<table>
<thead>
<tr>
<th>Expression: Circle of Evaluation:</th>
<th>Code:</th>
<th>Data Types:</th>
<th>Key Words: (&quot;Function Imposters&quot;):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
</tr>
<tr>
<td>The Design Recipe:</td>
<td>Contract:</td>
<td>Purpose Statement:</td>
<td></td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>●</td>
<td>●</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Examples Mapping:</td>
<td>Cond:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Coordinates
<table>
<thead>
<tr>
<th>What do you Notice?</th>
<th>What do you Wonder?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Reverse Engineer a Video Game

<table>
<thead>
<tr>
<th>Thing in the Game</th>
<th>What Changes About It?</th>
<th>More Specifically?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td>Lower Limit</td>
<td>Upper Limit</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>

( Inspired by @MrStadel's Estimation180.com )
Coordinates & Estimation
<table>
<thead>
<tr>
<th>What do you Notice?</th>
<th>What do you Wonder?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Estimating Coordinates

The coordinates for the PLAYER (NinjaCat) are: ( , )

The coordinates for the DANGER (Dog) are: ( , )

The coordinates for the TARGET (Ruby) are: ( , )
Brainstorm Your Own Game

Created by: ____________________________

Background

Our game takes place: ____________________________

In space? The desert? A mall?

Player

The Player is a ____________________________
The Player moves only up and down.

Target

Your Player GAINS points when hit The Target.
The Target is a ____________________________
The Target moves only to the left or right.

Danger

Your Player LOSES points when they hit The Danger.
The Danger is a ____________________________
The Danger moves only to the left or right.

Artwork/Sketches/Proof of Concept
Order of Operations
(Circles of Evaluation)
Translate the Circles of Evaluation into code. Use the space below each Circle of Evaluation to create your own challenging Circle of Evaluation!

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>Circle of Evaluation</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(3 \times 7) - (1 + 2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$3 - (1 + 2)$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$3 - (1 + (5 \times 6))$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(1 + (5 \times 6)) - 3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Translating Circles of Evaluation to Code (2)

Translate the Circles of Evaluation into code. Use the space below each Circle of Evaluation to create your own challenging Circle of Evaluation!
## Circles of Evaluation — Square Roots

Translate the Circles of Evaluation into code. Use the space below each Circle of Evaluation to create your own challenging Circle of Evaluation!

<table>
<thead>
<tr>
<th>Math Expression</th>
<th>Circle of Evaluation</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\sqrt{9}$</td>
<td></td>
<td><code>sqrt(9)</code></td>
</tr>
<tr>
<td>$\sqrt{5} + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\sqrt{4} + 1$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$3\sqrt{3} + \sqrt{7}$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Domain & Range
(Contracts)
Exploring Image Functions

With your partner, discover as many Image-producing functions as you can!

Use the space below to draw the Circles of Evaluation for the new functions. You can also sketch a picture of what the function produces, as shown.

<table>
<thead>
<tr>
<th>star</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
</tr>
<tr>
<td>&quot;solid&quot; &quot;block&quot;</td>
</tr>
</tbody>
</table>

produces →

Mystery Functions!

There is a function called text. Try to figure out how to use it!

There is a function called radial-star with 5 inputs. What do they stand for?
### Reading for Domain and Range

1) What is the **name** of the function being used in:

```plaintext
string-length(zwsp)(zwsp)'broccoli'(zwsp) {plus} #
```

2) What is the **domain** of the outermost function being used in:

```plaintext
scale(zwsp)(zwsp)2, circle(zwsp)(zwsp)40, "solid", "blue"(zwsp)(zwsp)
```

3) What is the **domain** of the innermost function being used in:

```plaintext
scale(zwsp)(zwsp)2, circle(zwsp)(zwsp)40, "solid", "blue"(zwsp)(zwsp)
```

4) How many **arguments** does the `+` operator take in:

```plaintext
string-length(zwsp)(zwsp)'broccoli'(zwsp) {plus} #
```

5) What is the **range** of the function `string-length`?

