Name: _____

Reactive

Fall 2025 Student Workbook - Pyret Edition

BOOTSTRAP

Equity • Scale • Rigor

Workbook v3.1

Brought to you by the Bootstrap team:

- Emmanuel Schanzer
- Kathi Fisler
- Shriram Krishnamurthi
- Dorai Sitaram
- Joe Politz
- Ben Lerner
- Nancy Pfenning
- Flannery Denny
- Rachel Tabak

Table of Contents

Computing Needs All Voices	1
Simple Data Types	5
Contracts for Strings and Images	8
Function Composition	17
Defining Values	21
Transforming and Composing Images	27
Functions Make Life Easier!	29
Functions: Contracts, Examples & Definitions	34
Solving Word Problems with the Design Recipe	40
Introduction to Data Structures	45
Structures, Reactors, and Animations	52
Functions That Ask Questions	65
Key Events	71
Refactoring	78
Your Own Drawing Functions	79
Build Your Own Animation	80
Distance in Video Games	83
Adding Collisions	92
Scoring	96
Adding Levels	97
Making Pong	97
Going Deeper: Nested Structures	98
Feature: Timers	99

Pioneers in Computing and Mathematics

The pioneers pictured below are featured in our Computing Needs All Voices lesson. To learn more about them and their contributions, visit https://bit.ly/bootstrap-pioneers.

We are in the process of expanding our collection of pioneers. If there's someone else whose work inspires you, please let us know at https://bit.ly/pioneer-suggestion.

Notice and Wonder

Write down what you Notice and Wonder from the What Most Schools Don't Teach video.
"Notices" should be statements, not questions. What stood out to you? What do you remember? "Wonders" are questions.

What do you Notice?	What do you Wonder?

Windows and Mirrors

1) Think about the stories you've just encountered. Identify something(s) from the film and/or posters that served as a mirror for you, connecting you with your own identity and experience of the world. Write about who or what you connected with and why.

2) Identify something(s) from the film or the posters that served as a window for you, giving you insight into other people's experiences or expanding your thinking in some way.

Reflection: Try Thinking About Ketchup

This reflection is designed to follow reading LA Times Perspective: A solution to tech's lingering diversity problem? Try thinking about ketchup

1) Think of a time when someone else had a strategy or idea that you would never have thought of, but was interesting to you and/or pushed your thinking to a new level.

2) Think of a time when you had an idea that felt "out of the box". Did you share your idea? Why or why not?

3) The author argues that tech companies with diverse teams have an advantage. Why?

4) What suggestions did the article offer for tech companies looking to diversify their teams?

5) What is one thing of interest to you in the author's bio?

6) Based on your experience of exceptions to mainstream assumptions, propose another pair of questions that could be used in place of "Where do you keep your ketchup?" and "What would you reach for instead?"

Perspective: A solution to tech's lingering diversity problem? Try thinking about ketchup

By Dexter Thomas • Published March 16, 2016 6:24 PM PT in the Los Angeles Times

Diversity is a hot, and controversial, topic in Silicon Valley. But why do so many people care about it?

At first glance, the answer may seem simple: Improving minorities' access to tech jobs is the right thing to do.

But when I moderated a panel Monday at SXSW on diversity in the tech industry, I was surprised none of the panelists talked much about what was "right."

Instead, they talked about what was right for business.

Sarah Wagener, vice president of talent acquisition and diversity at Pandora, agreed during the panel that pushing to hire more diverse candidates is the "right thing" to do.

"But," she said, "it's been the 'right thing to do' for a long time, and we're still having this conversation."
If you're trying to make the case at your company for diversifying your workforce, she said, your argument needs to be focused on "real business outcomes."

In other words, recruiting people from underrepresented backgrounds should be understood not as an obligation that could lower the bar and weigh your company down, but as an opportunity that could raise the bar, and lift your company above the competition.

Instantly, Wagener's statements reminded me of ketchup.

If you haven't heard it yet, the "ketchup question" is a thought experiment that's become something of a meme in some corners of the tech community thanks to a popular episode of the Reply All podcast. It starts as an innocent question:

Where do you keep your ketchup?

If you're like most people in the United States, odds are that you keep your ketchup in the refrigerator. But depending on where you grew up, you might keep it in the cupboard.

Imagine that you reach for the ketchup bottle and find it empty. You need a substitute sauce, and grab whatever is nearby. If that bottle is in the refrigerator, you may opt for mayo. But if it's in the cupboard, the seasoning closest at hand might be malt vinegar, or Tabasco, or salt and pepper.

Start-up culture is often centered around new ways of solving "problems" — ride-sharing apps such as Lyft and Uber solve the problem of getting around town without a car, for example. The "ketchup question" shows how a slight difference in perspective can lead a coworker toward a completely different solution that might never occur to you. That extra perspective could lead to a fresh new idea that could take your company to the top.

But without a diverse team? It's gonna be mayo every time.

What do we do about it?

Most people aren't chief executives of a major company, and may feel like they have no sway in the hiring process.
So I asked two of the panelists to give some suggestions that could be useful for employees of all levels, regardless of the industry in which they work.

Karla Monterroso, vice president of programs at Code 2040, an organization that works to place black and Latino students in engineering internships at tech companies, said that job listings could be an unexpected barrier to attracting diverse talent.
Using seemingly innocent words like "hacker" or "rockstar" in job listings could unintentionally give the impression to some women that the company would not be a hospitable place to work, said Monterroso. She recommended reading articles on the topic of bias and having informal conversations with coworkers.

More directly, she said, using these articles as "evidence" to suggest small changes in recruitment practices could be an easy first step in attracting new talent.

James Talbot, a software engineer at San Francisco web publishing startup Medium, was concerned with what happens after a new recruit is hired. He suggested using social media to follow people who have different perspectives than you, for 30 days. The key, he said, is to listen to what they have to say, simply exposing yourself to their conversations — not commenting or arguing with them.
This is important, he said, because even after a recruiter hires a person from an underrepresented community, adapting to the workplace environment can be another challenge. If people get into a job but have to deal with racist or sexist comments and insensitive treatment, they may simply leave – and take their unique perspectives and talent elsewhere.
People often say that the cause of the lack of diversity in many tech companies is the lack of an easy way to find available candidates.

"People always give excuses, saying the problem is the 'pipeline,'" Talbot said.

"But who wants to be on a pipeline into a sewer?"

Dexter Thomas is from San Bernardino and is a PhD candidate in East Asian studies at Cornell University. He has taught media studies and Japanese and is writing a book about Japanese hip-hop. Thomas began working in new media as a student director of programming at KUCR-FM (88.3), independently producing podcasts as well as music and news programs. He has written for several outlets internationally on topics as diverse as Internet and youth culture, social justice and video games. He left The Times in 2016.

(handout)

Introduction to Programming in a Nutshell

The **Editor** is a software program we use to write Code. Our Editor allows us to experiment with Code on the right-hand side, in the **Interactions Area**. For Code that we want to *keep*, we can put it on the left-hand side in the **Definitions Area**. Clicking the "Run" button causes the computer to re-read everything in the Definitions Area and erase anything that was typed into the Interactions Area.

Data Types

Programming languages involve different ***data types***, such as Numbers, Strings, Booleans, and even Images.

- Numbers are values like `1`, `0.4`, `1/3`, and `-8261.003`.
  - Numbers are *usually* used for quantitative data and other values are *usually* used as categorical data.
  - In Pyret, decimals *must* start with a zero. For example, `0.22` is valid, but `.22` is not.
- Strings are values like `"Emma"`, `"Rosanna"`, `"Jen and Ed"`, or even `"08/28/1980"`.
  - All strings *must* be surrounded by quotation marks.
- Booleans are either `true` or `false`.

All values evaluate to themselves. The program `42` will evaluate to `42`, the String `"Hello"` will evaluate to `"Hello"`, and the Boolean `false` will evaluate to `false`.

Operators

Operators (like `+`, `-`, `*`, `<`, etc.) work the same way in Pyret that they do in math.

- Operators are written between values, for example: `4 + 2`.
- In Pyret, operators must always have spaces around them. `4 + 2` is valid, but `4+2` is not.
- If an expression has different operators, parentheses must be used to show order of operations. `4 + 2 + 6` and `4 + (2 * 6)` are valid, but `4 + 2 * 6` is not.

Applying Functions

Functions work much the way they do in math. Every function has a name, takes some inputs, and produces some output. The function name is written first, followed by a list of ***arguments*** in parentheses.

- In math this could look like $f(5)$ or $g(10, 4)$.
- In Pyret, these examples would be written as `f(5)` and `g(10, 4)`.
- Applying a function to make images would look like `star(50, "solid", "red")`.
- There are many other functions in Pyret, for example `sqr`, `sqrt`, `triangle`, `square`, `string-repeat`, etc.

Functions have ***contracts***, which help explain how a function should be used. Every Contract has three parts:

- The *Name* of the function - literally, what it's called.
- The *Domain* of the function - what *type(s) of value(s)* the function consumes, and in what order.
- The *Range* of the function - what *type of value* the function produces.

Strings and Numbers

*Make sure you've loaded code.pyret.org (CPO), clicked "Run", and are working in the **Interactions Area** on the right. Hit Enter/return to evaluate expressions you test out.*

Strings

String values are always in quotes.

- Try typing your name *(in quotes!)*.
- Try typing a sentence like "I'm excited to learn to code!" *(in quotes!)*.
- Try typing your name with the opening quote, but *without the closing quote.* Read the error message!
- Now try typing your name *without any quotes.* Read the error message!

1) Explain what you understand about how strings work in this programming language. _____

_____

Numbers

2) Try typing `42` into the Interactions Area and hitting "Enter". Is `42` the same as `"42"`? Why or why not?

_____

3) What is the largest number the editor can handle?

_____

4) Try typing `0.5`. Then try typing `.5`. Then try clicking on the answer. Experiment with other decimals.

Explain what you understand about how decimals work in this programming language. _____

_____

5) What happens if you try a fraction like `1/3`? _____

_____

6) Try writing **negative** integers, fractions and decimals. What do you learn? _____

_____

_____

Operators

7) Just like math, Pyret has ***operators*** like `+`, `-`, `*` and `/`.
Try typing in `4 + 2` and then `4+2` (without the spaces). What can you conclude from this?

_____

8) Type in the following expressions, **one at a time**: `4 + 2 * 6` `(4 + 2) * 6` `4 + (2 * 6)` What do you notice?

_____

9) Try typing in `4 + "cat"`, and then `"dog" + "cat"`. What can you conclude from this?

_____

_____

Booleans

Boolean-producing expressions are yes-or-no questions, and will always evaluate to either `true` ("yes") or `false` ("no").
What will the expressions below evaluate to? *Write down your prediction, then type the code into the Interactions Area to see what it returns.*

	Prediction	Result		Prediction	Result
1) `3 <= 4`			2) `"a" > "b"`		
3) `3 == 2`			4) `"a" < "b"`		
5) `2 < 4`			6) `"a" == "b"`		
7) `5 >= 5`			8) `"a" <> "a"`		
9) `4 >= 6`			10) `"a" >= "a"`		
11) `3 <> 3`			12) `"a" <> "b"`		
13) `4 <> 3`			14) `"a" >= "b"`		

15) In your own words, describe what `<` does. _____

_____

16) In your own words, describe what `>=` does. _____

_____

17) In your own words, describe what `<>` does. _____

_____

	Prediction:	Result:
18) `string-contains("catnap", "cat")`		
19) `string-contains("cat", "catnap")`		

20) In your own words, describe what `string-contains` does. Can you generate another expression using `string-contains` that returns true?

_____

★ There are infinite string values ("a", "aa", "aaa"…) and infinite number values out there (…-2,-1,0,-1,2…). But how many different *Boolean* values are there? _____

Applying Functions

Open code.pyret.org (CPO) and click "Run". We will be working in the Interactions Area on the right.

Test out these two expressions and record what you learn below:

- `regular-polygon(40, 6, "solid", "green")`
- `regular-polygon(80, 5, "outline", "dark-green")`

1) You've seen data types like Numbers, Strings, and Booleans. What data type did the `regular-polygon` function produce? _____

2) How would you describe what a regular polygon is? _____

_____

3) The `regular-polygon` function takes in four pieces of information (called arguments). Record what you know about them below.

	Data Type	Information it Contains
Argument 1		
Argument 2		
Argument 3		
Argument 4		

There are many other functions available to us in Pyret. We can describe them using ***contracts***. The Contract for `regular-polygon` is:

```
# regular-polygon :: Number, Number, String, String -> Image
```

- Each Contract begins with the function name: *in this case* `regular-polygon`
- Lists the data types required to satisfy its Domain: *in this case* Number, Number, String, String
- And then declares the data type of the Range it will return: *in this case* Image

Contracts can also be written with more detail, by annotating the Domain with *variable names*:

```
# regular-polygon :: ( Number , Number , String , String ) -> Image
                     size     number-of-sides  fill-style  color
```

4) We know that a square is a regular polygon because _____

_____

5) What code would you write to make a big, blue square using the `regular-polygon` function?

_____ (_____ , _____ , _____ , _____)
function-name — size :: Number — number-of-sides :: Number — fill-style :: String — color :: String

6) Pyret also has a `square` function whose contract is:

```
# square :: ( Number , String , String ) -> Image
             size     fill-style  color
```

What code would you write to make a big blue square using the `square` function?

_____ (_____ , _____ , _____)
function-name — size :: Number — fill-style :: String — color :: String

7) Why does `square` need fewer arguments to make a square than `regular-polygon`? _____

_____

★ Where else have you heard the word ***contract*** used before?

Practicing Contracts: Domain & Range

Note: The contracts on this page are not defined in Pyret and cannot be tested in the editor.

is-beach-weather

Consider the following Contract:

```
# is-beach-weather :: Number, String -> Boolean
```

1) What is the **Name** of this function? _____

2) How many arguments are in this function's **Domain**? _____

3) What is the **Type** of this function's **first argument**? _____

4) What is the **Type** of this function's **second argument**? _____

5) What is the **Range** of this function? _____

6) Circle the expression below that shows the correct application of this function, based on its Contract.

A. `is-beach-weather(70, 90)`
B. `is-beach-weather(80, 100, "cloudy")`
C. `is-beach-weather("sunny", 90)`
D. `is-beach-weather(90, "stormy weather")`

cylinder

Consider the following Contract:

```
# cylinder :: Number, Number, String -> Image
```

7) What is the **Name** of this function? _____

8) How many arguments are in this function's **Domain**? _____

9) What is the **Type** of this function's **first argument**? _____

10) What is the **Type** of this function's **second argument**? _____

11) What is the **Type** of this function's **third argument**? _____

12) What is the **Range** of this function? _____

13) Circle the expression below that shows the correct application of this function, based on its Contract.

A. `cylinder("red", 10, 60)`
B. `cylinder(30, "green")`
C. `cylinder(10, 25, "blue")`
D. `cylinder(14, "orange", 25)`

Matching Expressions and Contracts

Match the Contract (left) with the expression that uses it correctly (right).
Note: The contracts on this page are not defined in Pyret and cannot be tested in the editor.

