `onEvent` is created for when the `upButton` is clicked.
- When the up button is clicked, the `temp` variable is increased by one.
- Then the `tempF` variable is updated with the current value of `temp` joined with the letter `F`.
- The text on the screen is set to display the value stored in the variable `tempF`
- A sound is played


🗣️ Discussion Goal

On line 3, the Fahrenheit temperature is converted to Celsius. `Math.round` rounds a given value to the nearest integer.

🗣️ Discussion Goal

`getText` gets the string of text that is found in the element with the given id. In this case, the id is `"nameInput"`. In our app, the name written in the `"nameInput"` element is concatenated with `"Hi"` and then stored in the variable `myName`.

Guess what! There's a pattern for working with Strings. When we want to combine Strings with other Strings, numbers, or even with another variable we call that concatenation.


 **Display:** Talk through the pattern with students.


Prompt: Explain to a partner how the "Variable with String Concatenation Pattern" works.

Wrap Up (5 mins)

Remarks

The patterns we learned today are present in many different kinds of apps. As we learn new patterns, they will show up in the "Help & Tips" tab at the top of your screen on programming levels.

 **Prompt:** Let's review. What can be stored in a variable? Why is using a meaningful name for the variable important?

 **Journal:** Add to your journal the definition for variable.

Remarks

Today we've learned a lot about how variables work in actual programs.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Explain in your own words the process of creating and updating a variable. How does the Counter Pattern with Event work?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways
- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result

Discussion Goal

Students should step through each line of the pattern, explaining what's happening.

Counter Pattern with Event:

- myVar gets the value 0
- When the event is triggered, myVar is updated getting the current value of myVar and adding 1.

The **Counter Pattern with Event** might be used to update a score in a game when an item is clicked.

Discussion Goal

Students should be able to verbally explain how the **Variable with String Concatenation Pattern** works.

- myString gets the value "rock"
- myOtherStrings gets the value "roll"
- myStory gets the value that is stored in myString ("rock") and combines that with a string "and" and the value stored in myOtherString ("roll")
- myStory now stores the value: "rock and roll"

Discussion Goal

Answers: What can be stored in a variable? * At this point, students should know that these two things can be stored in variables. Students will learn about other data types that can be stored in variables in later lessons. * Numbers * Strings * Including concatenated strings (temp + " F")

Why is using a meaningful name for the variable important? *When you write code, you are not just writing for the computer, you are also writing for people. Using meaningful names help people be able to read your code easier.* It also helps readers predict what values may be stored in the variables.



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 3: Variables Practice

Overview

In this lesson students spend most of their time practicing using the skills and processes they have learned about variables. At the conclusion of the lesson students discuss the main things they realized and still have questions about at the conclusion of this lesson.

Purpose

This lesson is students primary opportunity to get hands on with variables in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing. In the following lesson students are encouraged to work independently.

Agenda

Warm Up (0 mins)

Practice Lessons:

Quick Warm Up

Activity (40 mins)

Practice Time

Wrap Up (5 mins)

Synthesizing Discussion

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Write programs that use variables and expressions with the support of sample code.
- Debug programs that use variables and expressions

Preparation

- ☐ Review the video in the level progression that covers global vs. local variables. This is a tricky topic.
- ☐ Review other programming levels to be better prepared to support students

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **EIPM: A Short Introduction** - Resource
[Make a Copy](#)
- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

Teaching Guide

Warm Up (0 mins)

Quick Warm Up

Remarks

🔗 Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!

Activity (40 mins)

Practice Time

Group: It is recommended that students work in pairs for this lesson and a number of the activities feature discussion prompts. Consider using pair programming, having drivers and navigators switch every 3 minutes, not every level.

Distribute: Optionally pass out a plastic cup or other manipulative they can place on their computer when they are stuck as a signal that they need support.

Remarks

Today you're mostly going to practice what we've learned about programming with variables. As always you should be using the debugging process to help you as you work on issues.

📖 Let's review the debugging process first.

Debugging is the process of finding and fixing problems in code. We previously talked about these four steps: Describe, Hunt, Try, and Document.

📖 Today we're going to also focus on three particular debugging skills

1. Slowing down code with the speed slider
2. Using console.log to get output
3. Using the Watch area to watch your variables change values

📖 🔗 **Do This:** Direct students to Code Studio, Lesson 3 Level 2.

Levels 2-3 Assigning Numbers and Strings: These levels only use the console.log() command which prints commands in the debug console. Some key points in these levels.

- Students should practice using the "Watch" area to track how variables change over a program
- Students are asked to discuss what they see with a partner
- These levels heavily reinforce language around assigning and creating variables

Teaching Tip

This is the first official "Practice" lesson in the EIPM model. Review the EIPM model in the **EIPM: A Short Introduction - Resource** .

Practice Lessons:

Overview: Students practice using the new concept through a scaffolded series of programming activities.

- Students work independently or in pairs
- Teacher introduces debugging practices that the beginning of the Activity and circulates the room during the lesson to provide targeted support.

Goal: Students gain confidence in writing and debugging programs that use the new concept.

Practice



Teaching Tip

Move Quickly to the Activity: There's a lot in the main activity of today's lesson. You may optionally wish to do a quick vocabulary review or address any questions that came up in the last lesson. Otherwise, give students more time to get hands on with some code.

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

- Level 3 is a debugging-heavy level which introduces more rules about syntax.

Levels 4-8 Variables and Operators: These levels have students practice writing more complex expressions and using operators. A few tricky things to look out for

- These levels transition from using only `console.log` to using full apps with user interfaces.
- Level 6 introduces `\n`, the new line character
- Students continue to be encouraged to discuss what they are learning with a neighbor
- Levels 7 and 8 introduce the use of the counter pattern with numbers and strings. Students may need to consult the Help and Tips tab for support.

🔍 **Regroup:** Bring the class back together to watch the Scope Practice Video.

📺 **Display:** Scope Practice video. This video is located on the slides and on Level 9. You may opt to have students watch it alone or as a class.

📺 **Do This:** Discuss debugging scope issues with the next three slides before directing students back to Code Studio.

📺 **Levels 9-10 Debugging Scope Issues:** These levels identify a common bug that can come up when working with variables. While students should be aware of this bug, they don't quite have the background they'd need to fully understand it. For now support students in following the three main takeaways.

- Create variables once
- Create variables at the top of your program
- Don't create variables inside `onEvent()` or `function()` blocks

📺 **Level 11 Putting It All Together:** This level asks students to put together many of the concepts they've seen so far. This is an opportunity to have a bit more of a "blank screen moment" while still being able to use some starter code as a guide.

Wrap Up (5 mins)

Synthesizing Discussion

🗨️ **Prompt:** What aspects of working with variables do you feel like clicked today? What do you still feel like you have trouble with?

Discuss: Have students share with one another before sharing with the whole class.

🎤 **Remarks**

Variables can be a little bit tricky, but I saw a lot of good progress today in nailing down this concept.

We may have a few lingering questions, but you also have a lot of resources available. Next time you'll have a chance to put all this together by programming an app that starts with "the blank screen"!

💡 Teaching Tip

Reviewing a Map Level: Level 9 is a Map Level, which contains a review of variable scope. It is highly recommended that you watch the video in Level 9 prior to the lesson. Slides are available for this lesson if you would like to optionally review as a class.

💬 Discussion Goal

Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What will be displayed after this code segment is run?

myPoints - 2

myPoints - 5

myPoints - myPoints + 1

DISPLAY myPoints

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-1** - To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways
- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces



This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 4: Variables Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Photo Liker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Warm Up (2 mins)

Make Lessons:
Intro the Project

Activity (38 mins)

Build the Photo Liker

Wrap Up (5 mins)

Assessment: Make Project

View on Code Studio

Objectives

Students will be able to:

- Recognize the need for programming patterns with variables as part of developing a functioning app
- Implement programming patterns with variables to develop a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Preparation

- Review the different steps students will be asked to complete as they build the app
- Review the information covered in the slides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP Variables Make - Photo Liker App** - Activity Guide [Make a Copy](#)

Teaching Guide

Warm Up (2 mins)



Intro the Project

Remarks

For the past few days, we've learned a lot about storing and updating information in variables. Today you are going to have an opportunity to demonstrate your learning by making an app.

Activity (38 mins)

Build the Photo Liker

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Do This: Have students explore the working Photo Maker App in Level 2.



Prompt: If students are not working in pairs they should still discuss the prompts with a neighbor.

- What does this app do?
- What are the inputs?
- What are the outputs?
- What's one piece of information that might be stored in a variable?



Remarks

Now let's build this app. The screen has been set up for you - it's your job to add the code!

Distribute: Give students copies of **CSP Variables Make - Photo Liker App - Activity Guide** if you will be using it during the class.

Do This: Direct students to level three where they complete the Photo Liker App. Given this is students first Make project, it is highly recommended that students practice with this guide the first time. In future Make lessons they may opt not to use this guide. For students who need more detailed guidance once they've started programming, Step 3 includes step by step instructions that the student can follow.