6) Is `text` a **String**, `a function`, or an **Image**?

7) Is the **range** of `text` a **String** or an **Image**?

8) What is the first **argument** to the `circle` function in:

```plaintext
scale(zwsp)(zwsp)2, circle(zwsp)(zwsp)40, "solid", "blue"(zwsp)(zwsp)
```
Function Composition 1
You’ll be investigating these functions with your partner:

```scheme
# text :: String, Number, String -> Image
# scale :: Number, Image -> Image
# rotate :: Number, Image -> Image
# flip-horizontal :: Image -> Image
# flip-vertical :: Image -> Image
```

1) Make an image of your name, in big purple letters. Draw the Circle of Evaluation and write the code that will create this image.

Try using the `scale` function to make your name bigger or smaller. Draw the Circle of Evaluation (hint: use what you wrote above!), then write the code.

In your own words, what does `scale` do?

Try out `rotate`, `flip-horizontal`, and `flip-vertical`. Use the space below to write your code, then test out your code in WeScheme when you’re ready.
Draw a Circle of Evaluation and write the code for a **solid, green star, size 50**.

Using the Circle of Evaluation and code from above, draw a Circle of Evaluation and write the code for each of the exercises below.

<table>
<thead>
<tr>
<th>A solid, green star that is 3 times the size of the original (using scale)</th>
<th>A solid, green star is half the size of the original (using scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A solid, green star of size 50 that has been rotated 45 degrees</td>
<td>A solid, green star that is 3 times the size of the original and has been rotated 45 degrees</td>
</tr>
</tbody>
</table>
Defining Values
Defining Values — Explore!

```
shape1 = triangle(50, "solid", "red")
```

Type the line of code above into the Definitions Area of a new program, and press “Run”.

What happens when you enter `shape1` into the Interactions Area?

Brainstorm some other values to define. Use the space below to draw any Circles of Evaluation you need and to organize your thoughts.

**Ideas:** eye-color (a String), age (a Number), fav-shape (an Image)
Defining Values — Practice

Write the code to define PRIZE_STAR as a solid, green star of size 50.

Using the PRIZE_STAR definition from above, draw the Circle of Evaluation and write the code for each of the exercises. One Circle of Evaluation has been done for you.

<table>
<thead>
<tr>
<th>A solid, green star that is 3 times the size of the original (using scale)</th>
<th>A solid, green star is half the size of the original (using scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>scale 3 \text{ PRIZE_STAR}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A solid, green star of size 50 that has been rotated 45 degrees</th>
<th>A solid, green star that is 3 times the size of the original and has been rotated 45 degrees</th>
</tr>
</thead>
</table>

How does defining values help you as a programmer?
<table>
<thead>
<tr>
<th>What do you Notice?</th>
<th>What do you Wonder?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Defining Functions 1
Fast Functions

# :: ->

examples:

________ (________) is ________________

________ (________) is ________________

end

fun __________(__________):


eu examples:

________ (________) is ________________

________ (________) is ________________

end

fun __________(__________):


eu examples:

________ (________) is ________________

________ (________) is ________________

end

fun __________(__________):


end


end
## Mapping Examples with Circles of Evaluation

**Contract:**

**Purpose Statement:**

<table>
<thead>
<tr>
<th>If I type...</th>
<th>→</th>
<th>It should map to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE #1: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
<tr>
<td>Code:</td>
<td></td>
<td>Code:</td>
</tr>
<tr>
<td>EXAMPLE #2: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
<tr>
<td>Code:</td>
<td></td>
<td>Code:</td>
</tr>
</tbody>
</table>
## Mapping Examples with Circles of Evaluation