Contract		Expression
`# make-id :: String, Number -> Image`	1	A `make-id("Savannah", "Lopez", 32)`
`# make-id :: String, Number, String -> Image`	2	B `make-id("Pilar", 17)`
`# make-id :: String -> Image`	3	C `make-id("Akemi", 39, "red")`
`# make-id :: String, String -> Image`	4	D `make-id("Raïssa", "McCracken")`
`# make-id :: String, String, Number -> Image`	5	E `make-id("von Einsiedel")`

Contract		Expression
`# is-capital :: String, String -> Boolean`	6	A `show-pop("Juneau", "AK", 31848)`
`# is-capital :: String, String, String -> Boolean`	7	B `show-pop("San Juan", 395426)`
`# show-pop :: String, Number -> Image`	8	C `is-capital("Accra", "Ghana")`
`# show-pop :: String, String, Number -> Image`	9	D `show-pop(3751351, "Oklahoma")`
`# show-pop :: Number, String -> Number`	10	E `is-capital("Albany", "NY", "USA")`

Contracts for Image-Producing Functions

Log into [code.pyret.org (CPO)] and click "Run". Experiment with each of the functions listed below in the interactions area. Try to find an expression that produces an image. Record the contract and example code for each function you are able to use!

Name		Domain		Range
`# triangle`	`::`	`Number, String, String`	`->`	`Image`
`triangle(80, "solid", "darkgreen")`				
`# star`	`::`		`->`	
`# circle`	`::`		`->`	
`# rectangle`	`::`		`->`	
`# text`	`::`		`->`	
`# square`	`::`		`->`	
`# rhombus`	`::`		`->`	
`# ellipse`	`::`		`->`	
`# regular-polygon`	`::`		`->`	
`# right-triangle`	`::`		`->`	
`# isosceles-triangle`	`::`		`->`	
`# radial-star`	`::`		`->`	
`# star-polygon`	`::`		`->`	
`# triangle-sas`	`::`		`->`	
`# triangle-asa`	`::`		`->`	

Catching Bugs when Making Triangles

Learning about a Function through Error Messages

1) Type `triangle` into the Interactions Area of code.pyret.org (CPO) and hit "Enter". What do you learn? _____

2) We know that all functions will need an open parenthesis and at least one input! Type `triangle(80)` in the Interactions Area and hit Enter/return. Read the error message. What hint does it give us about how to use this function?

_____

3) Using the hint from the error message, experiment until you can make a triangle. What is the contract for `triangle`?

_____

4) Read the explanation below. Then explain the difference in your own words.

syntax errors - when the computer cannot make sense of the code because of unclosed strings, missing commas or parentheses, etc.
contract errors - when the function isn't given what it needs (the wrong type or number of arguments are used)

The difference between **syntax errors** and **contract errors** is: _____

_____

Finding Mistakes with Error Messages

The following lines of code are all BUGGY! Read the code and the error messages below. See if you can find the mistake WITHOUT typing it into Pyret.

5) `triangle(20, "solid" "red")`

```
Pyret didn't understand your program around
triangle(20, "solid" "red")
```

This is a _____ (contract / syntax) error. The problem is that _____

6) `triangle(20, "solid")`

```
This application expression errored:
triangle( 20 , "solid")
2 arguments were passed to the operator. The operator evaluated to a function accepting 3
parameters. An application expression expects the number of parameters and arguments to be the
same.
```

This is a _____ (contract / syntax) error. The problem is that _____

7) `triangle(20, 10, "solid", "red")`

```
This application expression errored:
triangle(20, 10, "solid", "red")
4 arguments were passed to the operator. The operator evaluated to a function accepting 3
parameters. An application expression expects the number of parameters and arguments to be the
same.
```

This is a _____ (contract / syntax) error. The problem is that _____

8) `triangle (20, "solid", "red")`

```
Pyret thinks this code is probably a function call:
triangle (20, "solid", "red")
Function calls must not have space between the function expression and the arguments.
```

This is a _____ (contract / syntax) error. The problem is that _____

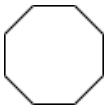
Using Contracts

For questions 1,2,4,5,8 & 9, use the contracts provided to find expressions that will generate images similar to the ones pictured.
Test your code in [code.pyret.org (CPO)] before recording it.

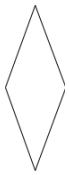
```
# ellipse :: ( Number , Number , String , String ) -> Image
             (width, height, fill-style, color)
```

1)		
2)		
3)	Write an expression using `ellipse` to produce a circle.	

```
# regular-polygon :: ( Number , Number , String , String ) -> Image
                     (side-length, number-of-sides, fill-style, color)
```

4)		
5)		
6)	Use `regular-polygon` to write an expression for a square!	
7)	How would you describe a **regular polygon** to a friend?	

```
# rhombus :: ( Number , Number , String , String ) -> Image
             (size, top-angle, fill-style, color)
```

8)		
9)		
10)	Write an expression to generate a `rhombus` that is a square!	

Triangle Contracts

Respond to the questions. Go to [code.pyret.org (CPO)] to test your code.

1) What kind of triangle does the `triangle` function produce? _____
There are lots of other kinds of triangles! And Pyret has lots of other functions that make triangles!

```
# triangle :: ( Number , String , String ) -> Image
              (  size   , fill-style , color )
# right-triangle :: ( Number , Number , String , String ) -> Image
                    (  base   , height , fill-style , color )
# isosceles-triangle :: ( Number , Number , String , String ) -> Image
                         (  leg    , angle  , fill-style , color )
```

2) Why do you think `triangle` only needs one number, while `right-triangle` and `isosceles-triangle` need two numbers?

_____

_____

3) Write `right-triangle` expressions for the images below using `100` as one argument for each.

_____

_____

4) Write `isosceles-triangle` expressions for the images below using `100` as one argument for each.

_____

_____

5) Write 2 expressions that would build **right-isosceles** triangles. Use `right-triangle` for one expression and `isosceles-triangle` for the other expression.

_____

_____

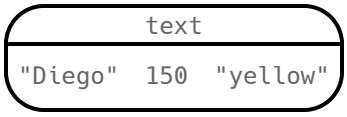
6) Which do you like better? Why? _____

Composing with Circles of Evaluation

Notice and Wonder

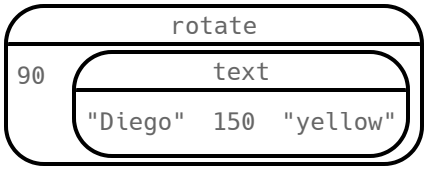
Suppose we want to see the `text` *"Diego" written vertically in yellow letters of size 150. Let's use Circles of Evaluation to look at the structure:*

We can start by generating the Diego image.



→

And then use the `rotate` function to rotate it 90 degrees.




```
text("Diego", 150, "yellow")
```

```
rotate(90, text("Diego", 150, "yellow"))
```

1) What do you Notice? _____

_____

2) What do you Wonder? _____

_____

Let's Rotate an Image of Your Name!

Suppose you wanted the computer to show your name in your favorite color and rotate it so that it's diagonal…

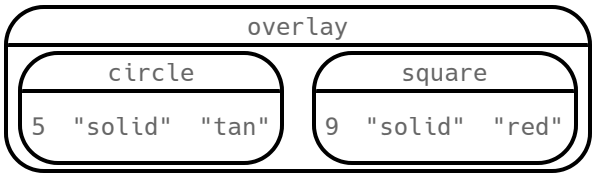
Write your name (any size), in your favorite color	**`rotate` the image so that it's diagonal**
3) Draw the circle of evaluation:	4) Draw the circle of evaluation:
5) Convert the Circle of Evaluation to code:	6) Convert the Circle of Evaluation to code:

Circle of Evaluation to Code (Scaffolded)

Complete the Code by Filling in the Blanks!

Finish the Code by filling in the blanks.

1)

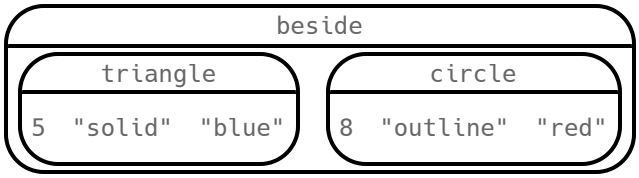



```
overlay(circle(____, "solid", _____________________), _____________________(9, _____________________, "red"))
```

Complete the Code by adding Parentheses

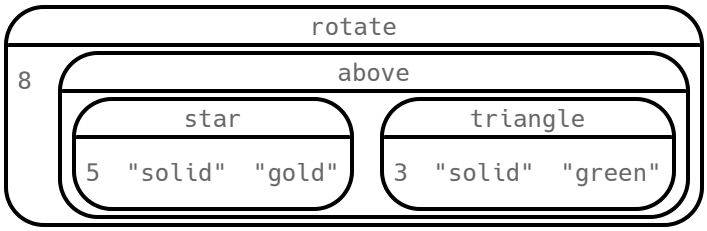
For each Circle of Evaluation, finish the Code by adding parentheses and commas.

2)



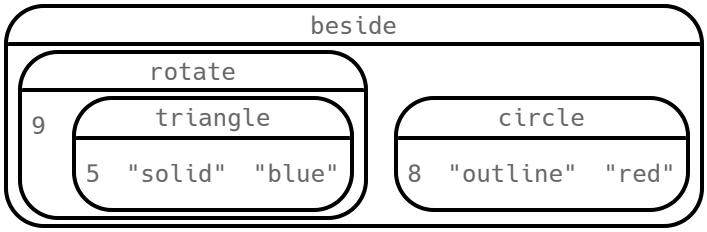

```
beside    triangle    5    "solid"    "blue"    circle    8    "outline" "red"
```

3)




```
rotate    8    above    star    5    "solid"    "gold"    triangle    3    "solid"    "green"
```

4)




```
beside    rotate    9    triangle    5    "solid"    "blue"    circle    8    "outline"    "red"
```

Frayer Model: Domain and Range

My Definition

Facts and Characteristics

Domain

Examples

Non-Examples

My Definition

Facts and Characteristics

Range

Examples

Non-Examples

(optional)

Frayer Model: Function and Variable

My Definition

Facts and Characteristics

Function

Examples

Non-Examples

My Definition

Facts and Characteristics

Variable

Examples

Non-Examples

(optional)

Radial Star

```
# radial-star :: ( Number , Number , Number , String , String ) -> Image
                   points   outer-radius  inner-radius  fill-style  color
```

Using the Contract above, match the images on the left to the expressions on the right. Test the code at code.pyret.org (CPO).

1

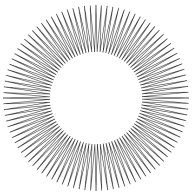
A

```
radial-star(5, 200, 50, "solid", "black")
```

2

B

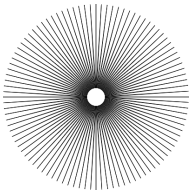
```
radial-star(7, 200, 100, "solid", "black")
```



3

C

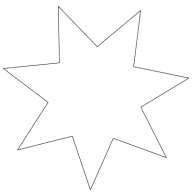
```
radial-star(7, 200, 100, "outline", "black")
```



4

D

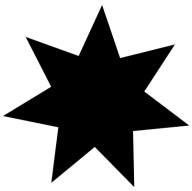
```
radial-star(10, 200, 150, "solid", "black")
```



5

E

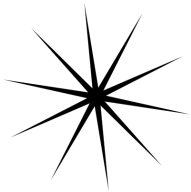
```
radial-star(10, 200, 20, "solid", "black")
```



6

F

```
radial-star(100, 200, 20, "outline", "black")
```



7

G

```
radial-star(100, 200, 100, "outline", "black")
```

(optional)

Triangle Contracts (SAS & ASA)

Type each expression (left) below into the code.pyret.org (CPO) and match it to the image it creates (right).

Expression			Image
`triangle-sas(120, 45, 70, "solid", "black")`	1	A	
`triangle-sas(120, 90, 70, "solid", "black")`	2	B	
`triangle-sas(120, 135, 70, "solid", "black")`	3	C	
`triangle-sas(70, 135, 120, "solid", "black")`	4	D	

Contracts

Think about how you would describe each `triangle-sas` argument to someone who'd never used the function before.

5) Annotate the Contract below using descriptive variable names.

```
triangle-sas :: ( ___Number___ , ___Number___ , ___Number___ , ___String___ , ___String___ ) -> Image
```

If you have a printed workbook, add examples of each of the triangle functions we've explored to your contracts pages.

★ If you have time, experiment with the `triangle-asa` function.

```
# triangle-asa :: ( ___Number___ , ___Number___ , ___Number___ , ___String___ , ___String___ ) -> Image
                    top-left-angle   left-side     bottom-angle    fill-style      color
```

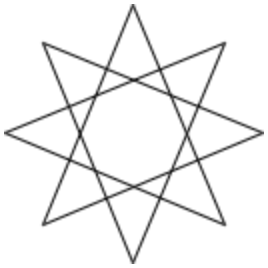
★ Why did these two functions need to take in one more Number than `right-triangle` did?

(optional)

Star Polygon

```
# star-polygon :: (  Number  ,  Number  ,  Number  ,  String  ,  String  ) -> Image
                   side-length  points-on-polygon  points-to-skip-for-star  fill-style  color
```

1. Using the Contract above, write expressions to create images like those pictured below.
2. Go to code.pyret.org (CPO) to test your code.
3. Then write expressions to generate two more star polygons of your choosing.
   Sketch them and record your working code.

1		
2		
3		
4		

(optional)

Function Composition — Green Star

1) Draw a Circle of Evaluation and write the Code for a **solid, green star, size 50**. *Then go to code.pyret.org (CPO) to test your code.*

Circle of Evaluation:

Code: _____

Using the star described above as the **original**, draw the Circles of Evaluation and write the Code for each exercise below. Test your code in the editor.

2) A solid, green star, that is triple the size of the original (using `scale`)	3) A solid, green star, that is half the size of the original (using `scale`)
4) A solid, green star of size 50 that has been rotated 45 degrees counter-clockwise	5) A solid, green star that is 3 times the size of the original **and** has been rotated 45 degrees

Function Composition — Your Name

You'll be investigating these functions with your partner:

```
# text :: String, Number, String -> Image
# flip-horizontal :: Image -> Image
# flip-vertical :: Image -> Image
# frame :: Image -> Image
# above :: Image, Image -> Image
# beside :: Image, Image -> Image
```

1) In the editor, write the code to make an image of your name in big letters in a color of your choosing using `text`. Then draw the Circle of Evaluation and write the Code that will create the image.

Circle of Evaluation for an "image of your name":

Code for an "image of your name": _____

Using the "image of your name" described above as the **original**, draw the Circles of Evaluation and write the Code for each exercise below. Test your ideas in the editor to make sure they work.

2) The framed "image of your name".	3) The "image of your name" flipped vertically.
4) The "image of your name" above a vertical reflection of the "image of your name"	5) The "image of your name" flipped horizontally beside "the image of your name".

Function Composition — scale-xy

You'll be investigating these two functions with your partner:

```
# scale-xy :: ( Number , Number , Image ) -> Image
              x-scale-factor  y-scale-factor  img-to-scale
```

```
# overlay :: ( Image , Image ) -> Image
               top     bottom
```

The Image:	Circle of Evaluation:	Code:
	rhombus 40 90 "solid" "purple"	`rhombus(40, 90, "solid", "purple")`

Starting with the image described above, write Circles of Evaluation and Code for each exercise below. *Be sure to test your code!*

1) A purple rhombus that is stretched 4 times as wide.

2) A purple rhombus that is stretched 4 times as tall

3) The tall rhombus from #1 overlayed on the wide rhombus (#2).