Based on the needs of your classroom decide whether you will collectively go through the activity guide or have students complete it individually.

Submit: Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Teaching Tip

This is the first official "Make" lesson in the EIPM model. Review the EIPM model in the **EIPM: A Short Introduction - Resource**.

Make Lessons:

Overview: Students make a target app for which they are given the screen elements but little to no starter code.

- Students are provided high-level steps to break down the project
- Teacher supports students by directing them towards notes, previous work, and debugging strategies practiced in earlier lessons.

Goal: Students are able to independently decide when and how to use the new concept in the context of a larger project.



Teaching Tip



Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Discussion Goal

- What does this app do?
- It lets a user "like" an image and add comments to it.
- What are the inputs?
- The thumbs up and thumbs down button, the text input for adding a new comment, and the button to add the comment.
- What are the outputs?
- The number showing how many thumbs up the image has.
- What's one piece of information that might be stored in a variable?
- The number of likes and all the comments added so far.

Wrap Up (5 mins)

Remarks

  Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Run the code on turtle mode
- Add the variables to the watcher
- Use `console.log()` to get output from the app
- Explain the code to a friend
- Read the code carefully line by line, trying to decide which one is causing the error.

Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 5: Conditionals Explore

Overview

Students learn the basics of conditionals through an unplugged activity using the sticky notes and plastic baggies from the Variables Explore lesson. Flowcharts are introduced as a way to understand how computers make decisions using Boolean expressions.

Purpose

The warm up activity is designed to provide context for the Conditionals progression. The subsequent activity provides students a physical mental model they will be able to use when they start programming with conditionals in the subsequent lessons.

Agenda

Warm Up (5 mins)

Preview Conditionals

Activity (30 mins)

Conditionals

Wrap Up (10 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Use appropriate vocabulary to describe Boolean expressions and conditional statements
- Evaluate expressions that include Boolean values, comparison operators, and logical operators
- Trace simple programs that use Boolean expressions and conditional statements

Preparation

- ▣ 3 sandwich baggies per pair of students
- ▣ packs of red, yellow, and blue stickies
- ▣ pens / pencils
- ▣ 1 dry erase marker per four students (pairs can share)
- ▣ Review the slides and click through all animations

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CS Principles: Conditionals - Part 1 Boolean Expressions** - Video

Teaching Guide

Warm Up (5 mins)

Preview Conditionals

Prompt: Imagine you want to make a decision about what to wear to an event. Name two pieces of information you'd want. How would you use them in your decision?

Activity (30 mins)

Conditionals

Group: Group students in pairs.

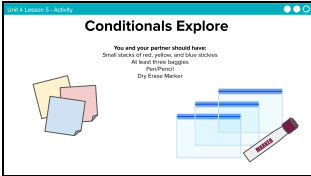
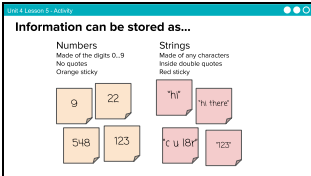
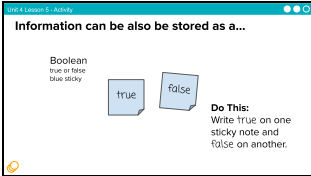
Distribute: Give each pair of students:

- A small stack of red, yellow, and blue sticky notes
- A pen/pencil
- 3 plastic baggies
- A dry erase marker to share with another group

Display: Use the activity slides for this lesson to guide the unplugged activity on Conditionals.

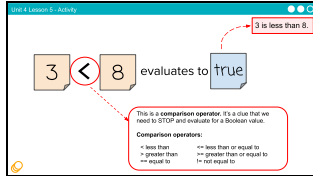
Teaching Tip

Supplies Substitutions: There's no need to use stickie notes if you have other scraps of colored paper. Also consider cutting stickies in 4 to make them go further. If you don't have dry erase markers handy consider using pieces of masking tape on the baggies.

Slides	Speaker Notes
	<p>Say: Today we are going to explore Conditionals.</p>
	<p>Say: Previously, we learned that information can be stored as numbers or strings in variables. These are two different types of data. Numbers and strings can be combined together using operators to make expressions. Expressions are evaluated before storing in variables.</p>
	<p>Say: Boolean values are another type of data. They can store the values “true” or “false”. Why would we want that information? Booleans are used to make decisions. If something is true, do this. If something is false, do that.</p> <p>Click for animation</p> <p>Do This: You only need two blue sticky notes today. Write down “true” on one and “false” on the other.</p>

Slides

Speaker Notes



Say: Let's look at this expression. Take a guess as to what Boolean value this evaluates to.

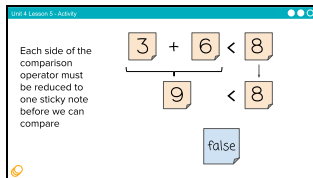
Do This: Hold up the sticky note with the correct value.

 **Click for animation**

Say: The expression evaluates to true, because 3 is less than 8.

 **Click for animation**

Say: The less than symbol is a comparison operator. When we see a comparison operator, we need to stop and evaluate for a Boolean value. There are six different comparison operators - you may be familiar with some of them from math class. Notice the double equal signs for "equal to". Here we are distinguishing from a single equal sign that is used to assign value to a variable. We read a single equal sign as "gets the value". A double equal sign can be read as "equal to".



Say: Both sides of the relational operator should be reduced to a single value before we can compare. Think of it in terms of sticky notes. You should have only one sticky note on each side.

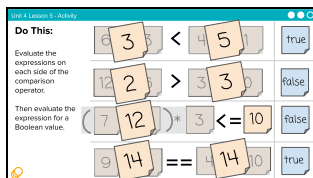
Do This: What does $3 + 9$ equate to?

 **Click for animation**

Say: Right: 9! Now we can compare 9 and 8 using the relational operator: less than.


Do This: Does this evaluate to true or false? Hold up the correct sticky note.

 **Click for animation**



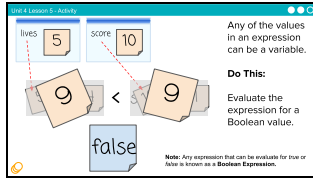
Say: Let's evaluate these expressions line by line. It may be helpful to use your sticky notes here.

Do This: Reduce each side of the relational operator to one sticky note. Then evaluate for a Boolean value. Compare your answers with a partner.

 **Click for animation:** When the class is finished, click through to view the answers. The third line may be a little tricky. Remind students to evaluate the information in the parenthesis first. In the fourth line, draw attention to the double equal signs and remind the class double equal signs are a signal to compare if both sides are the same.

Slides

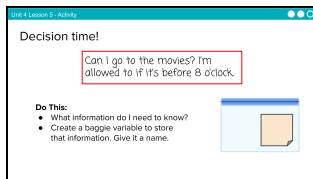
Speaker Notes



Say: The expressions that we evaluate can also contain variables. Remember, variables store information. This information can be different data types like numbers, strings, or even Boolean values themselves. For now, we are going to stick with numbers.

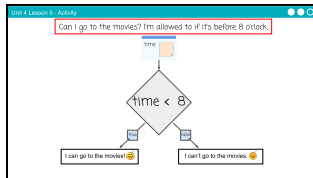
Do This: With a partner, reduce both sides of the relational operator. Then evaluate for a Boolean value.

Click for animation: When the class is finished, click through for the answer. Draw attention to the note: any expression that can be evaluated for true or false is known as a Boolean Expression.



Say: At the beginning of class, we talked about how Booleans are used to make decisions. Let's see that in action. Consider the following question: Can I go to the movies? I'm allowed to if it's before 8 o'clock. This may seem like a simple decision! But let's think about this like a computer.

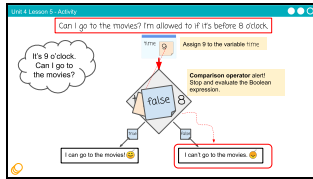
Do This: What information would the computer need to know in order to answer the question. Create a baggie variable to store the information. Give it a name.



Say: This is a flowchart. It's an organized way to make a decision or come to a conclusion. Creating a flowchart helps us think like a computer. At the top, we are listing the variables that are needed. Your variable might be called time or clock or really anything. It doesn't matter as long as you use the same name every time you refer to the variable. For this example, I'm naming my variable "time". Let's step through the flowchart with an example to see how it works.

Slides

Speaker Notes



👉 **Click for animation**

Say: It's 9 o'clock. Can I go to the movies?

👉 **Click for animation**

Say: First I assign 9 to the variable time, then I move on to the diamond section.

👉 **Click for animation**

Say: I see a comparison operator, which let's me know that we need to stop and evaluate the Boolean expression.

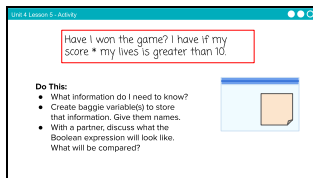
👉 **Click for animation**

Say: First reduce both sides to a single value. Then evaluate for a Boolean value. In this case, The Boolean expression evaluates to false.