<table>
<thead>
<tr>
<th>Purpose Statement</th>
<th>Code:</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE #1: Circle of Evaluation:</td>
<td>Code:</td>
</tr>
<tr>
<td>EXAMPLE #2: Circle of Evaluation:</td>
<td>Code:</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>It should map to...</th>
<th>Code:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Up</th>
<th>Code:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code:</td>
<td></td>
</tr>
</tbody>
</table>
Defining Functions 2
<table>
<thead>
<tr>
<th>What do you Notice?</th>
<th>What do you Wonder?</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Word Problem: rocket-height

Directions: A rocket blasts off, traveling at 7 meters per second. Use the Design Recipe to write a function 'rocket-height', which takes in a number of seconds and calculates the height.

Contract and Purpose Statement

Every contract has three parts…

# _______________; _______________ -> _______________

# _______________

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Don't care
Defining Functions 3
(Word Problems & Composite Functions)
# Solving Word Problems with the Design Recipe

Use The Design Recipe to create models for the word problems and write the appropriate functions.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1)</strong> Write a function <code>globo-gym</code> that takes in a number of months and produces the cost for attending the gym that many months when there is a $150 initiation fee and a monthly fee of $45/month.</td>
<td><strong>2)</strong> Write a function <code>rect-perimeter</code> that takes in the length and width of a rectangle and returns the perimeter of that rectangle.</td>
</tr>
<tr>
<td><strong>3)</strong> Write a function <code>lawn-area</code> that takes in a length and width of a lawn and returns the area of that rectangular lawn.</td>
<td><strong>4)</strong> Write a function <code>rectprism-vol</code> that takes in the length, width, and height of a rectangular prism and returns the Volume of a rectangular prism.</td>
</tr>
<tr>
<td><strong>5)</strong> Write a function <code>rideshare</code>, that takes in a number of miles and produces the cost of a ride for that many miles at $2.50 plus $1.50/mile.</td>
<td><strong>6)</strong> Write a function <code>marquee</code> that takes in a message and returns that message in large gold letters.</td>
</tr>
<tr>
<td><strong>7)</strong> Write a function <code>split-tab</code> that takes in a cost and the number of people sharing the bill and splits the cost equally.</td>
<td><strong>8)</strong> Write a function <code>num-cube</code> that takes in a number and returns the cube of that number.</td>
</tr>
<tr>
<td><strong>9)</strong> Write a function <code>circle-area</code> that takes in a radius and returns the area of the circle.</td>
<td><strong>10)</strong> Write a function <code>tip-calculator</code> that takes in the cost of a meal and returns the 15% tip of that meal.</td>
</tr>
<tr>
<td><strong>11)</strong> Write a function <code>minimum-wage</code>, that takes in a number of hours worked and returns the amount a worker will get paid at $10.25/hr.</td>
<td><strong>12)</strong> Write a function <code>moving</code> that takes in the days and number of miles driven and returns the cost of renting a truck. The truck is $55 per day and each driven mile is 15¢.</td>
</tr>
</tbody>
</table>
Making Sense of Word Problems & Writing Quality Purpose Statements

1st Read: What is this problem about?

2nd Read: What are the Quantities?

3rd Read: What is a good Purpose Statement?

Stronger & Clearer

Purpose Statement 1st Revision:

Purpose Statement 2nd Revision:
## Mapping Examples with Circles of Evaluation

<table>
<thead>
<tr>
<th>If I type...</th>
<th>→</th>
<th>It should map to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE #1: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
<tr>
<td>Code:</td>
<td>→</td>
<td>Code:</td>
</tr>
<tr>
<td>EXAMPLE #2: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
<tr>
<td>Code:</td>
<td>→</td>
<td>Code:</td>
</tr>
</tbody>
</table>
### Making Sense of Word Problems & Writing Quality Purpose Statements

