Unit 2
Contracts, Strings and Images

Unit Overview

Students are introduced to a set-mapping representation for functions, in which the function object exists as a means of translating points from a Domain into a Range. Coupled with their understanding of Circles of Evaluation, students generalize their understanding of functions to include other datatypes, including Strings and Images.

Agenda

30 min	Introduction
15 min	Contracts
10 min	Strings
35 min	Creating Images
5 min	Closing

Learning Objectives:

- Students will be able to write contracts for functions of Numbers, Strings and Images.
- Given an example of a function, students will be able to identify the Name, Domain and Range of that function.

Product Outcomes:

- Students learn to make and manipulate the basic elements of their games–numbers, strings, and images

State Standards See our Standards Document provided as part of the Bootstrap curriculum.
Length: 90 minutes

Materials and Equipment:

- *Student workbook folders - in pairs! - with names on folders*
- *Pens/pencils for the students, fresh whiteboard markers for the teachers*
- *On student machines: "Images.rkt" from source-files.zip | A blank WeScheme file])*
- *Class posters (List of rules, language table, course calendar)*
- *Language Table (See below)*

Preparation:

- *Write agenda on board*
- *Display class posters and Language Table*
- *Seating arrangements: ideally with clusters of desks/tables*
- *Optional: demo machine with projector to show the interactions and definitions windows*

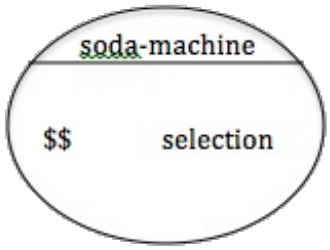
Types	Functions
Number	`+ - * / sq sqrt expt`

Introduction (Time 30 minutes)

- Welcome back! Do you remember what we did in the last class?
- At the end of class, you learned something extremely important: the Circles of Evaluation, and how to write simple programs in Racket.
- Suppose I have the math expression "2 + 3." How would I draw a circle of evaluation for this? How would I convert this into code?
- Remember that the parentheses are like our circles; they help us put our functions and inputs into the right order.
- How about "2 + 3 divided by 1 - 2"? (Answer: MAKE A PICTURE!!!!)
- Let's do a review challenge:
    - Turn to Page 3 in your workbooks, where it says "Circles Competition." On each row there is a mathematical expression in the first column, with room to draw the Circle of Evaluation and the Racket Code on the second and third columns.
    - Each row will be a round for this activity, so for Round 1 we'll just be looking at the first row. For this first row, we've even helped you out a little.
    - You will have one minute to do the following in groups: Draw the Circle of Evaluation, and then convert it to Racket code. Everyone must have the answer written down for your group to win points. GO!
    - *During the minute, walk around and see how groups are doing. Comment on good teamwork when you see it. Don't tutor much; let kids fail: they'll get it in review before the next round. When time is up (really 1 minute: the idea is to go quickly), give them a countdown: "30... 10... 5... 4... 3... 2... 1... PENCILS DOWN, EYES UP HERE!" Wait for total silence and complete attention.*
    - *Review and discuss. Assign points.*
    - Repeat for each additional round
- *Have kids complete the competition - give as little help as possible at this point. After the time is up, have students volunteer their answers. Review as necessary.*

Contracts (Time 15 minutes)

- Suppose you have to describe a soda machine. You know that the input to this machine will be money, and you have to get a soda in return. When you use a soda machine, does it only give you one type of soda? Wouldn't it be lame if you had to have a different machine for Coke than you did for Sprite?
- A soda machine, for example, takes in money and your selection, and gives you soda.




```
; soda-machine : Money Selection -> Soda
; Takes money and gives soda
```

- This explanation of a soda machine just talks about what goes in and what comes out. It doesn't talk about exactly how it works, and it doesn't require only a special kind of soda (like Coke or Sprite). It's useful to think about machines in this way, when all we want to know is how to use them. When you walk up to a soda machine, all you care about it tha you have some cash and you'd like to make a selection. Why worry about the details?
- How would you describe a coffeemaker? What kind of stuff goes into a coffeemaker? What kind of stuff comes out?
- How would you describe a lightbulb?
  Have the students brainstorm a few other machines, and their inputs and outputs.
- The **kinds of things** that go into a machine are called the Domain, and the **kind of thing** that comes out is the Range. When we pick a specific example (like selecting "Pepsi"), we are talking about inputs and outputs. So with our soda machine, the Domain would be "money and selection", while "one dollar and mountain dew" would be example inputs. A soda machine's range is Soda. Can you think of an example output? (Answer: Any type of money: quarters, dimes, dollars) Its range is soda.
- What is the domain and range of a coffeemaker? (Answer: The domain is coffee ingredients: coffee beans, water, etc. The range is types of coffee.)
- Functions in Racket are the same: the addition function needs two numbers as its domain, but those numbers don't have to be 4 and 5. They can be any numbers.
- You already know about a bunch of functions in Racket from math class: addition, subtraction, multiplication, division, etc.
- In Racket, a description of a function is called the **contract**. The contract tells you just what you need to know to use the function.
- A contract is also a promise: if you give the machine the kinds of things in its Domain, then you'll get something in the Range. If you give a pizza shop your order and some money, it promises to give you some hot, delicious pizza.
- Same thing in Racket: if you give plus two numbers, it will give you another number back.