★ Overlay a red rhombus onto the last image you made in #3.

More than one way to Compose an Image!

What image will each of the four expressions below evaluate to?

If you're not sure, go to code.pyret.org (CPO) and type them into the Interactions Area and see if you can figure out how the code constructs its image.

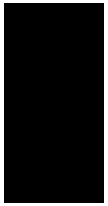
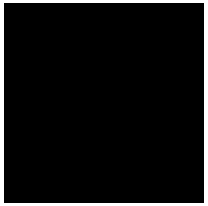
```
beside(rectangle(200, 100, "solid", "black"), square(100, "solid", "black"))

scale-xy(1, 2, square(100, "solid", "black"))

scale(2, rectangle(100, 100, "solid", "black"))

above(
    rectangle(100, 50, "solid", "black"),
    above(
        rectangle(200, 100,  "solid", "black"),
        rectangle(100, 50, "solid", "black")))
```

For each image below, identify 2 expressions that could be used to compose it. *The bank of expressions at the top of the page includes one possible option for each image.*

1		
2		
3		
★		

Function Cards

Print and cut these out, for use with the unplugged "function composition" activity.

```
# double :: Number -> Number
# consumes a number, and multiplies that number
by 2
```

```
# half :: Number -> Number
# consumes a number, and produces a number that
is half the input
```

```
# add5 :: Number -> Number
# consumes a number, adds five, and produces the
result
```

```
# sub10 :: Number -> Number
# consumes a number, subtracts ten, and produces
the result
```

```
# sqr :: Number -> Number
# consumes a number, squares it, and produces the
result
```

```
# neg :: Number -> Number
# consumes a number, multiplies it by -1, and
produces the result
```

```
# add1 :: Number -> Number
# consumes a number, adds one, and produces the
result
```

```
# f :: Number -> Number
# consumes a number, subtracts seven, and
produces the result
```

```
# g :: Number -> Number
# consumes a number, adds six, and produces the
result
```

```
# h :: Number -> Number
# consumes a number, subtracts one, and produces
the result
```

Defining Values in a Nutshell

In math, we use values, expressions and definitions.

- **Values** include things like: - 98.1 $^{2}/_{3}$ 42
- **Expressions** include things like: 1×3 $\sqrt{16}$ 5 - 2
  - These evaluate to results, and typing any of them in as code produces some answer.
- **Definitions** are different from values and expressions, because *they do not produce results*. Instead, they simply create names for values, so that those names can be re-used to make the Math simpler and more efficient.
  - Definitions always have both a name and an expression.
  - The name goes on the left and is defined by an equals sign to be the result of a value-producing expression on the right:
    $x = 4$
    $y = 9 + x$
  - The above examples tells us:
    "x is defined to be 4."
    "y is defined to be 13."
  - **Important: there is no "answer" to a definition**, and typing in a definition as code will produce no result.
  - Notice that *once a value has been defined, it can be used in subsequent definitions*. In the example above...
    The definition of `y` refers to `x`.
    The definition of `x`, on the other hand, *cannot* refer to `y`, because it comes before `y` is defined.

In Pyret, definitions are written the *exact same way*!

- Try typing these definitions into the Definitions Area on the left, clicking "Run", and then *using* them in the Interactions Area on the right.
  - `x = 4`
  - `y = 9 + x`

Just like in math, definitions in our programming language can only refer to previously-defined values.

- Here are a few more value definitions. Feel free to type them in, and make sure you understand them.
  - `x = 5 + 1`
  - `y = x * 7`
  - `food = "Pizza!"`
  - `dot = circle(y, "solid", "red")`

Defining Values - Explore

Open the Defining Values Starter File and click "Run".

1) What do you Notice?

2) What do you Wonder?

For each of the expressions listed below, write your *prediction* for what you expect Pyret to produce? Once you have completed your predictions, test them out one at a time in the Interactions Area.

	Prediction	Result		Prediction	Result
3) `x`			4) `x + 5`		
5) `y - 9`			6) `x * y`		
7) `z`			8) `t`		
9) `gold-star`			10) `my-name`		
11) `swamp`			12) `c`		

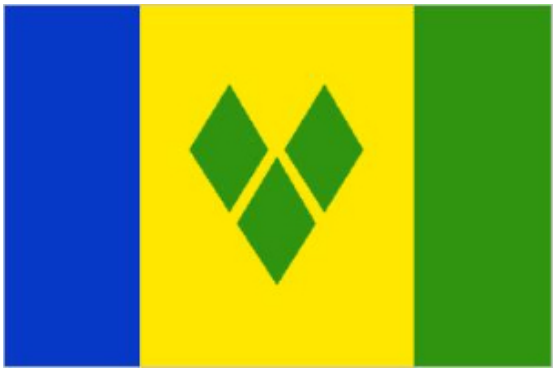
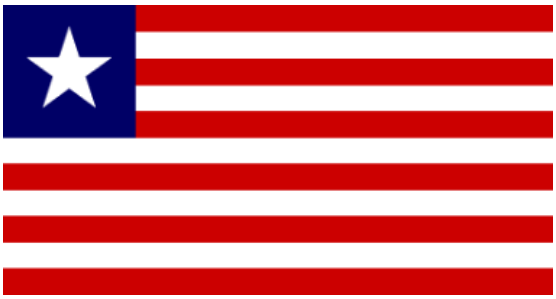
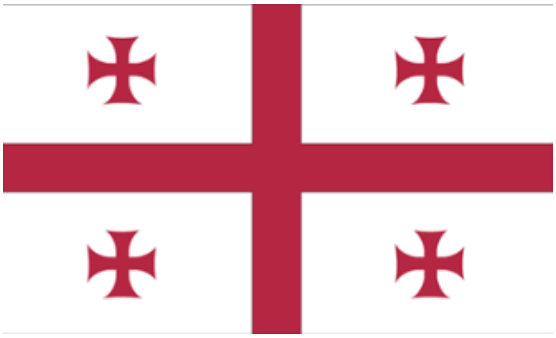
13) In the code, find the definitions of `exampleA`, `exampleB`, and `exampleC`. These all define the same shape, but their definitions are split across several lines. Suppose you *had* to split your code across multiple lines like this. Which one of these is the easiest to read, and why?

14) Define at least 2 more variables in the Definitions Area, click "Run" and test them out. Once you know they're working, record the code you used below.

15) What have you learned about defining values?

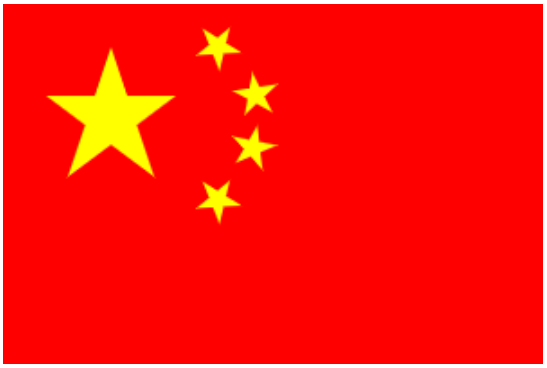
Which Value(s) Would it Make Sense to Define?

For each of the images below, identify which element(s) you would want to define before writing code to compose the image.
Hint: what gets repeated?

Philippines	St. Vincent & the Grenadines
	
1) _____	2) _____
Liberia	**Republic of Georgia**
	
3) _____	4) _____
Quebec	**South Korea**
	
5) _____	6) _____

Chinese Flag

The image value on the left called `china` is defined by the code on the right.



```
china =
  translate(
    rotate(40,star(15,"solid","yellow")),
    120, 175,
    translate(
      rotate(80,star(15,"solid","yellow")),
      140, 150,
      translate(
        rotate(60,star(15,"solid","yellow")),
        140, 120,
        translate(
          rotate(40,star(15,"solid","yellow")),
          120, 90,

translate(scale(3,star(15,"solid","yellow")),
            60, 140,
            rectangle(300, 200, "solid", "red"))))))
```

1) What image do you see repeated in the flag?

_____

2) **Highlight or underline** every place in the code ➡️ that you see the repeated expression for that image.

3) Write the code to **define a value** for the repeated expression.

_____

4) Open the Flag of China Starter File, **save a copy** and click "Run". **Simplify the code**, replacing the repeated expressions with the value you defined. Do you still get the same image when you click "Run"? If not, check your work.

5) Change the color of all the stars to black, then change their size to 20. Would this have been easier with the original code? Why or why not?

_____

6) Here is the same code shown above, but all crammed into one line.

```
china = translate(rotate(40, star(15, "solid", "yellow")), 120, 175, translate(rotate(80, star(15, "solid", "yellow")), 140, 150, translate(rotate(60, star(15, "solid", "yellow")), 140, 120, translate(rotate(40, star(15, "solid", "yellow")), 120, 90, translate(scale(3, star(15, "solid", "yellow")), 60, 140, rectangle(300, 200, "solid", "red"))))))
```

Is it easier or harder to read, when everything is all on one line? _____

7) Professional programmers *indent* their code, by breaking long lines into shorter, more readable lines of code. In the indented code at the top of the page, notice that each `translate` is followed by several lines of code that all line up with each other, and that the lines under the *next* `translate` are shifted farther and farther to the right. What do you think is going on?

_____

_____

_____

★ This file uses a function we haven't seen before! *Hint: Focus on the last instance of the function.* What is its name? _____.

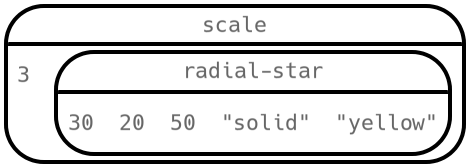
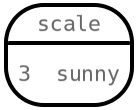
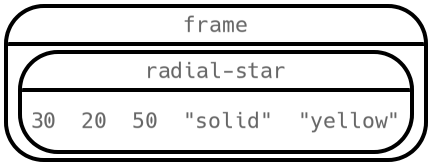
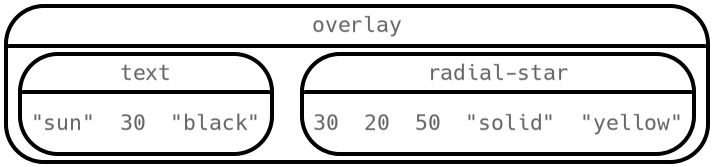
How many inputs are in its domain? _____. What are the types of those inputs? _____

Why Define Values?

Take a close look at the Original Circle of Evaluation & Code and how it got simplified.

1) Write the code that must have been used to define the value of `sunny`. _____

2) Complete the table using the first row as an example.

Original Circle of Evaluation & Code		**Use the *defined value* `sunny` to simplify!**
	→	
`scale(3, radial-star(30, 20, 50, "solid", "yellow"))`	→	Code: `scale(3, sunny)`
Second Circle of Evaluation & Code		**Use the *defined value* `sunny` to simplify!**
	→	
`frame(radial-star(30, 20, 50, "solid", "yellow"))`	→	Code:
Third Circle of Evaluation & Code		**Use the *defined value* `sunny` to simplify!**
	→	
`overlay(text("sun", 30, "black"), radial-star(30, 20, 50, "solid", "yellow"))`	→	Code:

3) Define `sunny` in the Definitions Area using the code you recorded at the top of the page.

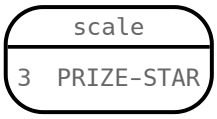
4) Test your code in the editor and make sure it produces what you would expect it to.

Writing Code using Defined Values

1) On the line below, **write the Code** to define `PRIZE-STAR` as the pink outline of a size 65 star.

Using the `PRIZE-STAR` definition from above, draw the Circle of Evaluation and write the Code for each of the exercises.
Be sure to test out your code in code.pyret.org (CPO) before moving onto the next item. One Circle of Evaluation has been done for you.

2) The outline of a pink star that is three times the size of the original (using `scale`)
Circle of Evaluation:



Code:

3) The outline of a pink star that is half the size of the original (using `scale`)
Circle of Evaluation:

Code:

4) The outline of a pink star that is rotated 45 degrees
(It should be the same size as the original.)
Circle of Evaluation:

Code:

5) The outline of a pink star that is three times as big as the original **and** has been rotated 45 degrees
Circle of Evaluation:

Code:

6) How does defining values help you as a programmer?

Making Sense of Coordinates

```
dot = circle(50, "solid", "red")
background = rectangle(300, 200, "outline", "black")
```

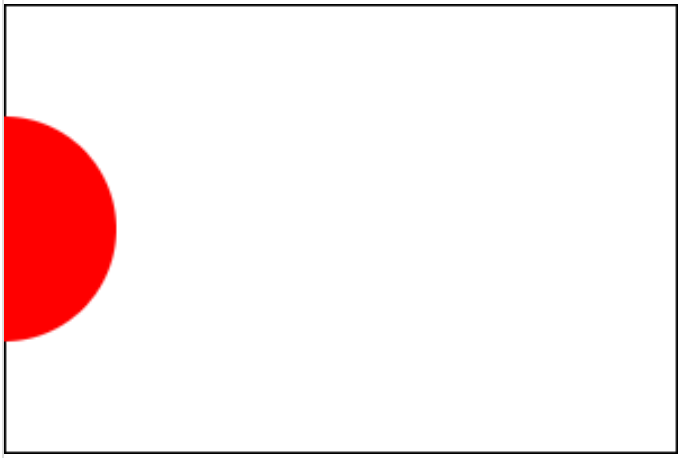
Think of the background image as a sheet of graph paper with the origin (0,0) in the bottom left corner. The width of the rectangle is 300 and the height is 200. The numbers in `translate` specify a point on that graph paper, where the center of the top image (in this case `dot`) should be placed.

What coordinates would you expect were used to place the `dot` for each of the following images?

1)

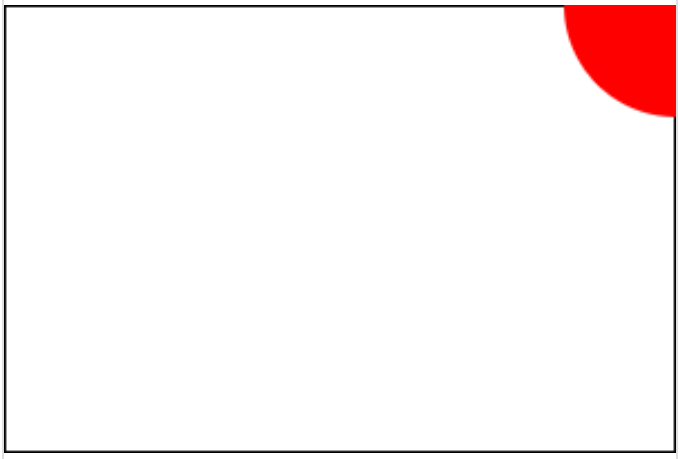
translate(dot, _____, _____background)

2)



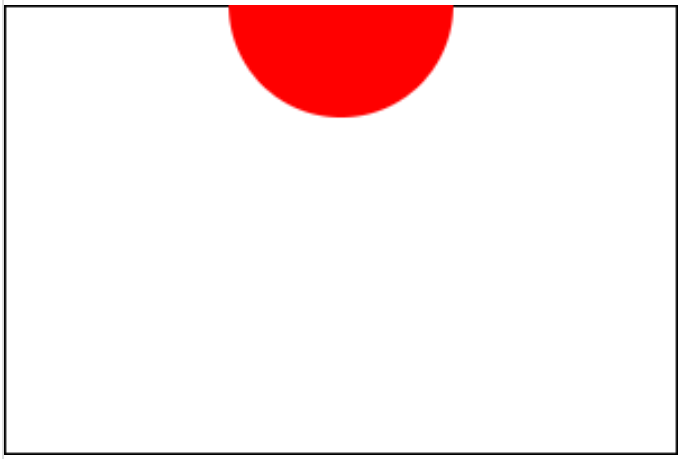
translate(dot, _____, _____background)

3)



translate(dot, _____, _____background)

4)



translate(dot, _____, _____background)

Investigating translate

Japan

For this section of the page, you will refer to the Flags Starter File.