#### with 3 Reads + Stronger & Clearer

<table>
<thead>
<tr>
<th>1st Read: What is this problem about?</th>
</tr>
</thead>
<tbody>
<tr>
<td>2nd Read: What are the Quantities?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3rd Read: What is a good Purpose Statement?</th>
</tr>
</thead>
</table>

### Stronger & Clearer

Purpose Statement 1st Revision:

Purpose Statement 2nd Revision:
## Mapping Examples with Circles of Evaluation

**Contract:**

**Purpose Statement:**

<table>
<thead>
<tr>
<th>If I type...</th>
<th>→</th>
<th>It should map to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE #1: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
<tr>
<td>Code:</td>
<td>Code:</td>
<td></td>
</tr>
</tbody>
</table>

| EXAMPLE #2: Circle of Evaluation | → | Circle of Evaluation: |
| Code: | Code: |
### Making Sense of Word Problems & Writing Quality Purpose Statements

1st Read: What is this problem about?

2nd Read: What are the Quantities?

3rd Read: What is a good Purpose Statement?

---

### Stronger & Clearer

Purpose Statement 1st Revision:

Purpose Statement 2nd Revision:
### Mapping Examples with Circles of Evaluation

**Contract:**

**Purpose Statement:**

<table>
<thead>
<tr>
<th>If I type...</th>
<th>→</th>
<th>It should map to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXAMPLE #1: Circle of Evaluation</td>
<td>→</td>
<td>Circle of Evaluation:</td>
</tr>
</tbody>
</table>

Code:

| EXAMPLE #2: Circle of Evaluation | → | Circle of Evaluation: |

Code:
Function Applications 2
(Animation with Functions)
Directions: Use the Design Recipe to write a function ‘update-danger’, which takes in the danger’s x-coordinate and produces the next x-coordinate.

Contract and Purpose Statement

Every contract has three parts…

# function name :: domain -> range
# what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

Definition

Write the definition, giving variable names to all your input values…

fun function name(input(s)):

variable(s)

end
Directions: Use the Design Recipe to write a function `update-target`, which takes in the danger’s x-coordinate and produces the next x-coordinate.

### Contract and Purpose Statement

Every contract has three parts…

# function name :: domain -> range

# what does the function do?

### Examples

Write some examples, then circle and label what changes…

examples:

- function name (input(s)) is what the function produces
- function name (input(s)) is what the function produces

end

### Definition

Write the definition, giving variable names to all your input values…

fun function name (variable(s)):

what the function does with those variable(s)

end
Function Applications 3
Word Problem: revenue

Directions: Use the Design Recipe to write a function ‘revenue’, which takes in the number of glasses sold at $1.75 apiece and calculates the total revenue.

Contract and Purpose Statement
Every contract has three parts…

# _______________ :: ____________________________ --> ______________________
#
# _______________ :: ____________________________
# what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

________________ (________________) is ____________________________

function name    input(s)    input(s)    what the function produces

________________ (________________) is ____________________________

function name    input(s)    input(s)    what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________________(________________):

________________    __________________

function name    variable(s)

what the function does with those variable(s)

end
Directions: Use the Design Recipe to write a function 'cost', which takes in the number of glasses sold and calculates the total cost of materials if each glass costs $0.30 to make.

Contract and Purpose Statement

Every contract has three parts…

# ____________ :: ______________ -> _______________
  function name               domain                      range

# _______________
what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

__________________________ (__________________) is ________________
  function name           input(s)          what the function produces

__________________________ (__________________) is ________________
  function name           input(s)          what the function produces

end

Definition

Write the definition, giving variable names to all your input values…

fun ______________________ (_________________):

__________________________ (__________________):
  function name           variable(s)

what the function does with those variable(s)

end
Word Problem: profit

Directions: Use the Design Recipe to write a function 'profit' that calculates Sally's total profit.