👉 **Click for animation**

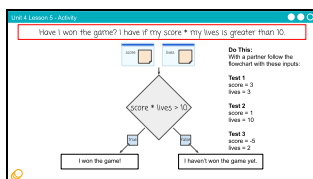
Say: Follow the "false" path, and we arrive at the result: "I can't go to the movies."

Note: If students are confused, return back to the previous slide and try running some examples together as a class. Students can hold up true or false when evaluating the Boolean expression. Once students get the hang of using a flowchart, continue on to the next slide.



Say: Here's another decision that needs to be made. I'm not sure if I have won the game I'm playing. I have won it if my score * (times) my lives is greater than 10.

Do This: Direct students to create baggie variables for the information that needs to be stored and write down the Boolean expression on a spare sticky note or scrap paper. Groups then compare their Boolean expressions.



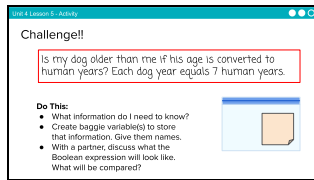
Say: Take a look at the Boolean expression in the decision section of the flowchart. Is this similar to what you and your partner wrote? Are there other ways we could construct this? (score and lives might have different names, score and lives might be in opposite order, you could write it as $10 < \text{score} * \text{lives}$).

👉 **Click for animation**

Do This: With a partner, try following the flowchart with a few different inputs.

Slides

Speaker Notes



Slide 4: Lesson 5 - Activity

Challenge!

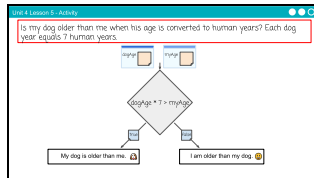
Is my dog older than me if his age is converted to human years? Each dog year equals 7 human years.

Do This:

- What information do I need to know?
- Create baggie variable(s) to store that information. Give them names.
- With a partner, discuss what the Boolean expression will look like. What will be compared?

Say: Here's a new decision we need to make.

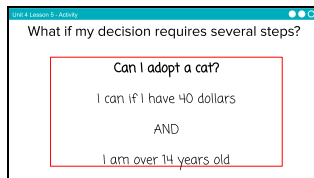
Do This: With a partner walk through the steps on the screen. Think about what the Boolean expression will look like.



Say: Here's a flowchart for the question we just considered.

Do This: With your partner, step through the flowchart with a few different inputs.

Note: After students have practice on their own, call on one group to give input values for `dogAge` and `myAge` and step through the flowchart as a class.



Slide 4: Lesson 5 - Activity

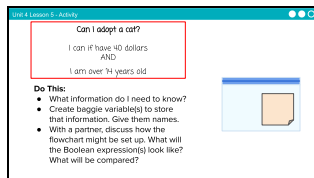
What if my decision requires several steps?

Can I adopt a cat?

I can if I have 40 dollars
AND
I am over 14 years old

Say: What happens if my decision requires several steps? I want to adopt a cat, but I have to have both 40 dollars and be over 14 years old.

Note: Emphasize precise language here. The Boolean is checking if I *have* 40 dollars - that's exactly 40, not one cent more! In the Conditional Practice lesson, students will return to this flowchart and edit it to account for having more than 40 dollars.



Slide 4: Lesson 5 - Activity

Can I adopt a cat?

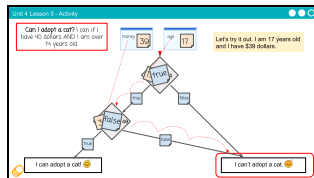
I can if I have 40 dollars
AND
I am over 14 years old

Do This:

- What information do I need to know?
- Create baggie variable(s) to store that information. Give them names.
- With a partner, discuss how the flowchart might be set up. What will the Boolean expression(s) look like? What will be compared?

Say: Let's get setup for stepping through the decision making process for this question.

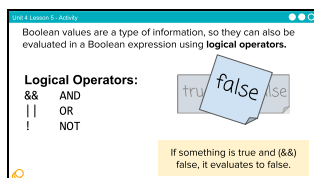
Do This: Set up your baggies, and discuss with a partner how the flowchart might be set up.



Say: This flowchart looks a little different from the previous one. There are two stopping points where a Boolean expression is evaluated. Let's try this out with an example.

Click for animation: Click through, one step at a time with the class. When you get to a decision point, before clicking to the next animation have students evaluate the expression and hold up a blue sticky note.

Say: Notice in question box, I've stated that I can adopt the cat if I have 40 dollars AND (emphasize) I am over 14 years old. I am checking two things. Let's look at a way to simplify that process.



Slide 4: Lesson 5 - Activity

Boolean values are a type of information, so they can also be evaluated in a Boolean expression using logical operators.

Logical Operators:

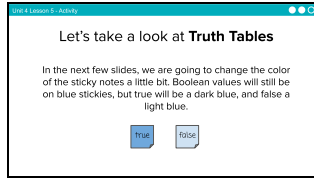
- && AND
- || OR
- ! NOT

If something is true and (&&) false, it evaluates to false.

Say: In the previous example, we had two Boolean expressions to evaluate. We can use logical operators to compare the results of those Boolean expressions. The logical operators we will look at are `&&` (AND),

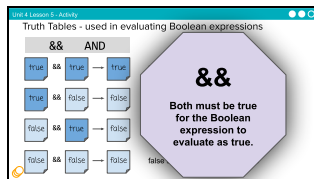
Slides

Speaker Notes



Say: Now we're going to take a look at something called a Truth Table. We are going to change the color of the sticky notes a little bit. Boolean values will still be on blue stickies, but *true* will be a dark blue, and *false* a light blue.

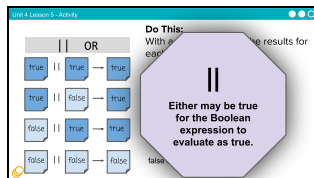
Note: there is no need to supply two different colors of blue sticky notes for these slides. Students can follow along.



Say: This is a Truth Table. It demonstrates all the possible combinations of true and false and what they would evaluate to if combined with the logical operator && (AND). The first one seems obvious to us - if something is true and another thing is true, then the whole thing is true!. Read through each row, and consider what is being stated. In rows 2 and 3, if one value is false, the whole thing evaluates to false. This is what happened with the cat example. On the final line, if both values are false, the final evaluation is false.

Click for animation

Say: What's the takeaway? In evaluating two Boolean expressions with the AND operator, both must be true in order for the whole expression to evaluate as true.



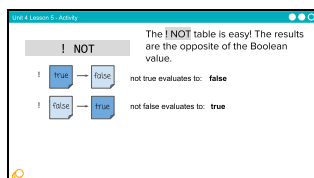
Say: Now let's consider a Truth Table for OR.

Do This: With a partner, evaluate each row. For example, if the first sticky is *True* or the second sticky is *True* what do you think the whole expression will evaluate to: true or false?

Click for animation: Stop line by line and allow time for groups to make their predictions.

Click for animation

Say: For the OR Truth Table, if either sticky note is *True* the expression evaluates to true.



Say: The ! NOT table is simple because it reverses the value.

Click for animation: As a class quickly state what the opposite values are and click through the animations.

Slides

Speaker Notes

Slide 4: Lesson 5 - Activity

Use logical operators to combine several Boolean expressions into one expression.

Can I adopt a cat?

I can if I have 40 dollars
AND
I am over 14 years old

Say: Let's return to the cat example. How can we write a single expression that accounts for all the information that needs to be checked?

Do This: With a partner, create the expression using logical operators.

Note: It's ok if students struggle with this activity. After a few minutes, move on to the next slide and reveal the answer. This will help students make the connection.

Slide 4: Lesson 5 - Activity

Can I adopt a cat? I can if I have 40 dollars AND I am over 14 years old. Let's try it out. I am 15 years old and I have \$40 dollars.

Truth Table

I have 40 dollars	I am over 14 years old	I can adopt a cat?
True	True	True
True	False	False
False	True	False
False	False	False

I can adopt a cat. I can't adopt a cat.

Click for animation: Click through the animation step by step and follow along as a class. Allow time to predict at decision points. When the truth table (yellow box) appears on the screen, take a moment to remind students what this is and what it demonstrates. If a value is true and another value is true, it evaluates to true. It's not expected that students will memorize the truth tables, but they should be familiar with how they work.

Note: If you have extra time, try running this flowchart with other inputs. Try changing the conditions and then running the flowchart (i.e. I can adopt the cat if I have 40 dollars OR I am over 14 years old)

Say: Today we used flowcharts to demonstrate how computers make decisions. In the next class, we will investigate how decisions are made in code.

Slide 4: Lesson 5 - Activity

Decision:

Do This: Now it's your turn to make a flowchart. On a piece of paper:

- Think back to the activity where we discussed making a decision on what to wear to an event.
- Write down the variables in boxes at the top.
- Make the diamond shape.
- Write the Boolean expression that will be used to make the decision.
- Draw two paths from the diamond to the possible outcomes.
- Challenge: Use logical operators (AND, OR, NOT) your Boolean expression. Add extra branches with multiple decisions.
- Test your flowchart with a friend!