1) Each language has its own symbol for commenting code so that programmers can leave notes that won't be read by the computer. In Pyret, we use the hash mark (`#`). What color are comments in Pyret? _____

2) Type `japan-flag` into the Interactions Area. What do you get back? _____

_____

3) Type `japan` into the Interactions Area and compare the image to `japan-flag`.

- How are they alike? _____
- How are they different? _____

4) `japan` is composed using `dot` and `background`. Type each of those variables into the Interactions Area. What do you get back?

- `dot`: _____
- `background`: _____

5) These images are combined using the `translate` function. What is its contract? _____

6) Fix the `japan` code so that it matches the `japan-flag` image. What did you need to change? _____

_____

7) How can you prove that you have placed the `dot` in exactly the right location? _____

_____

The Netherlands

For this section of the page, you will refer to the Flags of Netherlands, France & Mauritius Starter File.

8) What was the programmer thinking when she coded the height of the red stripe as `200 / 3`? _____

_____

9) The center of the blue stripe is placed at (`150`, `200 / 6`). How did the programmer know to use 150 as the x-coordinate? _____

_____

10) What was the programmer thinking when she coded the y-coordinate as `200 / 6`? _____

_____

11) Explain the thinking behind coding the red stripe's y-coordinate as `5 * (200 / 6)`. _____

_____

12) What advantages are there to representing height / length / width as fractions (as we see in this code) rather than using a computed value?

_____

_____

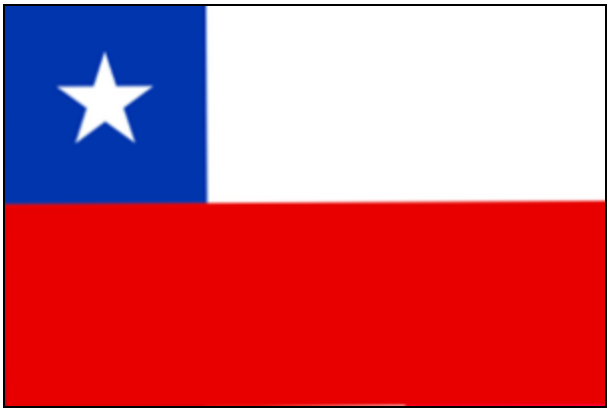
Decomposing Flags

Each of the flags below is shown with their width and height. Identify the shapes that make up each flag. Use the flag's dimensions to estimate the dimensions of the different shapes. Then estimate the x and y coordinates for the point at which the center of each shape should be located on the flag. *Hint: The bottom left corner of each flag is at (0,0) and the top right corner is given by the flags dimensions.*

1) **Cameroon** (450 x 300)

shape:	color:	width:	height:	x	y

2) **Chile** (420 x 280)

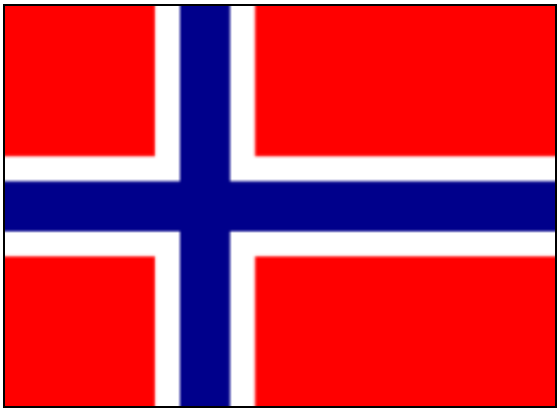


shape:	color:	width:	height:	x	y

3) **Panama** (300 x 200)

shape:	color:	width:	height:	x	y

4) **Norway** (330 x 240)



shape:	color:	width:	height:	x	y

(optional)

Coding and Designing the Alaskan Flag

Open the Flag of Alaska Starter File. Click run and type `alaska` *to see an image of the flag of Alaska.*

Exploring the Code

1) How many images are defined in the code? _____

2) How many images are placed using `translate` in order to generate the flag? _____

3) Why do your answers to these questions differ? _____

4) The code for the flag could have been written without defining any images. What are some reasons why defining images makes the code easier to work with?

The Story of the Flag of Alaska

Benny Benson holding the flag of Alaska that he designed

The Alaska state flag is based on a design created in 1926 for a Territory-wide contest for schoolchildren. The thirteen-year-old seventh-grade designer was Benny Benson from the Aleutian Islands. *(At the time, Alaska was not yet a state; it had been a US Territory since the land was purchased from Russia in 1867.)*

On the design submission, Benny had written the following explanation:

"The blue field is for the Alaska sky and the forget-me-not, an Alaska flower.
The North Star is for the future of the state of Alaska, the most northerly in the Union.
The dipper is for the Great Bear — symbolizing strength."

Benny's flag was officially adopted by the legislature in 1927.

Alaska was officially recognized as a state on January 3, 1959.

5) How old was Benny when Alaska achieved statehood?

6) Think of someone you know who is old enough to remember 1959. (Your teacher is not old enough!). Find a time this week to visit or call and ask them if they remember anything about when Alaska became a state! Record what you learn below.

(optional)

Defining Functions in a Nutshell

Functions can be viewed in *multiple representations*.

Contract and Purpose

You already know one of them: ***Contracts***, which specify the Name, Domain, and Range of a function. Contracts are a way of thinking of functions as a *mapping* between one set of data and another. For example, a mapping from Numbers to Strings:

```
# f :: Number -> String
```

Examples

The goal of the ***Examples*** step is to *find the pattern* that represents what the function does.

Examples are essentially input-output tables, showing what the functions does with a list of specific inputs. *In our programming language, we write the table columns as code.*

How f is used	What f does
f(1)	$1 + 2$
f(2)	$2 + 2$
f(3)	$3 + 2$
f(4)	$4 + 2$

```
examples:
  f(1) is 1 + 2
  f(2) is 2 + 2
  f(3) is 3 + 2
  f(4) is 4 + 2
end
```

Definition

The final step in the Design Recipe is to *generalize the pattern* we see in our examples by writing a formal **function definition**. To do this we replace the inputs with ***variables*** that can work with any input.
In the example below, the definition for the examples above is written in both math and code:

$$f(x) = x + 2$$

```
fun f(x): x + 2 end
```

Look for connections between these three representations!

- The function name is always the same, whether looking at the Contract, Examples, or Definition.
- The number of inputs in the Examples is always the same as the number of types in the Domain, which is always the same as the number of variables in the Definition.
- The "what the function does" pattern in the Examples is almost the same in the Definition, but with specific inputs replaced by variables.

The Great gt domain debate!

Kermit: The domain of `gt` is `Number, String, String`.
Oscar: The domain of `gt` is `Number`.
Ernie: I'm not sure who's right!

In order to make a triangle, we need a size, a color and a fill style…
but all we had to tell our actor was `gt(20)`…and they returned `triangle(20, "solid", "green")`.

Please help us!

1) What is the correct domain for gt?

2) What could you tell Ernie to help him understand how you know?

Let's Define Some New Functions!

1) **Let's define a function `rs` to generate solid red squares of whatever size we give them!**
If I say `rs(5)`, what would our actor need to say?

Let's write a few more examples:

`rs(_____)` → _____

`rs(_____)` → _____

`rs(_____)` → _____

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable:

`fun rs(_____):` _____ `end`

2) **Let's define a function `bigc` to generate big solid circles of size 100 in whatever color we give them!**
If I say `bigc("orange")`, what would our actor need to say?

Let's write a few more examples:

`bigc(_____)` → _____

`bigc(_____)` → _____

`bigc(_____)` → _____

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable:

`fun bigc(_____):` _____ `end`

3) **Let's define a function `ps` to build a pink star of size 50, with the input determining whether it's solid or outline!**
If I say `ps("outline")`, what would our actor need to say?

Write examples for all other possible inputs:

`ps(_____)` → _____

`ps(_____)` → _____

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable:

`fun ps(_____):` _____ `end`

4) Add these new function definitions to your gt Starter File and test them out!

Let's Define Some More New Functions!

1) **Let's define a function `sun` to write SUNSHINE in whatever color and size we give it!**
If I say `sun(5, "blue")`, what would our actor need to say?

_____

Let's write a few more examples:

`sun(_____, _____)` → _____

`sun(_____, _____)` → _____

`sun(_____, _____)` → _____

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable(s):

```
fun sun(_____, _____): _____ end
```

2) **Let's define a function `me` to generate your name in whatever size and color we give it!**
If I say `me(18, "gold")`, what would our actor need to say?

_____

Let's write a few more examples:

`me(_____, _____)` → _____

`me(_____, _____)` → _____

`me(_____, _____)` → _____

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable(s):

```
fun me(_____, _____): _____ end
```

3) **Let's define a function `gr` to build a solid, green rectangle of whatever height and width we give it!**
If I say `gr(10, 80)`, what would our actor need to say?

_____

Let's write a few more examples:

`gr(_____, _____)` → `rectangle(_____, _____, "solid", "green")`

`gr(_____, _____)` → `rectangle(_____, _____, "solid", "green")`

`gr(_____, _____)` → `rectangle(_____, _____, "solid", "green")`

What changes in these examples? Name your variable(s): _____
Let's define our function using the variable(s):

```
fun gr(_____, _____): _____ end
```

4) Add these new function definitions to your gt Starter File and test them out!

Describe and Define Your Own Functions!

1) **Let's define a function** _____ **to generate...**

_____

If I say _____, what would our actor need to say? _____

Let's write a few more examples:

```
_____(_____) → _____(_____)
_____(_____) → _____(_____)
_____(_____) → _____(_____)
```

What changes in these examples? Name your variable(s): _____

Let's define our function using the variable.

```
fun _____(_____): _____ end
```

2) **Let's define a function** _____ **to generate...**

_____

If I say _____, what would our actor need to say? _____

Let's write a few more examples:

```
_____(_____) → _____(_____)
_____(_____) → _____(_____)
_____(_____) → _____(_____)
```

What changes in these examples? Name your variable(s): _____

Let's define our function using the variable.

```
fun _____(_____): _____ end
```

3) **Let's define a function** _____ **to generate...**

_____

If I say _____, what would our actor need to say? _____

Let's write a few more examples:

```
_____(_____) → _____(_____)
_____(_____) → _____(_____)
_____(_____) → _____(_____)
```

What changes in these examples? Name your variable(s): _____

Let's define our function using the variable.

```
fun _____(_____): _____ end
```

4) Add your new function definitions to your gt Starter File and test them out!

Matching Examples and Contracts

Match each set of examples (left) with the Contract that best describes it (right).

Examples			Contract

`examples:`
`  f(5) is 5 / 2`
`  f(9) is 9 / 2`
`  f(24) is 24 / 2`
`end`
	1	A	`# f :: Number -> Number`

`examples:`
`  f(1) is rectangle(1, 1, "outline", "red")`
`  f(6) is rectangle(6, 6, "outline", "red")`
`end`
	2	B	`# f :: String -> Image`

`examples:`
`  f("pink", 5) is star(5, "solid", "pink")`
`  f("blue", 8) is star(8, "solid", "blue")`
`end`
	3	C	`# f :: Number -> Image`

`examples:`
`  f("Hi!") is text("Hi!", 50, "red")`
`  f("Ciao!") is text("Ciao!", 50, "red")`
`end`
	4	D	`# f :: Number, String -> Image`

`examples:`
`  f(5, "outline") is star(5, "outline", "yellow")`
`  f(5, "solid") is star(5, "solid", "yellow")`
`end`
	5	E	`# f :: String, Number -> Image`

Matching Examples and Function Definitions

(1) Find the variables in `gt` and label them with the word "size".

```
examples:
    gt(20) is triangle(20, "solid", "green")
    gt(50) is triangle(50, "solid", "green")
end
fun gt(size): triangle(size, "solid", "green") end
```

(2) Highlight and label the variables in the example lists below.

(3) Then, using `gt` as a model, match the examples to their corresponding function definitions.

Examples		Definition

`examples:`
`  f("solid") is circle(8, "solid", "red")`
`  f("outline") is circle(8, "outline", "red")`
`end`
	1	A `fun f(s): star(s, "outline", "red") end`

`examples:`
`  f(2) is 2 + 2`
`  f(4) is 4 + 4`
`  f(5) is 5 + 5`
`end`
	2	B `fun f(num): num + num end`

`examples:`
`  f("red") is circle(7, "solid", "red")`
`  f("teal") is circle(7, "solid", "teal")`
`end`
	3	C `fun f(c): star(9, "solid", c) end`

`examples:`
`  f("red") is star(9, "solid", "red")`
`  f("grey") is star(9, "solid", "grey")`
`  f("pink") is star(9, "solid", "pink")`
`end`
	4	D `fun f(s): circle(8, s, "red") end`

`examples:`
`  f(3) is star(3, "outline", "red")`
`  f(8) is star(8, "outline", "red")`
`end`
	5	E `fun f(c): circle(7, "solid", c) end`

Creating Contracts From Examples

Write the contracts used to create each of the following collections of examples. The first one has been done for you.

1)
```
# big-triangle :: Number, String -> Image
examples:
  big-triangle(100, "red") is triangle(100, "solid", "red")
  big-triangle(200, "orange") is triangle(200, "solid", "orange")
end
```

2)
```
examples:
  purple-square(15) is rectangle(15, 15, "outline", "purple")
  purple-square(6) is rectangle(6, 6, "outline", "purple")
end
```

3)
```
examples:
  sum(5, 8) is 5 + 8
  sum(9, 6) is 9 + 6
  sum(120, 11) is 120 + 11
end
```

4)
```
examples:
  banner("Game Today!") is text("Game Today!", 50, "red")
  banner("Go Team!") is text("Go Team!", 50, "red")
  banner("Exit") is text("Exit", 50, "red")
end
```

5)
```
examples:
  twinkle("outline", "red") is star(5, "outline", "red")
  twinkle("solid", "pink") is star(5, "solid", "pink")
  twinkle("outline", "grey") is star(5, "outline", "grey")
end
```

6)
```
examples:
  half(5) is 5 / 2
  half(8) is 8 / 2
  half(900) is 900 / 2
end
```

7)
```
examples:
  Spanish(5) is "cinco"
  Spanish(30) is "treinta"
  Spanish(12) is "doce"
end
```

Contracts, Examples & Definitions - bc

We've already found the Contract for gt, made Examples, and described the pattern with a Definition. Let's review the process.
Directions: Define a function called `gt`, which makes solid green triangles of whatever size we want.