Contract and Purpose Statement
Every contract has three parts…

# __________________ : __________________ --> __________________
    function name       domain               range
# __________________

what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces
function name (input(s)) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________________ (___________):
    function name variable(s)

what the function does with those variable(s)

end
Inequalities — Launch

What would each of the following expressions evaluate to? Write your guesses in the space provided, and then take turns typing them into the computer.

<table>
<thead>
<tr>
<th>Expression</th>
<th>Guess</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 + 4</td>
<td>5</td>
</tr>
<tr>
<td>0 &gt; 5</td>
<td>true</td>
</tr>
<tr>
<td>4 / 2</td>
<td>2</td>
</tr>
<tr>
<td>1 = 9</td>
<td>false</td>
</tr>
<tr>
<td>0 - 9</td>
<td>-9</td>
</tr>
<tr>
<td>2 &lt;= 2</td>
<td>true</td>
</tr>
<tr>
<td>string-length(&quot;bat&quot;)</td>
<td>3</td>
</tr>
<tr>
<td>string-equal(&quot;dog&quot;, &quot;cat&quot;)</td>
<td>false</td>
</tr>
</tbody>
</table>

1) What does the function < do?

2) What does the function string-equal() do?

3) Write the contract for these Boolean functions in your Contracts page.

4) How many Numbers are there in the entire universe? ____________

5) How many Strings are there in the entire universe? ____________

6) How many Images are there in the entire universe? ____________

7) How many Booleans are there in the Universe? ________
   What are they?
Sam the Butterfly

Open the “Sam the Butterfly” starter file and press “Run”. Hi, Sam!

Sam is in a 640 × 480 yard. Sam's mom wants Sam to stay in sight.

How far to the left and right can Sam go and still remain visible?

Use the new inequality functions to answer the following questions with code:

Sam is still visible on the left as long as…

Sam is still visible on the right as long as…

Use the space below to draw Circles of Evaluation for these two expressions:
Word Problem: is-safe-left

Directions: Use the Design Recipe to write a function ‘is-safe-left’, which takes in an x-coordinate and checks to see if it is greater than -50

Contract and Purpose Statement
Every contract has three parts…

# _______________ : ____________________ -> _______________

# function name: domain -> range

Examples
Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun ______________ (___________):

function name variable(s)

end
Directions: Use the Design Recipe to write a function ‘is-safe-right’, which takes in an x-coordinate and checks to see if it is less than 690

Contract and Purpose Statement
Every contract has three parts…

# __________ ; : _______________ -> __________

# ______________

Examples
Write some examples, then circle and label what changes…

examples:

____________________ (__________) is __________

____________________ (__________) is __________

end

Definition
Write the definition, giving variable names to all your input values…

fun ______________(__________):

____________________

end
Inequalities — Practice

Create the Circles of Evaluation, then convert the expressions into code in the space provided.

1) 2 is less than 5, and 0 is equal to 6

What will this evaluate to? ________________

2) 6 is greater than 8, or -4 is less than 1

What will this evaluate to? ________________

3) The String “purple” is the same as the String “blue”, and 3 plus 5 equals 8

What will this evaluate to? ________________

4) Write the contracts for and and or in your Contracts page.
Word Problem: is-onscreen

Directions: Use the Design Recipe to write a function ‘is-onscreen’, which takes in an x-coordinate and checks to see if Sam is safe on the left AND safe on the right

Contract and Purpose Statement

Every contract has three parts…

```
# function name :: domain -> range
#
```

Examples

Write some examples, then circle and label what changes…

```
examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end
```

Definition

Write the definition, giving variable names to all your input values…

```
fun function name (variable(s)):

function name variable(s) what the function does with those variable(s)

end
```
Piecewise Functions 1
Welcome to Alice’s Restaurant!

Alice has hired you to improve some code used at the restaurant. The code we’ll be improving on is shown below.