Do This: Give students a few minutes to craft their own flowcharts and test on each other following the instructions on the screen.

Note: Simple flowcharts are fine, or students can build more complex models. Students are ready to move on when the majority of the class has made and tested a flowchart.

Slide 4: Lesson 5 - Activity

Key Takeaways

- Boolean expressions can also include **Logical Operators** `&&`, `||`, `!` (AND, OR, NOT). Both sides of the logical operator are reduced to a single Boolean value.
- A truth table is used to evaluate the reduced Boolean expression to a single Boolean value.
- A decision is made with the single Boolean value.
- A flowchart illustrates the steps of making a decision with a Boolean expression.

`true && false` evaluates to `false`

`true || false` evaluates to `true`

Click for animation: Review the key takeaways on the screen with students.

Slide 4: Lesson 5 - Activity

Key Takeaways

- Boolean expressions can also include **Logical Operators** `&&`, `||`, `!` (AND, OR, NOT). Both sides of the logical operator are reduced to a single Boolean value.
- A truth table is used to evaluate the reduced Boolean expression to a single Boolean value.
- A decision is made with the single Boolean value.
- A flowchart illustrates the steps of making a decision with a Boolean expression.

`true && false` evaluates to `false`

`true || false` evaluates to `true`

Click for animation: Review the key takeaways with students.

Wrap Up (10 mins)

Video: CS Principles: Conditionals - Part 1 Boolean Expressions - Video As a class watch the video on Boolean expressions.

Journal: Have students add the following words and definitions to their journals: Boolean Value and Boolean Expression.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Can a computer evaluate an expression to something between true and false? Can you write an expression to deal with a "maybe" answer?

Teaching Tip

Running the Activity: This activity asks students to follow along as a number of core concepts for programming are introduced. The model is typically that a term or concept is introduced and modeled and then afterwards students are encouraged to try it out on their own. Trying it out typically means they are writing information on a sticky note and sharing it with another group before discussing the results with the whole class.

Slides with animations have an icon in the bottom left corner to let you know you need to click to reveal more of the slide's content.

To help you more easily prepare the activity and keep track of your instructions, detailed instructions have been included as speaker notes in the presentation. Here are some tips to help you throughout the presentation.

- There are opportunities throughout the presentation for students to actively engage. At these moments students should be making things with their manipulatives or using them to answer questions. Use these opportunities to check progress.
- There is a fair amount of new vocabulary introduced but it is introduced gradually and with intentional repetition. Make a point of actively modeling the use of new terms.
- The most important goal here is building a mental model. It is ok if students have some open questions that will get resolved over the subsequent conditional lessons.
- Both you and students can use the "Key Takeaways" to check your understanding at the end.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 6: Conditionals Investigate

Overview

In this lesson students work with partners to investigate three versions of the "Thermostat App" to understand how boolean expressions and conditional statements allow programs to make decisions. In each guided investigation students first watch a short video on a concept, then use a working app to predict how new features work, then investigate the code to see how those features are implemented, and finally modify the code to add expanded features. To conclude the lesson, students review and discuss common programming patterns with conditionals.

Purpose

After building a conceptual model for boolean expressions and conditional statements in the previous lesson, this lesson allows students to see how they are actually implemented in code. This lesson also introduces common programming patterns when using variables. Students will have some opportunities to modify working code in this lesson, but the most significant practice with conditional statements and boolean expressions will come in the following lesson.

Agenda

Warm Up (5 mins)

Preview the Lesson

Activity (35 mins)

Investigation 1: If-Statements (Levels 2-3) - 15 mins

Investigation 2: If-Else Statements (Levels 4-6) - 10 mins

Investigation 3: Logical Operators AND OR (Levels 7-8) - 10 mins

Wrap Up (5 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Identify common programming patterns using boolean expressions and conditional statements
- Explain the purpose of those programming patterns with boolean expressions and conditional statements both in terms of how they work and what they accomplish
- Modify apps that make use of common programming patterns with boolean expressions and conditional statements to adjust their functionality

Preparation

- Review the example apps and the prompts that students will be asked to respond to for each
- Review the information covered in the slides

Links

Heads Up! Please make a copy of any documents you plan to share with students.


For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

Teaching Guide


Warm Up (5 mins)

Preview the Lesson

 **Prompt:** A water park will let a visitor on a ride if they are 48 or more inches tall OR they are 14 years old or older. Make a flowchart for this decision. Make sure to use comparison operators (`<`, `>`, `==`, etc.) and logical operators (`&&`, `||`, `!`) when you write your Boolean expression.

Discuss: Have partners share their responses at their tables. Then discuss answers briefly as a class.

Remarks

 In everyday conversation it is common to switch between using the words "when" and "if". Here's some examples of what this looks like

- "When the user clicks the button..."
- "If the user clicks the button..."
- "When the user has more than 100 points..."
- "If the user has more than 100 points..."

Today we want to be careful about how we use these words. To make things simple, we're going to use the following rules.



- "when": Means there is an `onEvent` to respond to user input. The app does something "when" the user clicks.
- "if": Means there is a conditional statement that decides what pieces of code to run. The app does something "if" a boolean expression evaluates to true.



We'll talk more about this later but for now keep an eye out for the difference between "when" and "if".

Activity (35 mins)

Group: Place students in pairs. One student per group should navigate to the lesson on Code Studio.

Investigation 1: If-Statements (Levels 2-3) - 15 mins

  **Level 2 - Video - Conditionals: If Statements** As a class watch the video on if-statements.

  **Level 3 - Lemon Squeeze App Pt 1:** This code investigation includes a number of steps to help students get familiar with a new app.

- **Form Pairs:** Place students in pairs
- **Play the Game:** Let students play the game for a couple minutes
- **Assign Code Sections:** Count off pairs by three and assign them to one of the three code sections.
- **Read Code:** Groups should carefully read the code for their section making sure they understand how it works. Give them 3 minutes to do so.
- **Explain Your Section:** Have groups find members of the other two sections and carefully explain how their section works. Give each group 1-2 minutes to do so.
- **Class Discussion:** Ask a few members of each section to quickly share out how their section works. Display the code at the front so you can talk through it together.

Discussion Goal

Optional Warmup Prompt: This prompt is optional and helpful further synthesize key points from the previous lesson. If you are able to quickly move to the main activity and believe your class does not need this review consider skipping this prompt.

Teaching Tip

Show the Video at the Front: In order to better keep the group together we recommend you show the videos at the front of the room.

Reinforce When vs. If: As you make your way through these examples model using "when" and "if" as described in the warm up. For example, "when" the buttons are clicked the app is deciding "if" the temperature can change.

- **Modify:** Have groups return to their original seats. Give them a couple minutes to work on the modify the app so that the game ends once a user has 0 lives remaining.

Investigation 2: If-Else Statements (Levels 4-6) - 10 mins

📺 Level 4 - Video - Conditionals: If-Else Statements

As a class watch the video on if-else statements.

📺 Level 5 - Video - Conditionals: If-Else-If Statements

As a class watch the video on if-else statements.

📺🔗 **Level 6 - Lemon Squeeze App Pt 2** This program is an updated version of the Lemon Squeeze app. This time students should continue to work in partners but do not need to work with other groups. They will need to

- **Play the Game:** Have students play the game
- **Discuss Changes:** Talk through how the game plays differently now
- **Find the if-else-if command:** Have students find the command
- **Draw a flowchart:** Students should draw a flow chart for the if-else-if command in their journals
- **Modify the code:** Modify the program to make the lemon even smaller when the player has more than 15 points.

💡 Teaching Tip

Running the First Investigation: This first investigation is supposed to get students moving around the room while also motivating them to carefully read a new program. As you circulate the room encourage them to read carefully, ask good questions, and be comfortable asking where they don't understand something.

💡 Teaching Tip

Running the Second Investigation: This second investigation remains focused on reading code, but students are more responsible than last time for showing their work. Students will need to create a flowchart for the new if-else-if statement and make more significant modifications to the code. Continue to emphasize open discussion and collaboration between partners.

Investigation 3: Logical Operators AND OR (Levels 7-8) - 10 mins

📺 Level 7 - Video - Conditionals: If-Else Statements

As a class watch the video on if-else statements.

📺🔗 **Level 8 - Lemon Squeeze App Pt 3:** This program is a final version of the Lemon Squeeze app. Again students should work with partners.

- **Play the Game:** Have students play the game
- **Discuss Changes:** Talk through how the game plays differently now
- **Find the if-else-if command:** Have students find the if-else-if command that was added
- **Draw a Flowchart:** Students should draw a flowchart of this new if-else-if command
- **Modify the code:** Modify the program to add a new username and password. Students will have to modify the code as well as the user interface.

💡 Teaching Tip

Running the Third Investigation: In this final investigation students are primarily working independently to design their flowchart and modify the code. While you should quickly have students share out what they noticed about how the app changed, spend most of your time circulating answering questions students have as they modify the programs or draw their flowcharts.

Wrap Up (5 mins)

📺💬 **Prompt:** What is the difference between an if-statement, an if-else statement, and an if-else-if statement? How are they similar?

