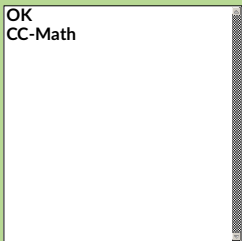


Simple Inequalities

(Also available for Pyret)

Students discover the Boolean data type, and apply knowledge of inequalities to simple programming problems.

Prerequisites	Problem Decomposition
Relevant Standards 	Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere).
Lesson Goals	<p>Students will be able to:</p> <ul style="list-style-type: none">• Describe the solution set of a simple inequality• Explain the 'Boolean' datatype
Student-Facing Lesson Goals	<ul style="list-style-type: none">• I can use two or more inequalities together and describe the area they enclose.• I can explain to someone else what a Boolean is.
Materials	<ul style="list-style-type: none">• Lesson slides (Google Slides)• Sam the Butterfly Starter File (WeScheme)• Inequalities Launch worksheet (original (Page 42), solutions)• Inequalities Explore worksheet (original (Page 43), solutions)• Left-and-Right (original (Page 44), solution)
Preparation	<ul style="list-style-type: none">• Make sure all materials have been gathered• Decide how students will be grouped in pairs
Supplemental Resources	<ul style="list-style-type: none">• Booleans Review (Quizizz)• Booleans Activity (Desmos Activity)• Inequalities Bundle (Desmos Activities)• Inequalities & Graphing Inequalities (Quizizz)• Inequality Graph Illustrator (Geogebra)
Key Points for the Facilitator	<ul style="list-style-type: none">• A Boolean is just another data type, like Number, or Image, but unlike the others there are only two values: <code>true</code> and <code>false</code> . While simple to explain, this different behavior can be confusing for some students.• Functions that produce Booleans are typically questions, so the names of the functions in this lesson read like questions. <p>For example, <code>safe-left?</code> , <code>onscreen?</code> are both functions that are asking if a condition, such as an image being on the screen, is true or false. * Role-playing can help students understand the jobs of <code>safe-left?</code> and <code>safe-right?</code> .</p>

For a textbook-like version of materials similar to these, you may wish to see the [prior unit-based version](#)

Glossary

Boolean :: a type of data with two values: true and false

coordinate :: a number or set of numbers describing an object's location

datatypes :: a way of classifying values, such as: Number, String, Image, Boolean, or any user-defined data structure

expression :: a computation written in the rules of some language (such as arithmetic, code, or a Circle of Evaluation)

Warmup

Students should have their workbook, pencil, and be logged into [WeScheme](#) and have their workbooks with a pen or pencil.

Introducing Booleans

15 minutes

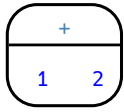
Overview

Students discover the concept of inequalities (or apply it, if they've seen it before) in programming, and extend their knowledge of data types, Contracts, and Circles of Evaluation.

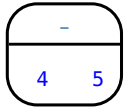
Launch

Ask students to evaluate Circles of Evaluation for simple expressions they've seen before, and ask them to convert them into code.

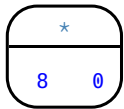
-



-

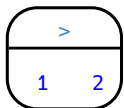


-

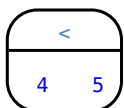


Then show them unfamiliar Circles of Evaluation, and ask them to hypothesize what they think they means, what they will evaluate to, and what the code would look like.

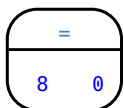
-



-



-



Have student type in these expressions. What did they get?

Values like `true` and `false` obviously aren't Numbers or Images. But they also aren't Strings, or else they would have quotes around them. We've found a *new datatype*, called a **Boolean**. Booleans are the answers to any yes-or-no question (for example: "Is five greater than two?", "Did a character hit a wall?", etc.)

Investigate

Have students open to the [Inequalities — Launch \(Page 42\)](#) worksheet and complete with a partner.

Synthesize

- Students will see functions on the worksheet that they've never encountered before! But instead of answering their questions, encourage them to make a *guess* about they do, and then type it in to discover for themselves.
- Explicitly point out that *everything they know still works!* They can use their reasoning about Circles of Evaluation and Contracts to figure things out.

Introducing Sam

30 minutes

Overview

Students are introduced to Sam the Butterfly: a simple activity in which they must write 1-step inequalities to detect when Sam has gone too far in one dimension.

Launch

Have students open the [Sam The Butterfly starter file](#) and click "Save."

Have students turn to the [Introducing Sam \(Page 43\)](#) and investigate the program with their partner.

Let students figure out that they need to press "Run" to see what the program does, and that the arrow keys control Sam.

- What is something you noticed about this program? Sam can be moved with the arrow keys, the *coordinates* are displayed at the top of the screen, the coordinates are all in the 1st quadrant, etc.
- What do you see when Sam is at (0,0)? Why is that? You only see part of Sam's wing. Sam's position is based on the center of Sam's image.
- How far can Sam go to the left and stay on the screen? Up to, but not beyond, an x of -40.
- How could we write this as an *expression*? $x \geq -40$, or $x > -50$

Every time Sam moves, we want to check and see if Sam is safe.

- There are three functions defined in this file. What are they?

Note: In this programming language, question marks are pronounced "huh?". So `safe-left?` would be pronounced "safe left huh?" This can be a source of some amusement for students!

Optional: For extra scaffolding...

- What *should* our left-checking function do? Check to see if x is greater than -50
- What *should* our right-checking function do? Check to see if x is less than 490
- What should `onscreen?` do? Answers may vary, let students drive the discussion, and don't give away the answer

Investigate

With their partners, students complete [Left and Right \(Page 44\)](#). Once finished, students can fix the corresponding functions in their Sam the Butterfly file, and test them out.

Students will notice that fixing `safe-left?` keeps Sam from disappearing off the left, but fixing `safe-right?` doesn't seem to keep Sam from disappearing off the right side! When students encounter this, encourage them to look through the code to try and figure out why. The answer will be revealed in the next lesson.

- Recruit three new student volunteers to roleplay those same functions, which have now been *corrected*. Make sure students provide correct answers, testing both `true` and `false` conditions using coordinates where Sam is onscreen and offscreen.

Common Misconceptions

- Many students - especially traditionally high-achieving ones - will be very concerned about writing examples that are "wrong." The misconception here is that an expression that produces `false` is somehow *incorrect*. You can preempt this in advance, by explaining that our Boolean-producing functions *should sometimes return false*, such as when Sam is offscreen.
- Push students to think carefully about corner-cases, such as when Sam is *exactly* at -50 or 490.

Synthesize

- Recruit three student volunteers to roleplay the functions `safe-left?`, `safe-right?` and `onscreen?`. Give them 1 minute to read the contract and code, as written in the program.
- For each of them, ask the volunteers what their name, Domain and Range are, and then test them out by calling out their name, followed by a number. (For example, "(safe-left? 20)!", "(safe-right? -100)!") **Note:** Do not ask `onscreen?` to roleplay beyond their contract! They'll get involved in the next lesson...

Additional Exercises

- Keeping Ninjacat in the Game ([original](#) , [answers](#))
- Converting Circles of Evaluation with Booleans to Code ([original](#) , [answers](#))
- Converting Circles of Evaluation with Booleans to Code ([original](#) , [answers](#))