Read through the code line-by-line with your partner before writing down your observations in the tables below.

cost :: String -> Number
# given a item, produce the cost of that item
fun cost(item):
  ask:
    | string-equal(item, 'hamburger') then: 6.00
    | string-equal(item, 'onion rings') then: 3.50
    | string-equal(item, 'fried tofu') then: 5.25
    | string-equal(item, 'pie') then: 2.25
    | otherwise: "Sorry, that's not on the menu!"
  end
end

- I notice...
- I wonder...

- Familiar things I see in the code
- Unfamiliar things I see in the code
Alice’s Restaurant - Explore

Alice’s code has some new elements we haven’t seen before, so let’s experiment a bit to figure out how it works! **Open the “Alice’s Restaurant starter file, click “Run”, and try using the `cost` function in the Interactions window.**

1) What does `cost("hamburger")` evaluate to?

2) What does `cost("pie")` evaluate to?

3) What if you ask for `cost("fries")`?

4) Explain what the function is doing in your own words.

5) What is the function’s name? Domain? Range?

6) What is the name of its variable?

7) Alice says onion rings have gone up to $3.75. Change the `cost` function to reflect this.

8) Try adding menu items of your own. What’s your favorite?

9) For an unknown food item, the function produces the String “That’s not on the menu!”

Is this a problem? Why or why not?

10) Suppose Alice wants to calculate the price of a hamburger, *including a 5% sales tax*. Draw a Circle of Evaluation for the expression below.
Directions: Alice’s Restaurant has hired you as a programmer. They offer the following menu items: hamburger ($6.00), onion rings ($3.50), fried tofu ($5.25) and pie ($2.25). Write a function called cost which takes in the name of a menu item and outputs the price of that item.

Contract and Purpose Statement

Every contract has three parts...

<table>
<thead>
<tr>
<th>function name</th>
<th>domain</th>
<th>range</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Examples

Write some examples, then circle and label what changes...

examples:

function name (input(s)) is what the function produces

end

Definition

Write the definition, giving variable names to all your input values...

fun function name (variable(s)):

ask:

<table>
<thead>
<tr>
<th>then:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
end

end
Word Problem: update-player

Directions: The player moves up and down by 20 pixels each time. Write a function called update-player, which takes in the player’s y-coordinate and the name of the key pressed (“up” or “down”), and returns the new y-coordinate.

Contract and Purpose Statement
Every contract has three parts…
# __________ : __________________ --> __________
  function name          domain          range

Examples
Write some examples, then circle and label what changes…

examples:
________________________ ( __________ ) is ___________________
  function name          input(s)        what the function produces
________________________ ( __________ ) is ___________________
  function name          input(s)        what the function produces
________________________ ( __________ ) is ___________________
  function name          input(s)        what the function produces
________________________ ( __________ ) is ___________________
  function name          input(s)        what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________________( __________ ):
  function name          variable(s)

  ask:
  | ____________________________________ then: ____________________________
  | ____________________________________ then: ____________________________
  | ________________________________ otherwise: ________________________

end

end
Challenges for Update Player

For each of the challenges below, see if you can come up with two EXAMPLEs of how it should work!

1) **Warping** - program one key to "warp" the player to a set location, such as the center of the screen

   examples:
   ```
   update-player( ) is
   update-player( ) is
   end
   ```

2) **Boundaries** - change update-player such that PLAYER cannot move off the top or bottom of the screen

   examples:
   ```
   update-player( ) is
   update-player( ) is
   end
   ```

3) **Wrapping** - add code to update-player such that when PLAYER moves to the top of the screen, it reappears at the bottom, and vice versa

   examples:
   ```
   update-player( ) is
   update-player( ) is
   end
   ```

4) **Hiding** - add a key that will make PLAYER seem to disappear, and reappear when the same key is pressed again

   examples:
   ```
   update-player( ) is
   update-player( ) is
   end
   ```
The Distance Formula
| What do you Notice? | What do you Wonder? |
Distance formula

Distance between 
(0, 0) and (3, 4)
Directions: Use the Design Recipe to write a function 'distance', which takes in the x and y coordinates of two objects and produces the distance between them in pixels.