📺 **Review Patterns:** Have students read the pattern introduced in Level 9. This pattern appears frequently with if-else-if statements. Students' understanding of this pattern is assessed in the Check Your Understanding Question.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: When creating an if-else-if statement you should always make your first condition the most specific. Write a short paragraph responding to the questions below.

- What does it mean to put the most specific case first?
- Why is it important to put the most specific case first? What types of errors does it help avoid?

Discussion Goal

Goal: Some key points to call out in this discussion.

- These statements are more the same than different. They all use boolean expressions to decide whether to run certain pieces of code.
- if-statements check if one boolean expression is true. If it is, a piece of code is run. Otherwise the code runs as normally.
- if-else statements add the functionality that if the condition is false it can still run some code before returning to running the program as normal
- if-else-if statements can check more than one boolean expression. They will only run the code for the first boolean expression that evaluates to true.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 7: Conditionals Practice

Overview

In this lesson students spend most of their time practicing using the skills and processes they have learned about conditionals. At the conclusion of the lesson students discuss the main things they realized and still have questions about at the conclusion of this lesson.

Purpose

This lesson is students primary opportunity to get hands on with conditionals in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing. In the following lesson students are encouraged to work independently.

Agenda

Warm Up (5 mins)

Quick Warm Up

Activity (35 mins)

Practice Time

Wrap Up (5 mins)

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Write programs that use boolean expressions and conditional statements with the support of sample code.
- Debug programs that use boolean expressions and conditional statements

Preparation

- Review the programming challenges students will be completing
- Review the Debugging Guide for ideas on how to support your students during the lesson

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation
- **CSP Debugging Guide**

Teaching Guide

Warm Up (5 mins)

Quick Warm Up

Remarks

🔔 Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!


Teaching Tip

Move Quickly to the Activity: There's a lot in the main activity of today's lesson. You may optionally wish to do a quick vocabulary review or address any questions that came up in the last lesson. Otherwise, give students more time to get hands on with some code.

Activity (35 mins)

Practice Time

Group: It is recommended that students work in pairs for this lesson and a number of the activities feature discussion prompts. Consider using pair programming, having drivers and navigators switch every 3 minutes, not every level.

 **Do This:** Direct students to Code Studio, Lesson 3 Level 2. Then briefly remind students about debugging skills that they will be using in today's activity.

Remarks

Today you're mostly going to practice what we've learned about programming with conditionals. As always you should be using the debugging process to help you as you work on issues. Today we're also going to be working on finding two types of errors

1. Syntax errors show up when you type code that breaks the rules of the programming language. You can check for errors and warnings
2. Logic errors show up when you type valid code but it works incorrectly. Today you're going to focus on testing your code to make sure you don't have logic errors.

Other errors you may encounter include:

- Run-time error - a mistake in the program that shows when running the program. These are defined by the programming language.
- Overflow error - an error that occurs when a computer tries to handle a number outside of the defined range of values.

🔔 **Levels 2-4:** These levels only use the `console.log()` command which prints commands in the debug console. Here are a few things to keep an eye out for

- Levels 2-3 ask students to write Boolean expressions using comparison operators. Students may need to quickly review the comparison operators `<`, `>`, `<=`, `>=`, `==`, `!=`
- Level 4 asks students to write Boolean expressions with logical operators `&&`, `||`, `!`

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

Levels 5-9: These levels practice if-statements while working with a star color-changing app.

- Levels 5-6 involve setting up an if-statement that becomes an if-else statement.

- In Level 7 students follow a pattern to create a lengthy if-else-if statement.
- For Level 8, make sure students slow down the running of the code to understand what's happening. It's suggested that students use the slider to slow down the code.
- Level 9 demonstrates that Boolean expressions can be written as conditional statements, and vice versa

Levels 10-11: The levels return to the "Can I Adopt a Cat?" flowchart from the Conditionals Explore activity. Students will use the flowchart to work out the logic of the if-statements in a their program.

- A new block appears in these levels: `getNumber()` . This is different than `getText()` . `getNumber()` gets a number from a user input that can be used mathematically.
- Level 11 can be completed many different ways. There are different combinations of Boolean expressions using `&&` and `||` . Students should regularly test their apps to see if their Boolean expressions are working properly.

Extension Opportunities:

- Level 4: Students can add more variables and create complex Boolean expressions. One challenge might be to assign a String to a variable and compare that string to another.
- Level 10: There are multiple solutions. If students build their if-statement using only `&&` encourage them to figure out how to build it using only `||` . They many need to switch the content of the if and else branches.
- Level 11: Create another input (i.e. How many cats do you already own?). Students use this information to craft more complex if-statements.

Wrap Up (5 mins)

Prompt: What aspects of working with conditionals do you feel like clicked today? What do you still feel like you have trouble with?

Remarks

Conditionals can be a little bit tricky, but I saw a lot of good progress today in nailing down this concept. We may have a few lingering questions, but you also have a lot of resources available. Next time you'll have a chance to put all this together by programming an app that starts with "the blank screen"!

Discussion Goal

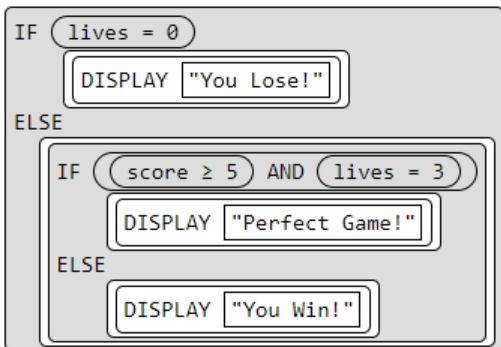
Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What will be displayed after this code segment is run?

```
score ← 4
lives ← 3
```



