Simple Data Types

(Also available in Pyret)

Students begin to program, exploring how Numbers, Strings, Booleans and operations on those data types work in a programming language. Booleans offer an excellent opportunity for students to explore the meaning and real-world uses of inequalities.

Lesson Goals	Students will be able to… • Identify examples of the following data types: Numbers, Strings, and Booleans • Write Numbers, Strings, and Booleans in the Interactions Area • Write expressions that produce values of those types
Student-facing Lesson Goals	• Let's explore Numbers, Strings and Booleans and identify what makes these data types unique.
Materials	• PDF of all Handouts and Page • Lesson Slides • Printable Lesson Plan (a PDF of this web page)
Preparation	• Make sure student computers can access WeScheme.
Key Points For The Facilitator	• Error messages are the computer trying to give us a clue that something is wrong. Model reacting to error messages with interest to demonstrate to students that the messages are a helpful tool.

Glossary

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

definitions area :: the left-most text box in the Editor where definitions for values and functions are written

editor :: software in which code can be written and evaluated

interactions area :: the right-most text box in the Editor, where expressions are entered to be evaluated

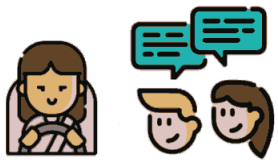
syntax error :: errors where the computer cannot make sense of the code (e.g. - missing commas, missing parentheses, unclosed strings)

Strings and Numbers *20 minutes*

Overview

Working together using a Driver/Navigator group setup, students experiment with the Editor. They explore Number and String data types, and how they behave in this programming language.

Launch



When programming in this class, you'll be working together using the *Driver/Navigator* model. Each group can only have one "Driver" - their hands are on the keyboard, and their job is to manage the typing, clicking, shortcuts, etc. If you're not a Driver, you're a "Navigator" - your job is to tell the Driver where to go, what to type, etc. A good Driver types only what the Navigator tells them to, and a good Navigator makes sure to give clear and precise instructions.

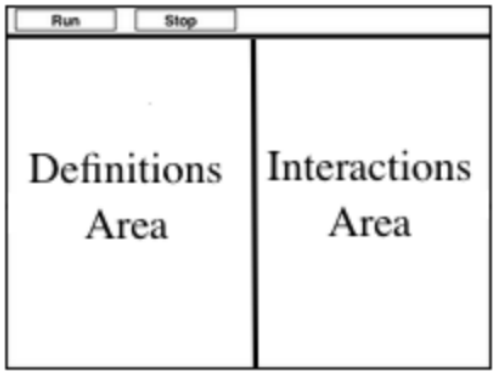
The Driver/Navigator Model

This model of pair programming is extremely useful for teasing apart the "thinking" step from the "typing" one. Students - especially those who are new to text-based programming or typing itself - can struggle to put their thoughts into the programming environment. This model allows them to focus on *communicating their ideas*, but letting the Driver focus on the coding. Likewise, the Driver has a chance to focus on syntax and programming. Finally, the requirement that ideas are translated through another person's hands is an excellent scaffold for getting students talking about their thinking and about code.

You can read more about the Driver/Navigator model here…

Have students open WeScheme.

This screen is called the ***Editor***, and it looks something like the diagram you see here. There are a few buttons at the top, but most of the screen is taken up by two large boxes: the ***Definitions Area*** on the left and the ***Interactions Area*** on the right.



The ***Definitions Area*** is where programmers define values and functions that they want to keep, while the ***Interactions Area*** allows them to experiment with those values and functions. This is like putting a set of function definitions on the board, and having students use those functions to compute answers on scrap paper.

We need to click "Run" to load the program when we first open a file and if we make a change in the ***Definitions Area***.

Clicking "Run" will also clear the ***Interactions Area***. For now, we will only be writing programs in the **Interactions Area** on the right.

Investigate

Math is a language, just like English, Spanish, or any other language. Languages have **nouns** (e.g. “ball”, “tomato”, etc.) and **verbs**, which are actions we can perform on these nouns (e.g. - I can “throw a ball”). Math and programming also have **values**, like the numbers 1, 2 and 3. And, instead of verbs, they have functions, which are actions we can perform on values (e.g. - “I can square a number”).

Languages also have rules for **syntax**. In English, for example, words don’t have `!` and `?` in the middle. In math and programming numbers don’t have `&` in them.

Languages also have rules for **grammar**. *The cat sat.* is a sentence, whereas *The sat cat.* is nonsense, even though all the words are valid syntax. The order of the words matters!

Keeping the importance of **syntax** and **grammar** in mind is helpful when learning to program!.

- Complete Strings and Numbers.
- What did you Notice? What do you Wonder?
  - *Check out the Synthesize section, below, for a list key take-aways from this activity.*
- Did you get any error messages? What did you learn from them?
  - *Most of the error messages we’ve seen were drawing our attention to **syntax errors**: missing commas, unclosed strings, etc.*

Synthesize

Our programming language knows about many types of numbers, and they behave pretty much the way they do in math. Discuss what students have learned:

- Numbers and Strings evaluate to themselves.
- Our Editor is pretty smart, and can automatically switch between showing a rational number as a fraction or a decimal, just by clicking on it!
- Anything in quotes is a String, even something like `"42"`.
- Strings *must* have quotation marks on both sides.

Booleans *20 minutes*

Overview

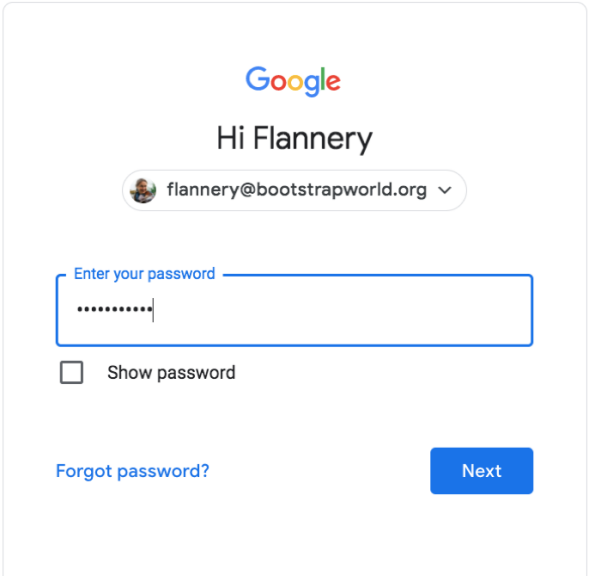
This lesson introduces students to ***Booleans***, a unique data type with only two values: "true" and "false", and why they are useful in both the real world and the programming environment.

Launch

What's the answer: is 3 greater than 10?

Boolean-producing expressions are yes-or-no questions and will always evaluate to either `true` ("yes") or `false` ("no"). The ability to separate inputs into two categories is unique and quite useful!

For example:



- Some rollercoasters with loops require passengers to be a minimum height to make sure that riders are safely held in place by the one-size-fits all harnesses. The gate keeper doesn't care exactly how tall you are, they just check whether you are as tall as the mark on the pole. If you are tall enough, you can ride, but they don't let people on the ride who are shorter than the mark because they can't keep them safe.
- When you log into your email, the computer asks for your password and checks whether it matches what's on file. If the match is `true` it takes you to your messages, but, if what you enter doesn't match, you get an error message instead.

Brainstorm other scenarios where Booleans are useful in and out of the programming environment.

Investigate

- In pairs, complete Booleans.

Students will make predictions about what a variety of Boolean expressions will return and testing them in the editor. Debrief student answers as a class.

Synthesize

What sets Booleans apart from other data types?