Contract and Purpose Statement

Every contract has three parts…

`# gt :: Number -> Image`
function name | Domain | Range

Examples

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

`gt(10)` **is** `triangle(10, "solid", "green")`
function name | input(s) | what the function produces

`gt(20)` **is** `triangle(20, "solid", "green")`
function name | input(s) | what the function produces

end

Definition

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

fun `gt(size)`:
function name | variable(s)

`triangle(size, "solid", "green")`
what the function does with those variable(s)

end

Now, let's apply the same steps to think through a new problem!
Directions: Define a function called `bc`, which makes solid blue circles of whatever radius we want.

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

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, Examples & Definitions - Stars

Directions: Define a function called `sticker`, which consumes a color and draws a solid 50px star of the given color.

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

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: Define a function called `gold-star`, which takes in a radius and draws a solid gold star of that given size.

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

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, Examples & Definitions - Name

Directions: Define a function called `name-color`, which makes an image of your name at size 50 in whatever color is given.

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

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: Define a function called `name-size`, which makes an image of your name in your favorite color (be sure to specify your name and favorite color!) in whatever size is given.

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

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

Do the Examples Have the Same Contracts?

For each pair of Examples below, decide whether the two examples have the same Contract. If they do, fill in the Contract in the space provided. If not, write a few words explaining how you know their contracts aren't the same.

1) _____

```
examples:
  mystery(30) is 30 * 50
  mystery(10) is text("Welcome!", 10, "darkgreen")
end
```

2) _____

```
examples:
  mystery(30, 40) is 40 - (2 * 30)
  mystery(10, 15) is 15 - (2 * 10)
end
```

3) _____

```
examples:
  mystery("New York") is text("New York", 20, "red")
  mystery(20) is text("New York", 20, "red")
end
```

4) _____

```
examples:
  mystery("green", 32) is circle(32, "outline", "green")
  mystery(18, "green") is circle(18, "outline", "green")
end
```

5) _____

```
examples:
  mystery(6, 9, 10) is 6 / (9 + 10)
  mystery(3, 7) is 3 / (7 + 10)
end
```

6) _____

```
examples:
  mystery("red", "blue") is text("blue", 25, "red")
  mystery("purple", "Go Team!") is text("Go Team!", 25, "purple")
end
```

(optional)

Do the Examples Have the Same Contracts? (2)

For each pair of Examples below, decide whether the two examples have the same Contract. If they do, fill in the Contract in the space provided. If not, write a few words explaining how you know their contracts aren't the same.

1) _____

```
examples:
  mystery(triangle(70, "solid", "green")) is triangle(140, "solid", "green")
  mystery(circle(100, "solid", "blue")) is circle(200, "solid", "blue")
end
```

2) _____

```
examples:
  mystery("red") is triangle(140, "solid", "red")
  mystery("blue", "circle") is circle(140, "solid", "blue")
end
```

3) _____

```
examples:
  mystery("+", 4, 5) is 4 + 5
  mystery("sqrt", 25) is sqrt(25)
end
```

4) _____

```
examples:
  mystery("circle", 4) is PI * sqr(4)
  mystery("square", 5) is sqr(5)
end
```

5) _____

```
examples:
  mystery("dog") is 3
  mystery("cat") is "kitten"
end
```

6) _____

```
examples:
  mystery("dog") is 3
  mystery("kitten") is 6
end
```

(optional)

Matching Examples and Contracts (2)

Match each Example on the left with its Contract on the right. NOTE: Multiple examples may match to the same Contract!

Contract		Examples	

`examples:`
`  match(circle(10, "solid", "green")) is rotate(37, circle(10, "solid", "green"))`
`end`
	1	A	`# match :: Number, Image -> Image`

`examples:`
`  match(triangle(20, "solid", "blue"), 3) is scale(3, triangle(20, "solid", "blue"))`
`end`
	2		

`examples:`
`  match(circle(20, "outline", "gold")) is rotate(37, circle(20, "outline", "gold"))`
`end`
	3	B	`# match :: Image, Number -> Image`

`examples:`
`  match(30, "red") is 30 + string-length("red")`
`end`
	4		

`examples:`
`  match(circle(10, "solid", "orange"), 22) is scale(22, circle(10, "solid", "orange"))`
`end`
	5		

`examples:`
`  match(10, "blue") is 10 + string-length("blue")`
`end`
	6	C	`# match :: Image -> Image`

`examples:`
`  match(5, star(20, "solid", "red")) is rotate(90 - 5, star(20, "solid", "red"))`
`end`
	7		

`examples:`
`  match(abs(-4), "45") is 4`
`end`
	8	D	`# match :: Number, String -> Number`

(optional)

Matching Examples and Contracts (3)

Match each Example on the left with its Contract on the right. NOTE: Multiple examples may match to the same Contract!

Contract		Examples

`examples:`
`  match(1.5) is "greater than 1"`
`end`
	1	

`examples:`
`  match(24) is star(24 * 2, "outline", "purple"`
`)`
`end`
	2	

`examples:`
`  match(string-length("tabletop")) is "8"`
`end`
	3	A `# match :: Number -> String`

`examples:`
`  match(star(20, "outline", "red"), 3) is 3 *`
`image-height(star(20, "outline", "red"))`
`end`
	4	B `# match :: Number -> Image`

`examples:`
`  match(circle(10, "solid", "silver"), 16) is`
`16 * image-height(circle(10, "solid", "silver")`
`)`
`end`
	5	C `# match :: Number, Number ->`
		`Number`

`examples:`
`  match("triangle", "blue") is triangle(40,`
`"outline", "blue")`
`end`
	6	D `# match :: String, String ->`
		`Image`

`examples:`
`  match(30) is star(30 * 2, "outline", "purple"`
`)`
`end`
	7	E `# match :: Image, Number ->`
		`Number`

`examples:`
`  match(string-length("coffee"), string-length(`
`"tea")) is 6 + 3`
`end`
	8	

(optional)

Solving Word Problems in a Nutshell

Being able to see functions as Contracts, Examples or Definitions is like having three powerful tools. These representations can be used together to solve word problems! We call this **The Design Recipe**.

1) When reading a word problem, the first step is to figure out the **Contract** for the function you want to build. Remember, a Contract must include the Name, Domain and Range for the function!

2) Then we write a **Purpose Statement**, which is a short note that tells us what the function *should do*. Professional programmers work hard to write good purpose statements, so that other people can understand the code they wrote! Programmers work on teams; the programs they write must outlast the moment that they are written.

3) Next, we write at least two **Examples**. These are lines of code that show what the function should do for a *specific* input. Once we see examples of at least two inputs, we can *find a pattern* and see which parts are changing and which parts aren't.

4) To finish the Examples, we circle the parts that are changing, and label them with a short **variable name** that explains what they do.

5) Finally, we **define the function** itself! This is pretty easy after you have some examples to work from: we copy everything that didn't change, and replace the changeable stuff with the variable name!

Matching Word Problems and Purpose Statements

Match each word problem below to its corresponding purpose statement.

Annie got a new dog, Xavier, that eats about 5 times as much as her little dog, Rex, who is 10 years old. She hasn't gotten used to buying enough dogfood for the household yet. Write a function that generates an estimate for how many pounds of food Xavier will eat, given the amount of food that Rex usually consumes in the same amount of time.	**1**	**A**	Consume the pounds of food Rex eats and add 5.
Adrienne's raccoon, Rex, eats 5 more pounds of food each week than her pet squirrel, Lili, who is 7 years older. Write a function to determine how much Lili eats in a week, given how much Rex eats.	**2**	**B**	Consume the pounds of food Rex eats and subtract 5.
Alejandro's rabbit, Rex, poops about 1/5 of what it eats. His rabbit hutch is 10 cubic feet. Write a function to figure out how much rabbit poop Alejandro will have to clean up depending on how much Rex has eaten.	**3**	**C**	Consume the pounds of food Rex eats and multiply by 5.
Max's turtle, Rex, eats 5 pounds less per week than his turtle, Harry, who is 2 inches taller. Write a function to calculate how much food Harry eats, given the weight of Rex's food.	**4**	**D**	Consume the pounds of food Rex eats and divide by 5.

Writing Examples from Purpose Statements

We've provided contracts and purpose statements to describe two different functions. Write examples for each of those functions.

Contract and Purpose Statement

Every contract has three parts…

```
# triple:: Number -> Number
```
function name | Domain | Range

```
# Consumes a Number and triples it.
```
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

Contract and Purpose Statement

Every contract has three parts…

```
# upside-down:: Image -> Image
```
function name | Domain | Range

```
# Consumes an image, and turns it upside down by rotating it 180 degrees.
```
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

Fixing Purpose Statements

Beneath each of the word problems below is a purpose statement (generated by ChatGPT!) that is either missing information or includes unnecessary information.

- Write an improved version of each purpose statement beneath the original.
- Then, explain what was wrong with the ChatGPT-generated Purpose Statement.

1) **Word Problem:** *The New York City ferry costs $2.75 per ride. The Earth School requires two chaperones for any field trip. Write a function* `fare` *that takes in the number of students in the class and returns the total fare for the students and chaperones.*

ChatGPT's Purpose Statement: Take in the number of students and add 2.

Improved Purpose Statement: _____

_____

Problem with ChatGPT's Purpose Statement: _____

_____

2) **Word Problem:** *It is tradition for the Green Machines to go to Humpy Dumpty's for ice cream with their families after their soccer games. Write a function* `cones` *to take in the number of kids and calculate the total bill for the team, assuming that each kid brings two family members and cones cost $1.25.*

ChatGPT's Purpose Statement: Take in the number of kids on the team and multiply it by 1.25.

Improved Purpose Statement: _____

_____

Problem with ChatGPT's Purpose Statement: _____

_____

3) **Word Problem:** *The cost of renting an ebike is $3 plus an additional $0.12 per minute. Write a function* `ebike` *that will calculate the cost of a ride, given the number of minutes ridden.*

ChatGPT's Purpose Statement: Take in the number of minutes and multiply it by 3.12.

Improved Purpose Statement: _____

_____

Problem with ChatGPT's Purpose Statement: _____

_____

4) **Word Problem:** *Suleika is a skilled house painter at only age 21. She has painted hundreds of rooms and can paint about 175 square feet an hour. Write a function* `paint` *that takes in the number of square feet of the job and calculates how many hours it will take her.*

ChatGPT's Purpose Statement: Take in the number of square feet of walls in a house and divide them by 175 then add 21 years.

Improved Purpose Statement: _____

_____

Problem with ChatGPT's Purpose Statement: _____

_____

Word Problem: rocket-height

Directions: A rocket blasts off, and is now traveling at a constant velocity of 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…

_____ :: _____ -> _____
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

Rubric: Design Recipe

This rubric can be used for teachers to score students' Design Recipes or for peer review. If using this rubric for peer review, trade your Design Recipe with another student. Place this rubric and their Design Recipe side-by-side in front of you.

1) Go through the checklist in the left-hand column to assess their Contract. Check boxes or leave them blank depending on what you observe.

2) Once you have examined and analyzed the Contract, read the descriptive text (either "Wow!" or "Getting there") and check whichever one more accurately describes the work in front of you.

3) If the Design Recipe you're reviewing is "getting there," provide some descriptive feedback to help the student fix their work.

4) Repeat the process for the remaining sections of the Design Recipe.

The **CONTRACT**:	❏ Wow!	❏ Getting There…
❏ has correct function name ❏ has correct amount of Domain data types ❏ has correct data type(s) listed in the Domain ❏ has correct data type listed for the Range	The Contract you've written tells us a lot about how to use the function. In fact, we can figure out how to use your function just by looking at the Contract. You've included all essential information.	Something is missing from your Contract. It doesn't provide everything needed to understand the function. ***Here's what you need to do:***
The **PURPOSE STATEMENT**:	❏ Wow!	❏ Getting There…
❏ describes what the function consumes and produces ❏ describes how the result is computed, so that it can be combined to with the Contract to explain the Examples	The Purpose Statement is a concise and detailed restatement of the problem in your own words. It's a helpful explanation of what's happening in the problem.	Programmers and Mathematicians alike find it helpful to restate a problem in their own words. ***Your restatement is missing the following:***
The **EXAMPLES**:	❏ Wow!	❏ Getting There…
❏ have the correct function name ❏ have inputs that differ across Examples ❏ have the correct amount of Domain inputs ❏ have the correct expressions for what the function produces, using the given inputs ❏ have changeable parts circled and labeled	Your Examples not only help us to identify the pattern to define a function, they also let us double check that the functions we define do what we intend for them to do.	Your Examples do not help us to identify a pattern, or they don't allow us to double check our functions. ***Here's how you can improve that:***
The **DEFINITION**:	❏ Wow!	❏ Getting There…
❏ has the correct function name ❏ has the correct number, name, and order of variables (taken from the labels in the Examples section)	Your code correctly names the function, lists its variables, and states the expression to compute when the function is used!	Your Definition is missing something. ***Here's how to fix it:***

Writing Examples from Purpose Statements (2)

We've provided contracts and purpose statements to describe two different functions. Write examples for each of those functions.

Contract and Purpose Statement

Every contract has three parts…

`# half-image`:: `Image` -> `Image`
function name | Domain | Range

`# Consumes an image, and produces that image scaled to half its size.`
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

Contract and Purpose Statement

Every contract has three parts…

`# product-squared`:: `Number, Number` -> `Number`
function name | Domain | Range

`# Consumes two numbers and squares their product`
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

(optional)

Rocket Height Challenges

This page is designed to accompany work in the Rocket Height Starter File.

1) Can you make the rocket fly faster?

2) Can you make the rocket fly slower?

3) Can you make the rocket sink down instead of fly up?

4) Can you make the rocket accelerate over time, so that it moves faster the longer it flies?

5) Can you make the rocket blast off and then land again?

6) Can you make the rocket blast off, reach a maximum height of exactly 1000 meters, and then land?

7) Can you make the rocket blast off, reach a maximum height of exactly 1000 meters, and then land after exactly 100 seconds?

8) Can you make the rocket fly to the edge of the the universe?

(optional)

Design Recipe Telephone

Most computer programs are written by huge teams! It is critical that each team member records their thinking with enough detail for other team members to be able to pick up where they left off. We're going to practice collaborative programming through an activity called Design Recipe Telephone.

1. Prepare the class and the materials

Choose which set of word problems you are going to start with and print enough copies so that each student will get one word problem.
Divide the class into groups of three.
Give each student within each group a different word problem from the set.

Word Problem Set 1:	Word Problem Set 2:	Option 3:
Design Recipe Telephone Set 1: g Design Recipe Telephone Set 1: h Design Recipe Telephone Set 1: r ★*Once completed, the set of functions generated from these word problems can be used to fix the code in this Collaboration Starter File - For use with Design Recipe Telephone Set 1. If all the functions are defined correctly, the starter file will then generate a cool image!*	Design Recipe Telephone Set 2: symmetry Design Recipe Telephone Set 2: l-rect Design Recipe Telephone Set 2: right-trapezoid	Use any of the Design Recipe problems that students haven't solved before. ★*There is a large collection of math problems that would work well with the Design Recipe in the Additional Exercises section of our Solving Word Problems with the Design Recipe lesson.*

2. Describe the rules for the activity

- In this activity, each person in your group will start with a different word problem. You will each be doing *one step of each Design Recipe problem* . After you complete your step, you will fold your paper to hide the part that you were looking at so that only *your work and the rest of the recipe* are visible. Then you will pass your work to the person to your right.
- The person who has received your paper will review your work and complete the next step based solely on what you wrote down for them. If they don't have the information they need, they will give the paper back to you for revision.
- Meanwhile, you will receive a different problem from the person to your left. If at any point your realize that the person before you didn't provide enough information, you may hand the paper back to them for revision.