Question: The program below asks a user to type in a number and then will output a message. What number will a user need to input for the message "COLD" to be displayed?

```
number <- INPUT()

IF (number >= 10)
{
  IF (number <= 20)
  {
    DISPLAY("MEDIUM")
  }
  ELSE
  {
    DISPLAY("HOT")
  }
}
ELSE
{
  DISPLAY("COLD")
}
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **CRD-2** - Developers create and innovate using an iterative design process



This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 8: Conditionals Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Museum Ticket Generator app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. A progression of levels guides students on the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Warm Up (2 mins)

Intro the Project

Activity (38 mins)

Build the Museum Ticket Generator

Wrap Up (5 mins)

Assessment: Make Project

View on Code Studio

Objectives

Students will be able to:

- Recognize the need for programming patterns with Boolean expressions and conditional statements as part of developing a functioning app
- Implement programming patterns with boolean expressions and conditionals statements to develop a functioning app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Preparation

- Review the different steps students will be asked to complete as they build the app
- Review the information covered in the slides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP Conditionals Make - Museum Ticket Generator App** - Activity Guide

Make a Copy ▾

Teaching Guide

Warm Up (2 mins)



Intro the Project

Remarks

For the past few days, we've learned a lot about using conditional statements to help apps make decisions. In today's Make Project you'll be practicing both making flow charts and writing complex conditionals statements as you build a Museum Ticket Generator app.

Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Activity (38 mins)

Build the Museum Ticket Generator

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Distribute: Make sure students have access to scrap paper and pencils / paper for drawing flow charts.

Level 2 - Explore: Have students explore the working Museum Ticket Generator App in Level 2. If students are not working in pairs they should still discuss the prompts with a neighbor.



Prompt:

- What does this app do?
- What are the inputs?
- What are the outputs?
- What variables do you think would be necessary for this app to work?
- What kinds of conditional logic do you think are necessary to make it work?



Remarks


Now let's build the this app. The screen has been set up for you - it's your job to add the code!

Distribute: Give students copies of **CSP Conditionals Make - Museum Ticket Generator App - Activity Guide** if you will be using it during the class.

Do This: Direct students to level three where they complete the Museum Ticket Generator App. Based on the needs of your classroom decide whether you will collectively go through the activity guide or have students complete it individually. Afterwards give them time to work on their projects and circulate the room to offer support. Students who finish early can work on the extensions suggested in the activity guide.



Discussion Goal

- What does this app do?
 - Generates a ticket to a museum based on input data from the user.
- What are the inputs?
 - The dropdowns for age and day of the week
 - The discount code text field
 - The button to generate the ticket
- What are the outputs?
 - The ticket generated at the bottom of the screen
- What variables do you think would be necessary for this app to work?
 - The input information listed above will all need variables (age, day, discountCode), but they may want variables for other information calculated during the app.
- What kinds of conditional logic do you think are necessary to make it work?
 - The pricing rules help you know what to write. They'll definitely need to use if-else-if statements and AND / OR operators in order to implement these rules since they include multiple conditions to check.

 **Submit:** Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Wrap Up (5 mins)

Remarks

  Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Run the code on turtle mode
- Add the variables to the watcher
- Explain the code to a friend

Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Lesson 9: Functions Explore / Investigate

Overview

Students begin the lesson by considering two ways to write out the lyrics of a song, one that includes a lot of repeated text and one that does not. After exploring this example students complete a series of investigate activities in which functions have been used to remove repeated code from a program. At the conclusion of the lesson students discuss the concept of a function to synthesize their learning and add definitions to their journal.

Purpose

This lesson is both the Explore and Investigate components of the EIPM sequence for functions. Functions are best understood by actually using them in a program, and so while the Explore component is relatively shorter, students actually are given many opportunities to observe how functions are used in programs. There is a heavy emphasis on running programs slowly to see how functions change the order in which programs run.

If you are a teacher with more experience with the concept of functions you may be wondering where other related concepts like parameters, arguments, and return values, will be introduced. In this unit these additional concepts are not learning objectives, and they will not be discussed in detail until later units. For now, it is fine for students to think of functions simply as a way to name a collection of commands so that they can be used in multiple places within your code.

Agenda

Warm Up (5 mins)

Explore Song Lyrics

Activity (30 mins)

Investigate

Wrap Up (10 mins)

Synthesizing Discussion

Assessment: Check For Understanding

[View on Code Studio](#)

Objectives

Students will be able to:

- Use appropriate vocabulary to describe the declaring and calling of functions
- Trace the flow of execution in programs that declare and call functions
- Describe the way a function call interrupts the normal flow of execution within a program
- Modify programs that declare and call functions to adjust their functionality

Preparation

- ☐ Print copies of the activity guide of song lyrics used in the warm up or determine how you'll project it to the class.
- ☐ Review the apps that students will investigate in the app

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **Song Lyrics** - Activity Guide

Make a Copy ▾

Teaching Guide

Warm Up (5 mins)

Explore Song Lyrics

Distribute: Share print copies of **Song Lyrics - Activity Guide** or project the activity guide where all can see it.

Prompt:

- In Style 1, what line of the song do you sing after line 09? What about in Style 2?
- Style 2 uses fewer lines to write. Are there fewer lyrics to sing?
- What are the benefits of writing a song in Style 2?

Remarks

This second song is written in a style that we're going to use to write some of our programs. In fact, we're going to start off by looking at a program that "sings" this exact same song.

Activity (30 mins)

Investigate


Group: Group students in pairs.

 **Do This:** Direct students to Level 2 for the lesson on Code Studio.

Level 2 - Explore Song Lyrics: This level is a continuation of the Explore activity in the warmup. Students should watch the code run slowly to see how the code that is running will jump down to the bottom of the program (the function declaration) when the name of the function is used (the function call).

Remarks

This concept that we just explored in both text and a program is called a "function". We're going to watch a short video that explains it in more detail.

 **Level 3 - Video:** Watch the video as a class. The most important two takeaways are that

- Functions are declared in your program in order to give a group of commands a name
- Functions are called to run those commands.
- A function is only declared once but is called as many times as you wish.

 **Level 4 - Investigate Score Clicker**

- Discuss: Once students have had a chance to run the code ask them to explore it and be ready to share responses to the questions provided.
 - The function is declared on line 22 but called three times, on lines 5, 11, and 18.
 - This way of writing the program makes the code in each of the buttons much simpler to read. It also removes repeated code.
- Modify: Have students modify the program as instructed. All of the modifications should be made from within the function. Afterwards call out the following points.
 - If you change the function you can use those changes every place the function is called.
 - These changes make it easier to debug your program since you are not writing redundant code. You only need to rewrite and debug the code once.

Discussion Goal

Goal: This lesson does not feature a large hands-on explore segment, but students should still get a quick introduction to the concept of a function before jumping in the code to try it out. Run this discussion to help motivate the ideas underlying functions. Here are the main points to bring out.

- In Style 1 you read line 10 after line 09. This is the normal order we're used to.
- In Style 2 you sing the lyrics "out of order". After line 09 you're going to sing line 35
- It's shorter to write the song but it's the same song and will take the same time / lyrics to sing.
- Style 2 removes repetition and it also makes it a little easier to understand the overall structure of the song

Level 5 - Investigate Lemon Squeeze App with Functions

- **Modify:** Students in this level are asked to create a function themselves by identifying repeated code. They'll need to create a single function for this repeated code.
- **Discuss:** After creating their function lead a short discussion on the benefits of creating a function to remove repeated code.

Wrap Up (10 mins)

Synthesizing Discussion

Journal: Review the updateScreen pattern with students. Afterwards have them add to their journal definition for a function and a function call.

Prompt: Reflecting on today's lesson about functions:

- What did you learn?
- What are you uncertain about?

Discussion Goal

Goal: Use this quick discussion to identify any open questions that students have at the conclusion of the lesson. These can help you know where to focus attention in the next lesson.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: In your own words describe the benefits of creating functions in your code.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 10: Functions Practice

Overview

In this lesson students spend most of their time practicing using the skills and processes they have learned about functions. At the conclusion of the lesson students discuss remaining questions in anticipation of their Make project in the following lesson.

Purpose

This lesson is students primary opportunity to get hands on with conditionals in code prior to the Make activity in the following lesson. Give students as much class time as you can to work through these. For this lesson it's recommended that you place students in pairs as a support and to encourage discussion about the challenges or concepts they're seeing. In the following lesson students are encouraged to work independently.

Agenda

Warm Up (5 mins)

Quick Warm Up

Activity (35 mins)

Practice Time

Wrap Up (5 mins)

Assessment: Check For Understanding: AP Practice

View on Code Studio

Objectives

Students will be able to:

- Write programs that use functions with the support of sample code
- Debug programs that use functions
- Identify opportunities to use functions to reduce repeated code within a program

Preparation

- Review the video in the level progression that covers the topic of global vs. local variables
- Review the map level about "When to make a function?"
- Review other programming levels to be better prepared to support students

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

Teaching Guide

Warm Up (5 mins)

Quick Warm Up

Remarks

🔗 Today we're going to have a chance to practice programming with a lot of the concepts and patterns we've explored over the last two lessons. I encourage you to go through these with a partner, but pay close attention to what each other is doing. In our next lesson you're going to have to use a lot of these on an independent project, and these activities are good practice for what you'll find there! Alright, let's get to it!

Teaching Tip

Move Quickly to the Activity: There's a lot in the main activity of today's lesson. You may optionally wish to do a quick vocabulary review or address any questions that came up in the last lesson. Otherwise, give students more time to get hands on with some code.

Activity (35 mins)

Practice Time

Group: It is recommended that students work in pairs for this lesson and a number of the activities feature discussion prompts. Consider using pair programming, having drivers and navigators switch every 3 minutes, not every level.

Distribute: Optionally pass out a plastic cup or other manipulative they can place on their computer when they are stuck as a signal that they need support.

 **Do This:** Direct students to Code Studio, Lesson 10 Level 2

Remarks


Today you're mostly going to practice what we've learned about programming with conditionals. I'm here to help you when you need, and you can put this cup on your computer when you need help. However, I first want to remind you of the following:


- Use your debugging skills. Try to zoom in on precisely where you're getting stuck.
- Talk to your partner! That's what they're there for!
- Hover over blocks to read the documentation about how they work.
- Read the resources in the Help & Tips tab
- Talk to the group next to you. If another group asks for help make sure to take some time to talk it through with them.

Teaching Tip

Providing Support: Circulate around the room through the lesson encouraging students to use the strategies introduced at the beginning of the lesson. Students have a number of supports at their fingertips, so a big part of your role is helping build their independence in using those resources.

🔗 **Levels 2 - 6 Declare and Call Functions:** In these levels students practice declaring and calling functions. At first students practice calling functions that have already been declared for them. Students can focus their energy on the syntax of calling a function and how using functions changes the order in which lines of code run. Later in the progression they practice finding repeated code and are guided through how to create a function in its place.

 **Display:** Scope Practice video. This video is located on the slides and on Level 7. You may opt to have students watch it alone or as a class.

 🔗 **Levels 7 - 8 Function Scope:** These two levels return to a topic that was covered in the variables lessons as well: variable scope. While students do not need a deep understanding of scope at this point, they will in some instances encounter debugging challenges that arise because of it.

📍 **Level 9 - 10 Creating Functions:** In these levels students revisit the Movie Ticket app and are challenged to think through the process of declaring a function with code that they anticipate could be repeated. Level 9 is a map level about when students should decide to create functions.

Wrap Up (5 mins)

🗨️ **Prompt:** What aspects of working with functions clicked today? What do you still feel like you have trouble with?

Assessment: Check For Understanding: AP Practice

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What will be displayed after this code segment is run?

💡 Teaching Tip

Never - After - During - Before: In the map level students are introduced to a framework for thinking about their development with using functions. The primary question is "when should I make a function". In general, you want students to make their functions earlier, as this both improves the process of writing code and reflects deeper understanding of code structure. Throughout this course, you want to see them moving along the "never - after - during - before" scale.

At this point students are still learning to identify repeated code and replace it with a function. This would align with the "after" level. In the Functions Make project and the Unit 4 project they are encouraged to anticipate the need for functions in advance. Rather than write code twice and then remove the duplicate code by creating a function later, they should begin deciding early that they'll need a function. The "updateScreen()" pattern helps reinforce this point.

Reinforce this language in the classroom, though remember it's only a guide. Not every student will immediately be able to move on to "during" or "before". Different approaches also work better in different contexts, and many experienced programmers will typically operate in the "during" mode unless they're building a large and complex project.

💡 Teaching Tip

Level 7 is a Map Level, which contains a review of variable scope. It is highly recommended that you watch the video in Level 7 prior to the lesson. Slides are available for this lesson if you would like to optionally review as a class.

💬 Discussion Goal

Goal: Use this opportunity to address any lingering questions or misconceptions in the room. You can also use this as a source of discussion topics to kick off the following lesson. As you lead the discussion, call out the many resources students have access to help when they're getting stuck.