- When we write a contract in Racket, it's like writing a note to ourselves about how to use the program. Just like with our soda machine, we don't care how it works when we're writing the contract. All that matters is *how to use the function.*
- Since Contracts are for use by humans (and not computers), we put a semicolon (`;`) at the beginning when we write them into the computer. This tells the computer to ignore the line: it's for Humans Only!
- We'd like to create a list of contracts for ourselves, so that we can keep track of these functions and exactly how they work. Once we've practiced entering these contracts into our book, I'll show you new functions that let you work with words and pictures!
- Are you ready? Turn to the front of your workbooks, to the MOST IMPORTANT PAGE, where it says \"Contracts\".
- A contract specifies
    1. the name of the function,
    2. the domain, and
    3. the range.
- *Show the contract for `+` on the board, and have students volunteer the contracts for the other math functions covered so far.*
- *Add other functions, like `/` and `*`. Leave these contracts written on the board.*
- *If students already know about square roots or squaring, you can add these functions as well. If not, take a moment to review what they do.*

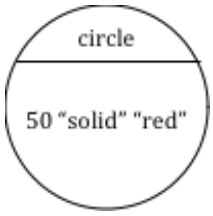
```
; sqrt : Number -> Number
; sqr : Number -> Number
```

Strings (Time 10 minutes)

- One of the following does not belong. Can you find it? 6, "cat", 0, -2, 7.5
- Why doesn't "cat" belong? Why isn't 7.5 the odd one out? Or -2?
  Circle "cat"
- What you have all stumbled onto is that not all values are the same. Some of them are Numbers and some are words.
- Words like "cat" are called Strings. A String is anything in quotation marks. 5 is not the same as "5"! The first is a number, and the second is a string.
- You'll notice that our math functions don't really work on Strings.
- What does it mean to add three and the word "cat"? It doesn't make sense, does it?
- Remember that Numbers in Racket are just simple programs, which evaluate to themselves. Strings work the same way! To create a String, you type in the word or words that you want, and put quotation marks on either side. The number `6.51` evaluates to itself. What will the string `"this is easy"` evaluate to? With your partner: Try creating a few strings on your own.
- Racket also gives us functions we can use to work with Strings. For example, there's a function called string-length. What do you think it does? I'll give you one hint: the contract.
  Write the contract on the board, but without the labels Name, Domain, and Range:
  `; string-length: String -> Number`
- What is the domain of this function? What is the range? (Answer: The domain is Strings, and the range is Numbers)
- *Now that the students have written down the contract, let them turn on their monitors and experiment, see if they can figure out what string-length does!*

Creating Images (Time 35 minutes)

- Let's talk about graphics.
- *On the board, draw the Circle of Evaluation for `(circle 50 "solid" "red")`*
- Take a look at this Circle of Evaluation.
- *Can someone tell me how to convert this into Racket code? Copy their answer on the board.*
- This uses a new function, which you've never seen before! What is its name?
- Every contract has three parts!
  Raise your hand if you can tell me what they are! (Name, domain, and range)
- Can you figure out the contract for circle? Based on the example, can you tell me what's in its Domain?
  Follow along on the board...




```
; circle: Number String String -> ...
(circle 50 "solid" "red")
```

- So what's the Range? What do you think this thing is going to give us back? A Number? A String? Type it in and try it

out!

- What it gives back is a new Type: Image!
- Now we're going to do the next step as a group.

- Exercise:

  Now let's examine the contract for another function. This time around, you're not allowed to touch the keyboard until you have copied the contract into your contract table, along with the contract for circle. Once you've done that, I want to see you try to draw a rectangle! GO!

  ```
  ; rectangle: Number Number String String -> Image
  ```

- Exercise:

  Here are a few more contracts. Once again, you have to have them written down before touching the keys. You'll have five minutes to figure out how to use each of these functions to make a shape! GO!

  ```
  ; ellipse: Number Number String String -> Image
  ; triangle: Number String String -> Image
  ; star: Number String String -> Image
  ; radial-star: Number Number Number String String -> Image
  ; text: String Number String -> Image
  ```

- *Let kids experiment with these functions and point out interesting results to the class.*
- Going Further - If time allows, you can go further into Manipulating Images or Making Flags!
- According to the domain of `circle`, the first input needs to be a number. That means it could be 4, 5, 100 or a 10000. But it could *also* be an expression that produces a number! Take a look at the following pieces of code:

  ```
  (circle (* 50 2) "solid" "red")
  (star (* 10 (string-length "hello")) "solid" "green")
  (text "purple" (string-length "purple") "purple")
  ```

- What images will they produce? See if you can figure it out, before typing it in to test your hypothesis. For practice, try converting each of these expressions into a Circle of Evaluation. Can see the connection between these expressions and the ones you did during the review game?

Closing (Time 5 minutes)

- *Who can tell us one thing we learned today?*
- *Who saw someone else in the class do something great?*
- *Cleanup, dismissal.*