Who's Doing What During Each Round of Design Recipe Telephone?

Round 1 - Writing Contract and Purpose Statements from the Word Problem

Student 1 - Problem A	Student 2 - Problem B	Student 3 - Problem C

everyone folds over the previous section, and passes their paper to the right

Round 2 - Writing Examples *based solely on the Contract and Purpose Statement*

Student 1 - Problem C	Student 2 - Problem A	Student 3 - Problem B

everyone folds over the previous section, and passes their paper to the right

Round 3 - Writing Function Definitions *based solely on the Examples*

Student 1 - Problem B	Student 2 - Problem C	Student 3 - Problem A

3. Practice makes perfect!

This activity can be repeated several times, or done as a timed competition between teams. The goal is to emphasize that each step - if done correctly - makes the following step incredibly simple.

4. Synthesize

The Design Recipe is a way of slowing down and thinking through each step of a problem.
If we already know how to get the answer, why would it ever be important to know how to do each step the slow way?

- *Sample Responses: Someday we won't be able to get the answer, and knowing the steps will help. We can help someone else who is stuck. We can work with someone else and share our thinking. We can check our work.*

(optional)

The Design Recipe (Restaurants)

Directions: Use the Design Recipe to write a function `split-tab` that takes in a cost and the number of people sharing the bill and splits the cost equally.

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

Directions: Use the Design Recipe to write a function `tip-calculator` that takes in the cost of a meal and returns the 15% tip for that meal.

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

(optional)

The Design Recipe (Direct Variation)

Directions: Use the Design Recipe to write a function `wage`, that takes in a number of hours worked and returns the amount a worker will get paid if their rate is $10.25/hr.

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

Directions: On average, people burn about 11 calories/minute riding a bike. Use the Design Recipe to write a function `calories-burned` that takes in the number of minutes you bike and returns the number of calories burned. .

Contract and Purpose Statement

Every contract has three parts…

_____ :: *Number* -> *Number*
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

(optional)

The Design Recipe (Slope/Intercept)

Directions: For his birthday, James' family decided to open a savings account for him. He started with $50 and committed to adding $10 a week from his afterschool job teaching basketball to kindergartners. Use the Design Recipe to write a function `savings` that takes in the number of weeks since his birthday and calculates how much money he has saved.

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

Directions: Use the Design Recipe to write a function `moving` that takes in the days and number of miles driven and returns the cost of renting a truck. The truck is $45 per day and each driven mile is 15¢.

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

(optional)

The Design Recipe (Negative Slope/Intercept)

Directions: An Olympic pool holds 660,000 gallons of water. A fire hose can spray about 250 gallons per minute. Use the Design Recipe to write a function `pool` that takes in the number of minutes that have passed and calculates how much water is still needed to fill it.

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

Directions: The community arts fund awards a $1500 grant each month to support a new mural. They started with $50000 in their account. Use the Design Recipe to write a function `funds-available` that takes in the number of months and calculates how much money they have left.

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

(optional)

The Design Recipe (Geometry - Rectangles)

Directions: Use the Design Recipe to write a function `lawn-area` that takes in the length and width of a rectangular lawn and returns its area.

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

Directions: Use the Design Recipe to write a function `rect-perimeter` that takes in the length and width of a rectangle and returns the perimeter of that rectangle.

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

(optional)

The Design Recipe (Geometry - Rectangular Prisms)

Directions: Use the Design Recipe to write a function `rectprism-vol` that takes in the length, width, and height of a rectangular prism and returns the Volume of a rectangular prism.

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

Directions: Use the Design Recipe to write a function `rect-prism-sa` that takes in the width, length and height of a rectangular prism and calculates its surface area (the sum of the areas of each of its six faces)

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

(optional)

The Design Recipe (Geometry - Circles)

Directions: Use the Design Recipe to write a function `circle-area-dec` that takes in a radius and uses the decimal approximation of pi (3.14) to return the area of the circle.

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

Directions: Use the Design Recipe to write a function `circumference` that takes in a radius and uses the decimal approximation of pi (3.14) to return the circumference of the circle.

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

(optional)

The Design Recipe (Geometry - Cylinders)

Directions: Use the Design Recipe to write a function `circle-area` that takes in a radius and uses the fraction approximation of pi ($^{22}/_{7}$) to return the area of the circle.

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

Directions: Use the Design Recipe to write a function `cylinder` that takes in a cylinder's radius and height and calculates its volume, making use of the function *circle-area* .

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

(optional)

The Design Recipe (Breaking Even)

Directions: The Swamp in the City Festival is ordering t-shirts. The production cost is $75 to set up the silk screen and $9 per shirt. Use the Design Recipe to write a function `min-shirt-price` that takes in the number of shirts to be ordered, *n*, and returns the minimum amount the festival should charge for the shirts in order to break even. (Assume that they will sell all of the shirts.)

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

(optional)

The Design Recipe (Marquee & Cubing)

Directions: Use the Design Recipe to write a function `marquee` that takes in a message and returns that message in large gold letters.

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

Directions: Use the Design Recipe to write a function `num-cube` that takes in a number and returns the cube of that number.

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

(optional)

Design Recipe Telephone Set 1: g

Directions: Hali is decorating her tree house and is having a hard time fitting everything on the walls. She's figured out that if her artwork were 3/8 of the original size it would all fit. Help her by writing a function `g` to scale down any image to a size she can use!

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

**★NOTE★ When writing examples, you can assume that we have predefined* `image-a` *and* `image-b`.*

(optional)

Design Recipe Telephone Set 1: h

Directions: Define a function h that will take an image and rotate it clockwise one-tenth of a turn. Hint: A full rotation is 360 degrees, which you may have heard people refer to in skateboarding or snowboarding tricks.

Contract and Purpose Statement

Every contract has three parts…

_____ :: _____ -> _____
function name — Domain (`Image`) — Range (`Image`)

_____
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

**★NOTE★ When writing examples, you can assume that we have predefined* `image-a` *and* `image-b`.*

(optional)

Design Recipe Telephone Set 1: r

A Contract worth remembering…

```
# regular-polygon :: Number, Number, String, String -> Image
# Takes in a size, the number of sides, a color, and a fill type and makes a shape with all equal sides and all angles congruent.
```

Directions: Zora's favorite shape is a regular pentagon and they want to decorate a special box with pentagons of every color. Help them to realize their dream by writing a function `r` that takes in a color and returns a solid 5-sided regular polygon of size 300 in the given color.

Contract and Purpose Statement

Every contract has three parts…

________________ :: *String* -> *Image*

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

____

(optional)

Design Recipe Telephone Set 2: symmetry

**★NOTE★When writing examples, you can assume that we have predefined* `image-a` *and* `image-b`.*

Directions: Nassim loves all things symmetrical. He figured out that if you flip an image horizontally and then place it beside the original image, you can turn any image into a symmetrical image. Help him to be more efficient by writing a new function `symmetry` that will take in any image and use it to make a new symmetrical image.

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

A Contract worth remembering:

```
# beside :: Image, Image -> Image
# places two images beside each other
```

(optional)

Design Recipe Telephone Set 2: l-rect

Directions: Ava loves purple rectangles that are 5 times as wide as they are tall. Help her out by writing a function `l-rect` that takes in a width and generates a solid rectangle that Ava would love.

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

(optional)

Design Recipe Telephone Set 2: right-trapezoid

★NOTE★An isosceles triangle has two sides that are the same length.

Directions: Zosia loves right-trapezoids composed of squares and isosceles-right-triangles. Write a function `right-trapezoid` that takes in the sidelength of the square and a color and returns a solid right-trapezoid.

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

A Contract worth remembering:

```
# right-triangle :: Number, Number, String, String -> Image
# Takes in 2 side lengths, a color, and a fill type and makes a right-triangle
```

(optional)

Intro to Data Structures

Word Problem: double-radius

Directions: Write a function `double-radius`, which takes in a radius and a color. It produces an outlined circle of whatever color was passed in, whose radius is twice as big as the input.

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: double-width

Directions: Write a function `double-width`, which takes in a number (the length of a rectangle) and produces a rectangle whose length is twice the given length.

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: next-position

Directions: Write a function `next-position`, which takes in two numbers (an x- and y-coordinate) and returns a DeliveryState, increasing the x-coordinate by 5 and decreasing the y-coordinate by 5.

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

Data Structure: CakeType

```
# A CakeType is a flavor, layers, & is-iceCream
data CakeType:

  | cake(_____________________________________________________________

         _____________________________________________________________

         _____________________________________________________________)
end
```

1) To make an instance of this structure, I would write:

```
cake1 = _____________________________________________________________________

cake2 = _____________________________________________________________________
```

2) To access the fields of `cake2`, I would write:

```
_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________
```

Word Problem: taller-than

Directions: Write a function called `taller-than`, which consumes two CakeTypes, and produces true if the number of layers in the first CakeType is greater than the number of layers in the second.

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: will-melt

Directions: Write a function called `will-melt`, which takes in a CakeType and a temperature, and returns true if the temperature is greater than 32 degrees, AND the CakeType is an ice-cream cake.

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

Vocabulary Practice

Below is a new structure definition:

```
data MediaType:
  | book(
      title :: String,
      author :: String,
      pubyear :: Number)
end

# an example book:
book1 = book("1984", "Orwell", 1949)
```

Fill in the blanks below with the vocabulary term that applies to each name. Here are the terms to choose from:

contract	example
header	field
data type	instance
constructor	data block
name	purpose

`author` is a _____

`book` is a _____

`MediaType` is a _____

`book1` is a _____

`title` is a _____

`data` `…` **`end`** is a _____

(optional)

Structures, Reactors, & Animations

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Design a Data Structure

```
# a ________________State is ________________________________________
data ________________State:
    | ____________( ______________________________________________________
                                        ______________________________________________________
                                        ______________________________________________________
end
```

Make a sample instance for each sketch from the previous page:

```
          sketchA            = _____________________________________________________

          sketchB            = _____________________________________________________

          sketchC            = _____________________________________________________
```

Word Problem: draw-state

Write a function called *draw-state* , which takes in a `SunsetState` and returns an image in which the sun (a circle) appears at the position given in the `SunsetState`. The sun should be behing the horizon (the ground) once it is low in the sky.

Contract and Purpose Statement

draw-state :: _____ -> Image

_____

Write an expression for each piece of your final image

SUN =	
GROUND =	
SKY =	

Write the draw-state function, using `translate` to combine your pieces

fun _____(_____):

_____

_____

_____ end

Word Problem: next-state-tick

Directions: Write a function called `next-state-tick`, which takes in a SunsetState and returns a SunsetState in which the new x-coordinate is 8 pixels larger than in the given SunsetState and the y-coordinate is 4 pixels smaller than in the given SunsetState.

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

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Design a Data Structure

```
# a ________________State is ________________________________________
data ________________State:
    | ____________( _______________________________________________________
                                        _______________________________________________________
                                        _______________________________________________________
end
```

Make a sample instance for each sketch from the previous page:

`__________sketchA__________ = _____________________________________________________`

`__________sketchB__________ = _____________________________________________________`

`__________sketchC__________ = _____________________________________________________`

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Design a Data Structure

```
# a ________________State is ________________________________________
data ________________State:
    | ____________( ______________________________________________________
                                        ______________________________________________________
                                        ______________________________________________________
end
```

Make a sample instance for each sketch from the previous page:

`__________sketchA__________ = ____________________________________________________`

`__________sketchB__________ = ____________________________________________________`

`__________sketchC__________ = ____________________________________________________`

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Design a Data Structure

```
# a _______________State is _______________________________________
data _______________State:
    | ___________( ____________________________________________________
                                        ____________________________________________________
                                        ____________________________________________________
end
```

Make a sample instance for each sketch from the previous page:

```
sketchA = ___________________________________________________

sketchB = ___________________________________________________

sketchC = ___________________________________________________
```

Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Data Type (Number, String, Image, Boolean ...)

Design a Data Structure

```
# a ________________State is ________________________________________
data ________________State:
    | ____________( ___________________________________________________
                    ___________________________________________________
                    ___________________________________________________
end
```

Make a sample instance for each sketch from the previous page:

```
sketchA = ______________________________________________________

sketchB = ______________________________________________________

sketchC = ______________________________________________________
```

Functions That Ask Questions

Word Problem: location

Directions: Write a function called `location`, which consumes a DeliveryState, and produces a String representing the location of a box: either "road", "delivery zone", "house", or "air".

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

_____(_____) **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

Syntax and Style Bug Hunting: Piecewise Edition

	Buggy Code	Correct Code / Explanation

1	`fun piecewisefun(n):`
	` if (n > 0): n`
	` else: 0`	

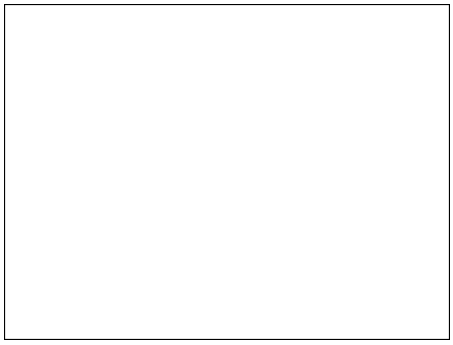
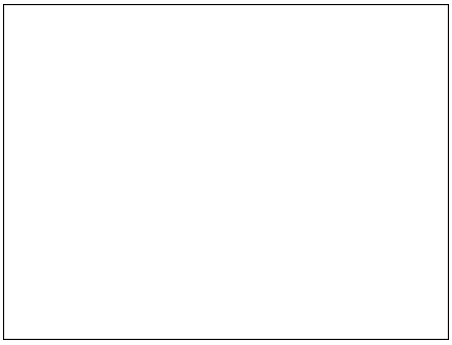
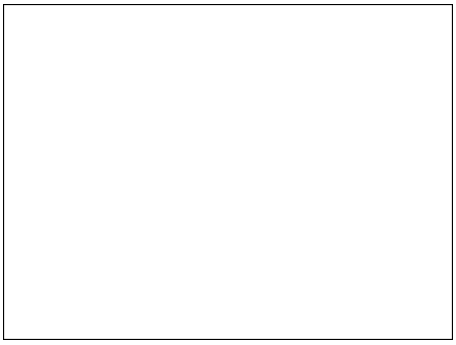
2	`fun cost(topping):`
	` if string-equal(topping,`
	`"pepperoni"): 10.50`
	` else string-equal(topping,`
	`"cheese"): 9.00`
	` else string-equal(topping,`
	`"chicken"): 11.25`
	` else string-equal(topping,`
	`"broccoli"): 10.25`
	` else: "That's not on the menu!"`
	` end`
	`end`	

3	`fun absolute-value(a b):`
	` if a > b: a - b`
	` b - a`
	` end`
	`end`	

4	`fun best-function(f):`
	` if string-equal(f, “blue”):`
	` “you win!”`
	` else if string-equal(f, “blue”):`
	` “you lose!”`
	` else if string-equal(f, “red”):`
	` “Try again!”`
	` else: “Invalid entry!”`
	` end`
	`end`	

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.
Draw a sketch for three distinct moments of the animation

		
Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

1) Make a sample instance for each sketch from the previous page:

_____ =

_____ =

_____ =

2) Write at least one NEW example for one of the functions on your To-Do list

3) If you have another function on your To-Do list, write at least one NEW example

Word Problem: draw-sun

Directions: Write a function called `draw-sun`, which consumes a SunsetState, and produces an image of a sun (a solid, 25 pixel circle), whose color is "yellow", when the sun's y-coordinate is greater than 225, "orange", when its y-coordinate is between 150 and 225, and "red" otherwise.