Contract and Purpose Statement

Every contract has three parts…

# function name : domain -> range

# what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s))

is what the function produces

end

Definition

Write the definition, giving variable names to all your input values…

fun function name (variable(s)):

what the function does with those variables
A retractable flag pole starts out 24 inches tall, and can grow at a rate of 0.6in/sec. An is elastic tied to the top of the pole and anchored 200 inches from the base, forming a right triangle. Define functions that compute the height of the pole and the area of the triangle after a given number of seconds.

```bash
# :: ->
#
examples:
    _______ (_______) is ________________
    _______ (_______) is ________________
end
Don't care
```

```bash
# :: ->
#
examples:
    _______ (_______) is ________________
    _______ (_______) is ________________
end
Don't care
```
Directions: Use the Design Recipe to write a function ‘is-collision’, which takes in the coordinates of two objects and checks if they are close enough to collide

Every contract has three parts…

# function name : domain -> range

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Definition

Write the definition, giving variable names to all your input values…

fun function name (variable(s)):

what the function does with those variable(s)

end
Design Recipe

Directions:

Contract and Purpose Statement
Every contract has three parts…
# function name :: domain -> range
# what does the function do?

Examples
Write some examples, then circle and label what changes…
examples:

    function name ( input(s) ) is what the function produces
    function name ( input(s) ) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun ( variable(s) ):

    function name variable(s)
    what the function does with those variable(s)

end

Directions:

Contract and Purpose Statement
Every contract has three parts…
# function name :: domain -> range
# what does the function do?

Examples
Write some examples, then circle and label what changes…
examples:

    function name ( input(s) ) is what the function produces
    function name ( input(s) ) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun ( variable(s) ):

    function name variable(s)
    what the function does with those variable(s)

end
Design Recipe

Directions:

Contract and Purpose Statement
Every contract has three parts…

# ___________________ :: ___________________ --> ___________________
  function name       domain           range

# ___________________ --> ___________________
  what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

__________ (__________) is ________________
  function name        input(s)           what the function produces

__________ (__________) is ________________
  function name        input(s)           what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________ (___________):
  function name        variable(s)

__________
  what the function does with those variable(s)

end
Design Recipe

Directions:

Contract and Purpose Statement
Every contract has three parts…

# function name :: domain -> range

# what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________(___________):

function name variable(s)

what the function does with those variable(s)

end

Directions:

Contract and Purpose Statement
Every contract has three parts…

# function name :: domain -> range

# what does the function do?

Examples
Write some examples, then circle and label what changes…

examples:

function name (input(s)) is what the function produces

function name (input(s)) is what the function produces

end

Definition
Write the definition, giving variable names to all your input values…

fun __________(___________):

function name variable(s)

what the function does with those variable(s)

end
Contracts tell us how to use a function. For example: \texttt{num-sqr :: (Number) -> Number} tells us that the name of the function is \texttt{num-sqr}, it takes one input (a \texttt{Number}), and it evaluates to a \texttt{Number}. From the contract, we know \texttt{num-sqr(4)} will evaluate to a \texttt{Number}.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
</tbody>
</table>
Contracts

Contracts tell us how to use a function. For example: `num-sqr :: (Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
</tbody>
</table>
Contracts

Contracts tell us how to use a function. For example: `num-sqr :: (Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>- &gt;</td>
</tr>
</tbody>
</table>
Contracts tell us how to use a function. For example: `num-sqr :: (Number) -> Number` tells us that the name of the function is `num-sqr`, it takes one input (a `Number`), and it evaluates to a `Number`. From the contract, we know `num-sqr(4)` will evaluate to a `Number`.

<table>
<thead>
<tr>
<th>Name</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
<tr>
<td>#</td>
<td>::</td>
<td>-&gt;</td>
</tr>
</tbody>
</table>