```
day_of_week = "Saturday"
```

```
day_of_week = "Monday"
```

```
IF ( day_of_week = "Saturday" ) OR ( day_of_week = "Sunday" )
```

```
  weekend
```

```
ELSE
```

```
  weekday
```

```
PROCEDURE weekday
```

```
  DISPLAY "School day"
```

```
PROCEDURE weekend
```

```
  DISPLAY "Day off"
```

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 11: Functions Make

Overview

Using Programming Patterns and a step-by-step approach students make their own version of a Quote Maker app. At the beginning of the lesson students are able to explore a working version of the app. They are then given the design elements of the app but begin with a blank screen. Students use an Activity Guide to go through the high level steps they should use to develop their app but leaves it to them to decide how to write the code. At the end students submit their apps which can be assessed using a provided rubric.

Purpose

This lesson is an opportunity for students to take on the "blank screen" and build the code that runs an app entirely from scratch. Guidance provided throughout the lesson helps students break down the large task of "building an app" into more incremental steps that they can use on future projects, including this unit's final project and the Create PT.

Agenda

Warm Up (2 mins)

Anticipating Functions

Activity (38 mins)

Build the Quote Maker App

Wrap Up (5 mins)

Assessment: Make Project

View on Code Studio

Objectives

Students will be able to:

- Recognize the need for a function to reduce repeated code while developing a functional app
- Implement a function using programming patterns while developing a functional app
- Write comments to clearly explain both the purpose and function of different segments of code within an app
- Use debugging skills as part of developing an app

Preparation

- Review the Activity Guide to decide if you will use it with your students

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP Functions Make - Quote Maker App** - Activity Guide [Make a Copy](#)

Teaching Guide

Warm Up (2 mins)



Anticipating Functions

Remarks

For the past few days, we've learned a lot about using functions to simplify code and make it easy to edit. So far we have learned how to use functions to remove repeated code in our programs. Early on it's common to realize you need a function only after you've completely written the same chunk of code in two places. As you get better, however, you'll realize sooner and sooner in the process that you're likely going to need a function, and will be able to avoid writing repeated code before putting it inside a function. Eventually you'll want to anticipate what your functions are going to be even before you start programming. These are the "during" and "before" levels we read about yesterday.

In today's Make lesson and the following project you'll have a chance to practice this. See if sometimes you can work that "during" or "before" category.

Teaching Tip

When to Make Functions: The end of the previous lesson and the beginning of this lesson both emphasize a transition students will go through in making functions. Early on they are likely to either not use functions or only realize the need one after they've already written duplicate code. As they get more experienced as programmers, however, they will likely realize sooner and sooner in the process that a function will be necessary. As students grow more experienced they should actually be able to anticipate and write their functions from the beginning of designing their program. Use this brief introduction to reinforce that language.

Activity (38 mins)

Build the Quote Maker App

Group: Make a determination as to whether this project will be completed in pairs or individually. You may even choose to let students decide.

Do This: Have students explore the working Quote Maker App in Level 1.



Prompt: If students are not working in pairs they should still discuss the prompts with a neighbor.

- What does this app do?
- What are the inputs?
- What are the outputs?
- How could a function be used in this app?



Remarks

Now let's build this app. The screen has been set up for you - it's your job to add the code!

Do This: Direct students to level three where they complete the Quote Maker App. An optional Activity Guide is provided if students would like guidance in creating the app. The most relevant programming pattern is displayed on a slide. Review this pattern quickly with students, if needed.

Submit: Encourage students to check the rubric on the last page of the Activity Guide before submitting.

Wrap Up (5 mins)

Remarks

💡 Awesome work today! Make sure to submit your project when you're done with it!

Assessment: Make Project

Use the rubric provided with the project to assess student projects.

Discussion Goal

- What does this app do?
 - Display a quote with user-defined fonts and background colors.
- What are the inputs?
 - The dropdowns for background color and font family.
 - The quote text field
 - The slider to generate the font size
- What are the outputs?
 - The fully designed quote displayed at the bottom of the screen
- How could a function be used in this app?
 - Students may struggle with this question, and it's ok to not have it fully answered before beginning the project.
 - A function can be used to update the screen.
 - The function contains the conditional that is checked everytime the user interacts with the screen.

Teaching Tip

Supporting Students: While students are working on their apps, circulate the room and check in with students who need a little help. Encourage students to collaborate and discuss bugs with each other.

Debugging: Review with students steps they can use to debug if they get stuck:

- Run the code on turtle mode
- Add the variables to the watcher
- Explain the code to a friend

Teaching Tip

Maximize Work Time: The wrap up is short to allow the maximum amount of time for students to complete the activity.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 12: Project - Decision Maker App Part 1

Overview

Using a Project Planning Guide, students work through the stages of creating an app from scratch. This is the first day of a three-day project. This lesson is devoted to the planning phase.

Purpose

The Practice PT gives students the opportunity to design and program an app from scratch. Welcome to The Decision Maker App! Students demonstrate mastery of variables, conditionals, and functions by combining these elements into a useful program designed to solve the problem of making a decision.

Agenda

Warm Up (5 mins)

Intro the Project

Activity (40 mins)

Plan the Decision Maker App

Wrap Up (0 mins)

View on Code Studio

Objectives

Students will be able to:

- See rubric for guidance in measuring student learning

Preparation

- Review the project guide and make sure students have access to copies

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP U4 Practice PT Rubric** - Rubric [Make a Copy](#)
- **CSP U4 Practice PT Planning Guide** - Planning Guide [Make a Copy](#)

Teaching Guide

Warm Up (5 mins)

Intro the Project

Remarks

🔗 Have you ever been stuck trying to make a decision? What movie should we watch? Where should we go for lunch? How many chocolate bars can I buy? For the next three days you will build an app to help people make a decision.

Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.


Activity (40 mins)

Plan the Decision Maker App

Level 1-2 - Explore: Have students explore the two sample apps in levels 1 and 2. These apps should spark ideas in what the students can create for this project.


 **Prompt:** For each app, discuss the following:


- What does this app do?
- What are the inputs?
- What are the outputs?
- What variables do you think would be necessary for this app to work?
- What kinds of conditional logic do you think are necessary to make it work?
- How could a function be used in this app?

 **Distribute:** Each student should have a copy of the Practice PT Decision Maker App Planning Guide.

Remarks

For this project you will create an app that helps a user make a decision. Your app must take in at least one number and one string from the user that will help to make the decision. All of this information will be used as part of the decision making process. In addition, your code must include at least one function used to update the screen.

 **Discuss:** Review the app requirements in the Planning Guide.

 **Display:** Show the steps students will complete today in the Planning Guide.

Step 1: Brainstorm App Ideas

- Students come up with three distinct ideas

Step 2: Choose one Idea

Step 3: Survey Your Classmates

Discussion Goal

Sample App #1: Where Should I Eat?

- Recommends a restaurant to the user.
- Inputs: text box for user name, drop down menu for restaurant type, up and down buttons to control the amount of dollars the user wants to spend.
- Outputs: Text box at the bottom of the screen which displays the recommendation and sound that plays when the user interacts with the app.
- Variables: username, type of restaurant, dollar amount, output text
- Conditional Logic: If the dollar amount within certain amounts, recommend different restaurants
- Function: Update the screen every time the user changes an input.

Sample App #2: Activity Finder

- Recommends an activity to the user.
- Inputs: text box for user name, drop down menu for time of day, drop down menu for activity level.
- Outputs: Text box at the bottom of the screen which displays the recommendation and sound that plays when the user interacts with the app.
- Variables: username, time of day, activity level, output text
- Conditional Logic: If the user selects certain times of day and activity levels, make targeted recommendations.
- Function: Update the screen every time the user changes an input.

- Students discuss with two classmates the decision their app will help the user make and decide what information is needed to make this decision.

Step 4: Storing Information

- Variables needed for storing information are listed in this section.

Step 5: Flowchart

- Students create a flowchart, following the "Can I adopt a cat?" sample flowchart from the Conditionals Explore lesson.

Step 6: Design User Interface

- There is space for students to design up to three screens. There are no screen requirements for the app, so students may use only one screen.

Wrap Up (0 mins)

No wrap up today. All time should be spent on the project.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

💡 Teaching Tip

Supporting students in Practice PT Lessons

The teacher plays the role of a guide throughout the Practice PT lessons. In preparation, you may want to set aside some time to complete the project yourself to identify potential points of confusion for your classroom.

In this first lesson, the classroom progresses together through steps 1-3, and then students work at the own pace for steps 4-6. Circulate the room and check in with students as needed to make sure instructions are clear and students understand expectations.

What should you expect?

- Students writing and sketching in the Planning Guide
- Active discussion around project ideas