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

_____ (_____) **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

Key Events

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.
Draw a sketch for three distinct moments of the animation

		
Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

1) Make a sample instance for each sketch from the previous page:

FULLPET =

```
pet(100, 100)
```

MIDPET =

```
pet(50, 75)
```

LOSEPET =

```
pet(0, 0)
```

2) Write at least one NEW example for one of the functions on your To-Do list

next-state-tick(FULLPET) is pet(FULLPET.hunger – 2, FULLPET.sleep – 1)

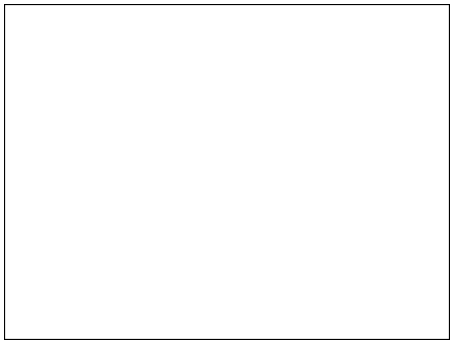
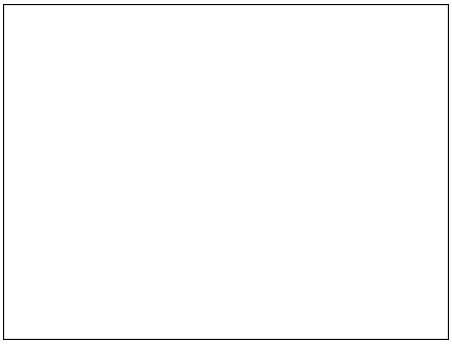
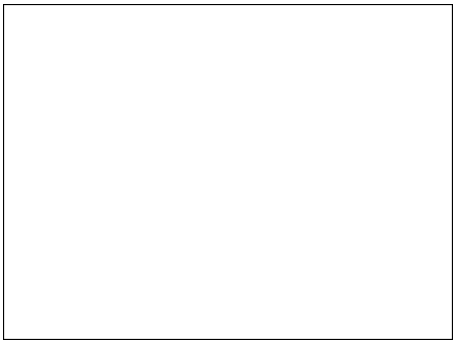
next-state-tick(MIDPET) is pet(MIDPET.hunger – 2, MIDPET.sleep – 1)

next-state-tick(LOSEPET) is LOSEPET

3) If you have another function on your To-Do list, write at least one NEW example

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.
Draw a sketch for three distinct moments of the animation

		
Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

1) Make a sample instance for each sketch from the previous page:

_____ =

_____ =

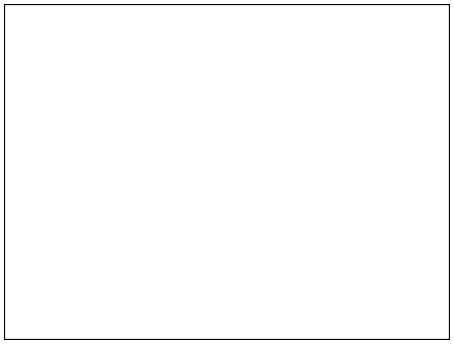
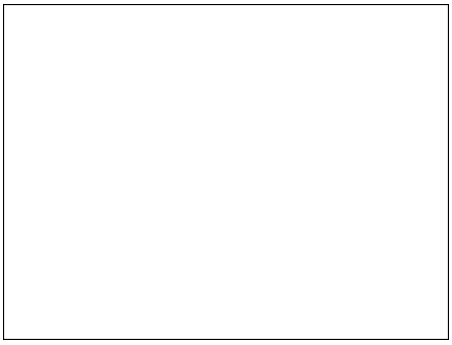
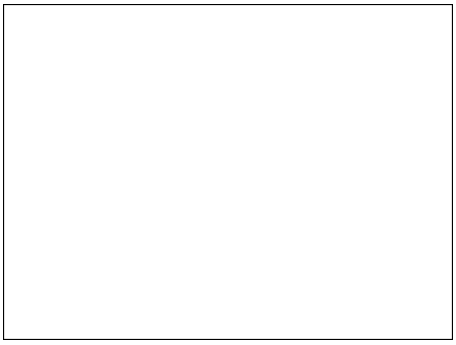
_____ =

2) Write at least one NEW example for one of the functions on your To-Do list

3) If you have another function on your To-Do list, write at least one NEW example

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.
Draw a sketch for three distinct moments of the animation

		
Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

1) Make a sample instance for each sketch from the previous page:

_____ =

_____ =

_____ =

2) Write at least one NEW example for one of the functions on your To-Do list

3) If you have another function on your To-Do list, write at least one NEW example

Refactoring

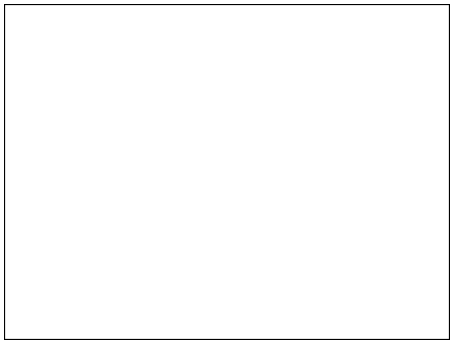
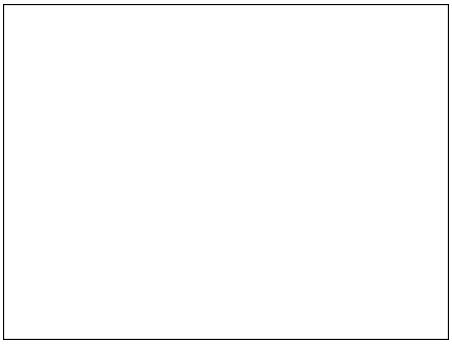
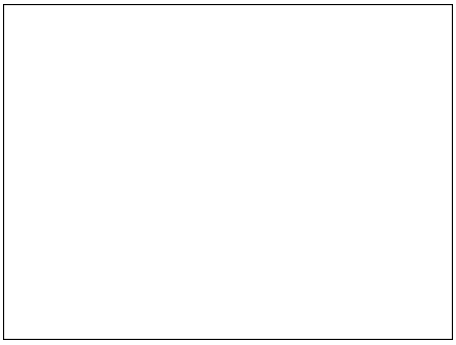
Your Own Drawing Functions

Build Your Own Animation

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation

		
Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	data type (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

Define the Data Structure

```
# a _____________State is ____________________________________________

data _____________State:

  | _________________ (_____________________________________________________

                       _____________________________________________________

                       _____________________________________________________

                       _____________________________________________________)
end
```

1) Make a sample instance for each sketch from the previous page

```
_____________ = ____________________________________________________

_____________ = ____________________________________________________

_____________ = ____________________________________________________
```

2) Write an example for one of the functions on the previous page

```
_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________

_____________________________________________________________________
```

Line Length Explore

Sign in to code.pyret.org (CPO) and open your Game File.

Defining `line-length`

Find the definition for the `line-length` *function and consider the code you see.*

1) What do you Notice?

2) What do you Wonder?

Using `line-length`

Click Run, and practice using `line-length` *in the* ***Interactions Area*** *with different values for* a *and* b.

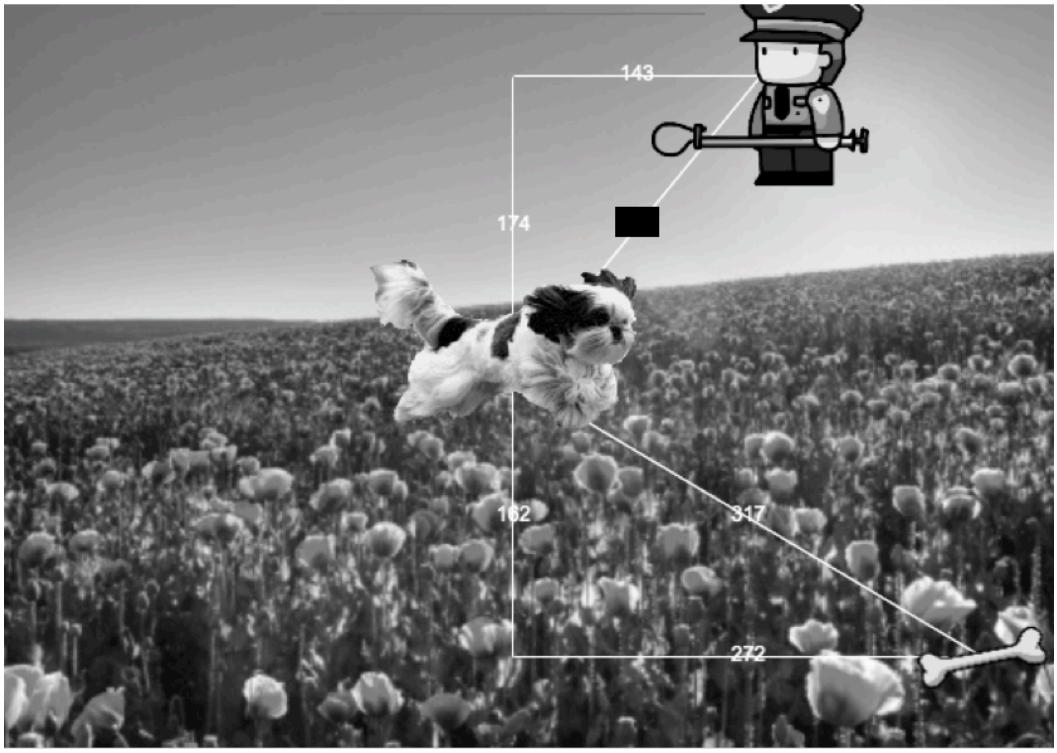
3) What does the `line-length` function *do?*

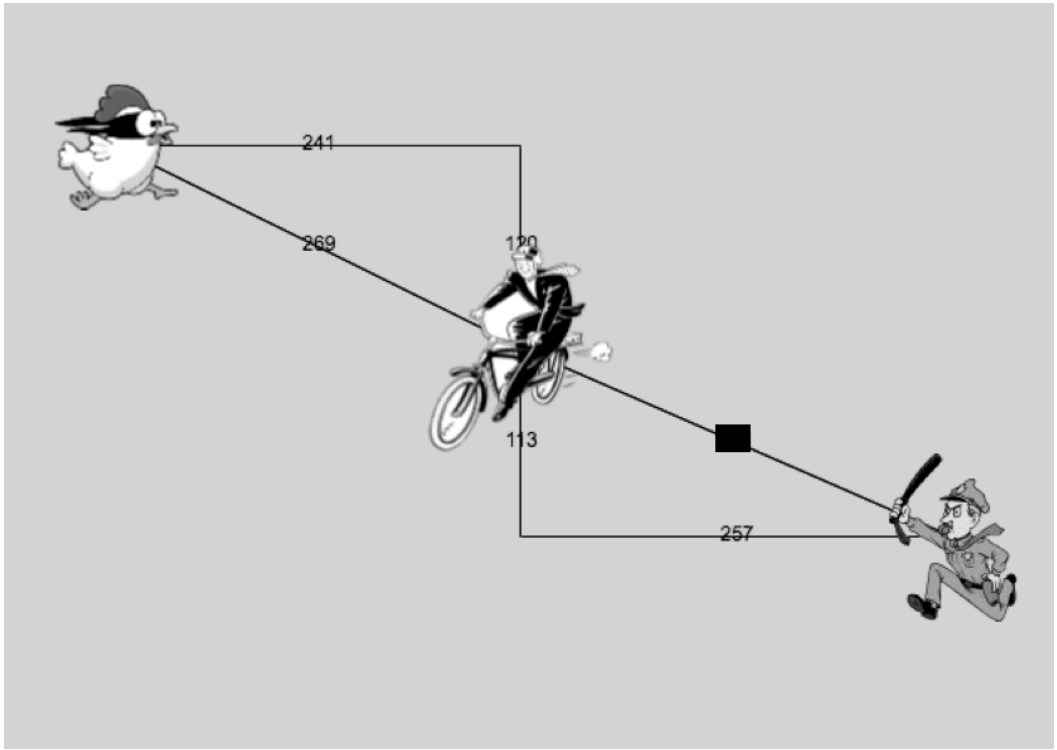
4) Why does it use conditionals?

5) Why is the distance between two points always positive?

Writing Code to Calculate Missing Lengths

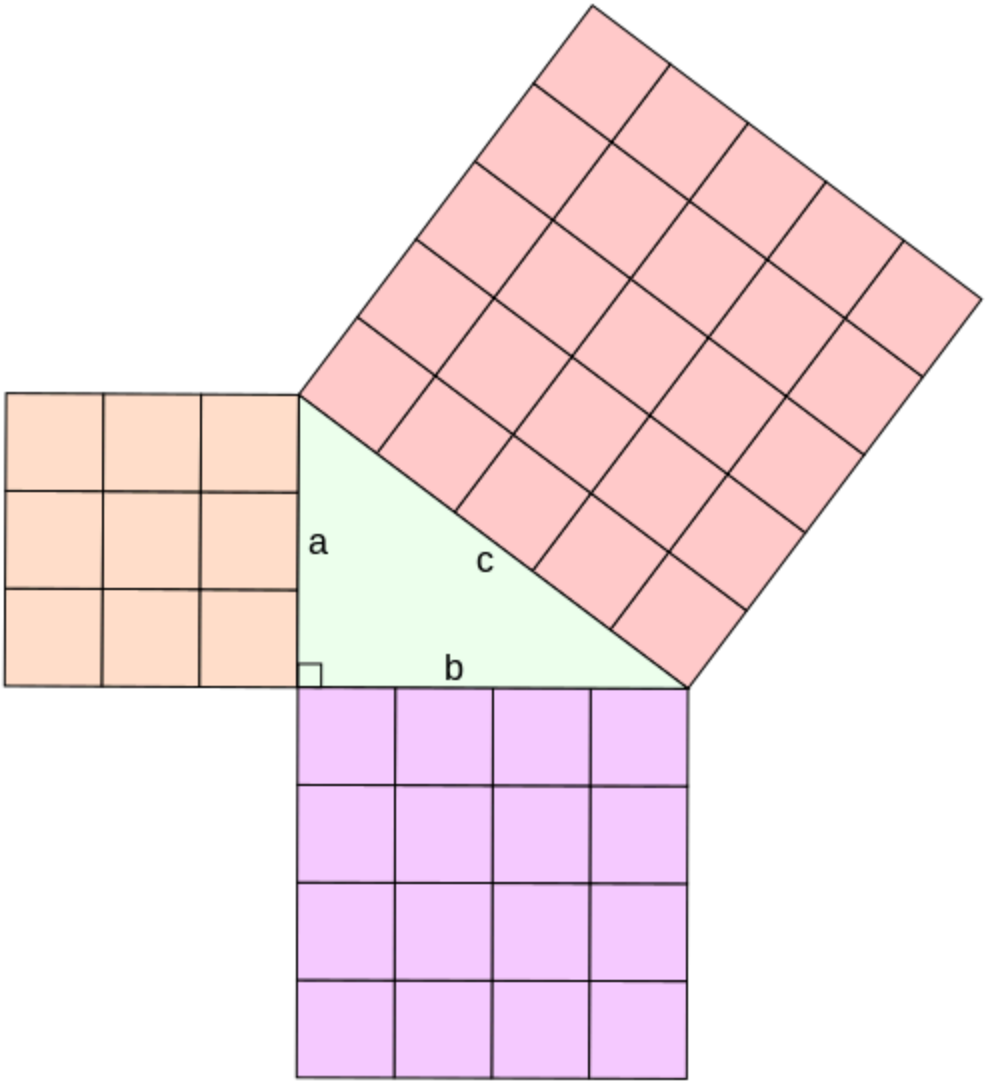
In each of the game screenshots below, one of the distance labels has been hidden. Write the code to generate the missing distance on the line below each image. *Hint: Remember the Pythagorean Theorem!*





Proof Without Words

Long ago, mathematicians realized that there is a special relationship between the three squares that can be formed using the sides of a right triangle.