- Students should only be on Code Studio if they have finished Steps 1-6 early and are ready to work in Design Mode in App Lab. If this is true, they can move on to the first level in Lesson 13.



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 13: Project - Decision Maker App Part 2

Overview

Students translate the plans they documented in Part 1 of the Practice PT to a working program in App Lab through a series of steps.

Purpose

The Practice PT gives students the opportunity to design and program an app from scratch. Welcome to The Decision Maker App! Students demonstrate mastery of variables, conditionals, and functions by combining these elements into a useful program designed to solve the problem of making a decision.

Agenda

Warm Up (2 mins)

Intro the Project

Activity (38 mins)

Build the Decision Maker App

Wrap Up (5 mins)

View on Code Studio

Objectives

Students will be able to:

- See rubric for guidance in measuring student learning

Preparation

- Make sure students will have access to their project guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP U4 Practice PT Planning Guide** - Planning Guide [Make a Copy](#)
- **CSP U4 Practice PT Rubric** - Rubric [Make a Copy](#)

Teaching Guide

Warm Up (2 mins)

Intro the Project

Remarks


🔗 Today we are continuing work on the Decision Maker App. Each level in Code Studio contains a step to think through while constructing your app.

Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Activity (38 mins)

Build the Decision Maker App

 **Do This:** Students now transition to Code Studio where they will build their app.

  **Display:** Display the steps for today along with the debugging steps on the board while students work.

Level 2: Design Mode

- Students create the screens they designed in Step 6 in the Planning Guide.
- Encourage students to use templates where appropriate to speed the process along.

Level 3: Create the Variables

- Using the list of variables from Step 4 in the Planning Guide students create their variables.

Level 4: Create the Function

- This is most likely the trickiest step.
- Students create an `updateScreen()` function with a conditional statement
- Students should reference the flowchart in Step 5 of the Planning Guide to help craft the Boolean expression(s) for the conditional statement.
- A comment is added to explain what it does (purpose) and how it does that (functionality)

Level 5: Add onEvents

- For each item that can be interacted with on the screen, students add an `onEvent` block where they update the appropriate variable and add a call to the `updateScreen()` function.

Teaching Tip

Supporting students in Practice PT Lessons

At this point in the project attention shifts to App Lab where students design and program their app. Continue to circulate the room and check in with students as needed to make sure instructions are clear and students understand expectations.

What should you expect?

- Planning Guides out on desks or tables so students can reference them while setting up their apps
- A healthy buzz in the classroom as students collaboratively debug
- There may be some student confusion on the steps to take in building their app. Direct students back to the instructions, and make sure they haven't skipped any of the levels where the steps are delineated.

Wrap Up (5 mins)

No wrap-up today. All time is spent on the project.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ AP - Algorithms & Programming



This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 14: Project - Decision Maker App Part 3

Overview

The final lesson in the Practice PT progression is devoted to feedback and improvements to the Decision Maker App. Students work with classmates to review and update the functionality of their apps before submitting the final project.

Purpose

The Practice PT gives students the opportunity to design and program an app from scratch. Welcome to The Decision Maker App! Students demonstrate mastery of variables, conditionals, and functions by combining these elements into a useful program designed to solve the problem of making a decision.

Agenda

Warm Up (2 mins)

Intro the Project

Activity (38 mins)

Test the Decision Maker App

Wrap Up (10 mins)

Assessment: Practice PT Decision Maker Project

View on Code Studio

Objectives

Students will be able to:

- See rubric for guidance in measuring student learning

Preparation

- ☐ Make sure students will have access to their project guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 4 - Variables, Conditionals, and Functions** - Presentation

For the Students

- **CSP U4 Practice PT Planning Guide** - Planning Guide [Make a Copy](#)
- **CSP U4 Practice PT Rubric** - Rubric [Make a Copy](#)

Teaching Guide

Warm Up (2 mins)

Intro the Project

Remarks

🔗 For the past few days we've been working on an app to help users make a decision. Now you will share that app with others, get feedback, make improvements, and submit for your final project.


Teaching Tip

Short Intro: The Warm Up today is short and light. Students should spend the maximum amount of time working on their projects.

Activity (38 mins)

Test the Decision Maker App

 **Group:** Put students in groups of 3-4.

 🔗 **Do This:** Direct students to Level 2 in Code Studio. They should pull out their Planning Guides and with their group complete Step 8: Testing.

Step 8: Testing

- Each app is tested by at least two students.
- The creator of the app watches these students use the app and records feedback from the testers and things the creator noticed while observing someone else using the app. For example, the creator may notice that the user has difficulty figuring out which button to click on the app to make it run. The creator notes this down in the Planning Guide.

Step 9: Pick Improvements

- In the Planning Guide, students pick at least one improvement that they can make based on feedback from their peers.

Step 10: Complete Your app

- Students use the feedback to update their app on Level 2.

Submit: Students complete a final check of their completed projects on Level 3. The rubric is displayed in the instruction box.

Wrap Up (10 mins)

Remarks

Congratulations! You built an app completely from scratch.

 **Reflection:** Complete the two reflection questions in the Planning Guide

- **Question 1:** Provide a written response that:
 - describes the overall purpose of the program

Teaching Tip

Supporting students in Practice PT Lessons

This is the final lesson where students complete their projects. The classroom energy will likely increase when students begin Step 8 and collect feedback from classmates. Encourage students to work productively during the testing phase so they have enough time to finish their projects.

What should you expect?

- It's ok to start the testing phase even if the apps aren't finished. Students should be able to explain to each other what they want their app to do when fully operational.
- Students should have their Planning Guides out again to record feedback and plans for implementing changes
- Frustration may be higher today if students feel rushed to debug and fix their projects for final submission. Make sure students have debugging partners and are actively making use of their debugging skills.
- If students are inspired to continue working on their projects or add additional features, you may need to extend the project an extra day or allow students to work outside of class.

- describes the functionality of your app
- describes the input and outputs of your app (Approx 150 words)
- **Question 2:** This project was created using a development process that required you to incorporate the ideas of your partner and feedback from your classmates. Provide a written response that describes one part of your app that was improved through input from EITHER your partner or feedback you received from classmates. Include:
 - Who specifically provided the idea or recommendation
 - What their idea or recommendation was
 - The specific change you made to your app's user interface or functionality in response to the recommendation
 - How you believe this change improved your app (Approx 150 words)

Assessment: Practice PT Decision Maker Project

Use the **CSP U4 Practice PT Rubric - Rubric** provided with the project to assess student work.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming



This curriculum is available under a
Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.

Lesson 15: Assessment Day

Overview

Students complete a multiple choice assessment which covers the unit topics.

Agenda

Assessment (25 mins)

Topic Coverage

Assessment Review (20 mins)

[View on Code Studio](#)

Preparation

- Preview the assessment questions

Teaching Guide

Assessment (25 mins)

📄🔒 Administer the Unit 4 Assessment, found on Code Studio. Make sure to unlock the assessment following instructions [here](#).

Assessment Review (20 mins)

Review the answers to the assessment with the class. Discuss any questions that come up and take note of topics where students may need extra review.

💡 Teaching Tip

Topic Coverage

The College Board has provided a bank of questions to help formatively assess student understanding of the content in the framework. These questions are mapped to topics with each topic having a handful of questions available.

The College Board has a few strict guidelines about how topic questions can be used. In particular, students may not receive a grade based on performance on topic questions nor can they be used for teacher evaluation. Beyond these requirements, however, they are primarily intended to formatively assess student progress and learning as they prepare for the end of course exam.

Within our own course we recommend that you use them in a variety of ways:

- Throughout the unit assign topic questions to students related to the topics students are learning about that day or that week
- Prior to the unit assessment assign topic questions to help students practice and prepare for the summative assessment
- After the unit assessment use these topic questions to help students track their progress towards preparation for the AP assessment

Unit 4: Variables, Conditionals, and Functions	Topic 1.4 Identifying and Correcting Errors
	<ul style="list-style-type: none">• Note: Students learn debugging practices in Unit 3 and continue to practice them throughout programming units. We recommend you initially use topic resources here and return to them later if you deem it necessary.
	Topic 3.1 Variables and Assignment
	Topic 3.3 Mathematical Expression
	Topic 3.5 Boolean Expressions
	Topic 3.6 Conditionals
	Topic 3.7 Nested Conditionals
	Topic 3.15 Random Values
	<ul style="list-style-type: none">• Note: While students are introduced to random values in Unit 3, we recommend you wait to use resources for this topic until Unit 4 when students have more experience programming expressions with random values

Click for more info: [Code.org CSP Topic Coverage](#)



This curriculum is available under a Creative Commons License (CC BY-NC-SA 4.0).

If you are interested in licensing Code.org materials for commercial purposes, **contact us**.