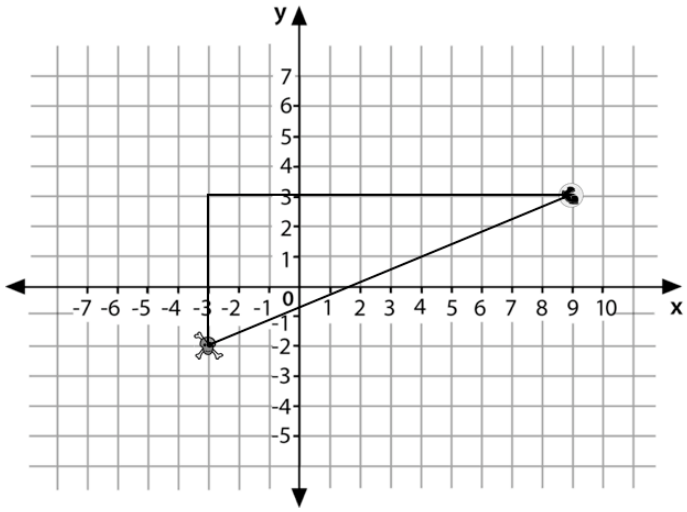
How would you describe the relationship you've observed between the three squares whose side-lengths are determined by the lengths of the sides of a right triangle?

Distance on the Coordinate Plane

Reading Code:

Distance between the Pyret and the boot:

```
sqrt(sqr(line-length(9, -3)) + sqr(line-length(3, -2)))
```



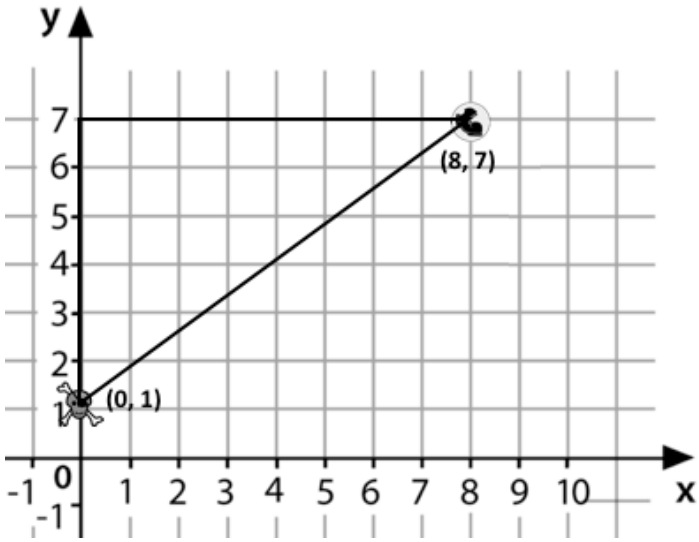
1) Where do the 9 and -3 come from? _____

2) Where to the 3 and -2 come from? _____

3) Explain how the code works. _____

_____

Writing Code



Now write the code to find the distance between this boot and pyret.

_____

_____

Circles of Evaluation: Distance between (0, 2) and (4, 5)

Suppose your player is at (0, 2) and a character is at (4, 5)...

1) Identify the values of x_1, y_1, x_2, and y_2

x_1	y_1	x_2	y_2
(x-value of 1st point)	*(y-value of 1st point)*	*(x-value of 2nd point)*	*(y-value of 2nd point)*

What is the distance between your player and the character?

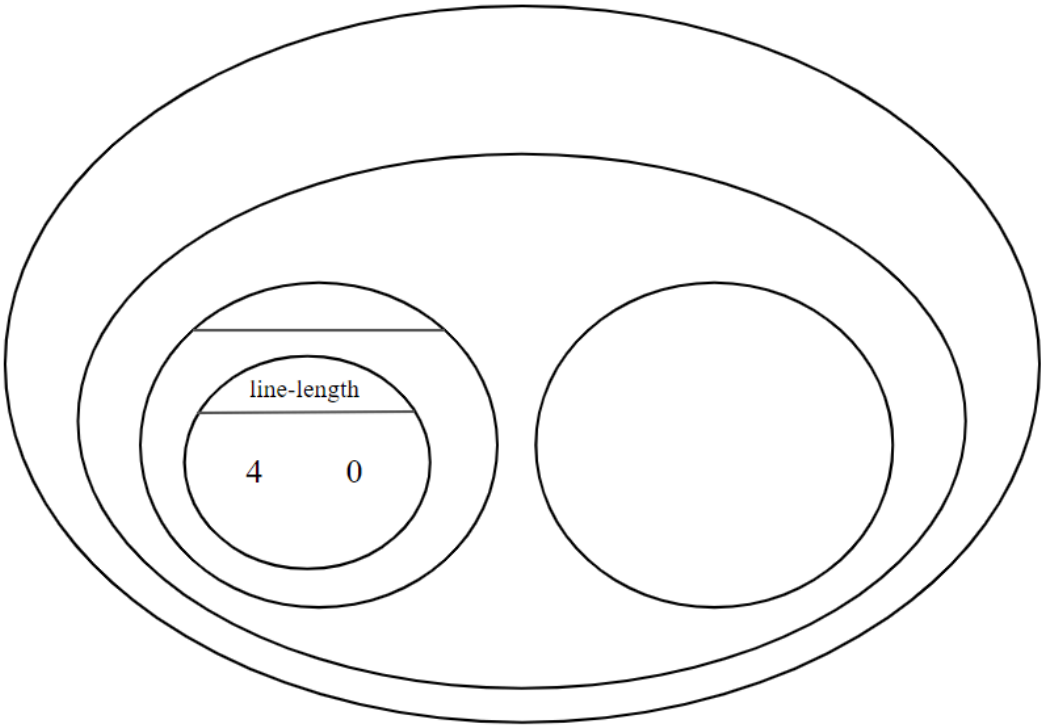
- We can use `line-length` to computer the horizontal and vertical distances and then use those to find the diagonal distance.
  - The horizontal distance between x_1 and x_2 is computed by `line-length(x2, x1)`.
  - The vertical distance between y_2 and y_1 is computed by `line-length(y2, y1)`.
- The hypotenuse of a right triangle with legs the lengths of those distances is computed by: $\sqrt{\text{line-length}(x_2, x_1)^2 + \text{line-length}(y_2, y_1)^2}$
- So, when we substitute these points in, the distance between them will be computed by:

$$\sqrt{\text{line-length}(4, 0)^2 + \text{line-length}(5, 2)^2}$$

2) The points are (0,2) and (4,5). Why aren't we using `line-length(0, 2)` and `line-length(4, 5)`?

---

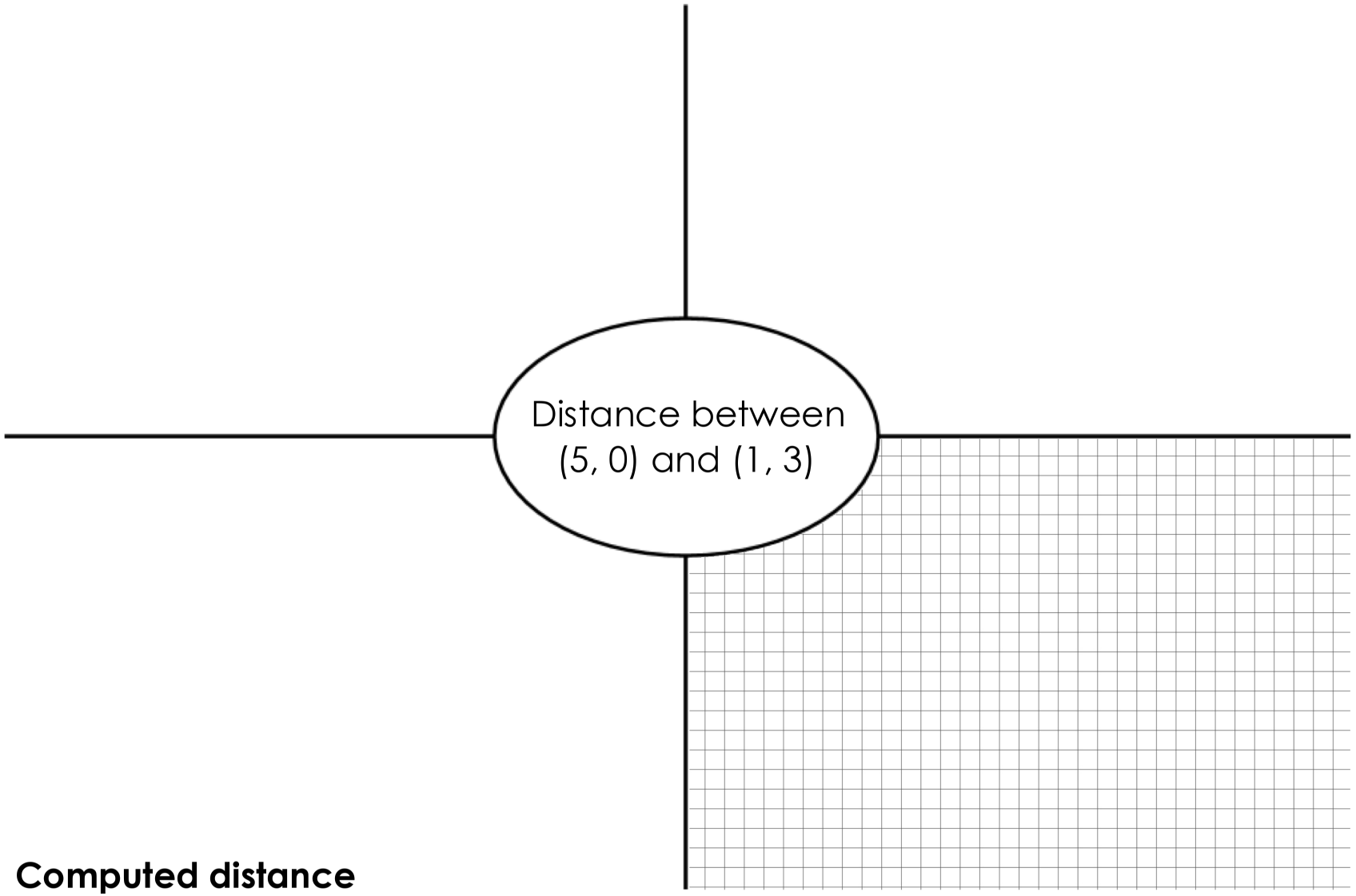
3) Translate the expression above, for (0,2) and (4,5) into a Circle of Evaluation below.
Hint: In our programming language `sqr` *is used for* x^2 *and* `sqrt` *is used for* $\sqrt{x}$



4) Convert the Circle of Evaluation to Code below.

Circle of Evaluation

Code

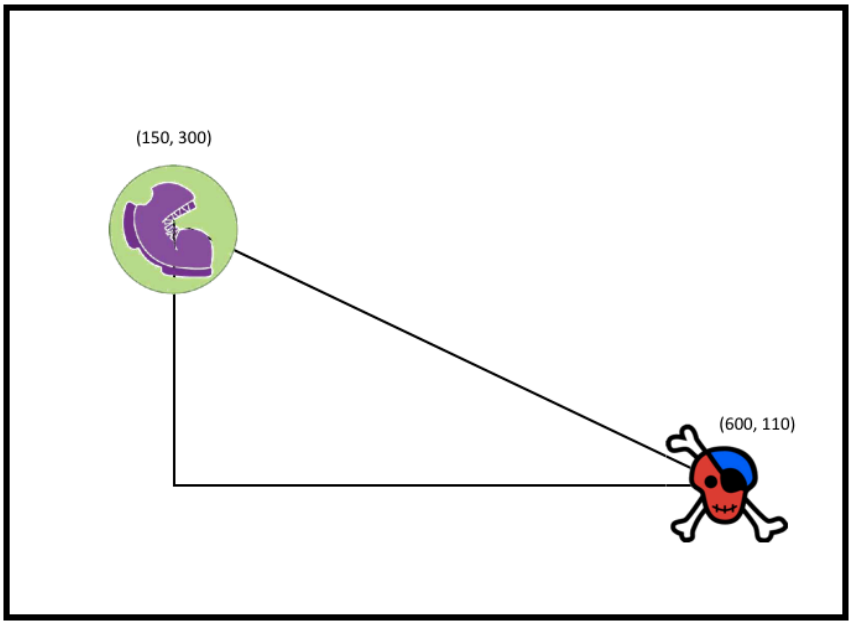


Computed distance between (5, 0) and (1, 3)

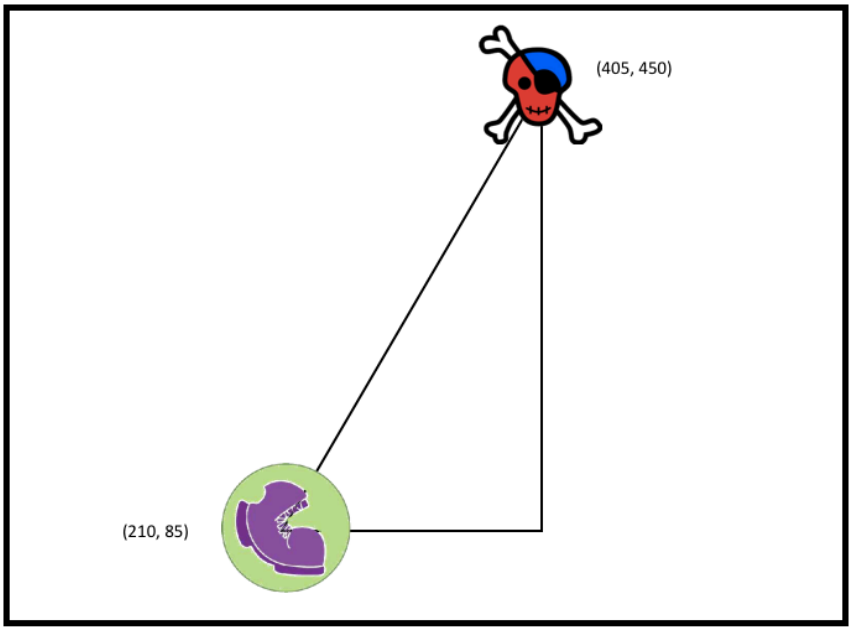
Graph

Distance From Game Coordinates

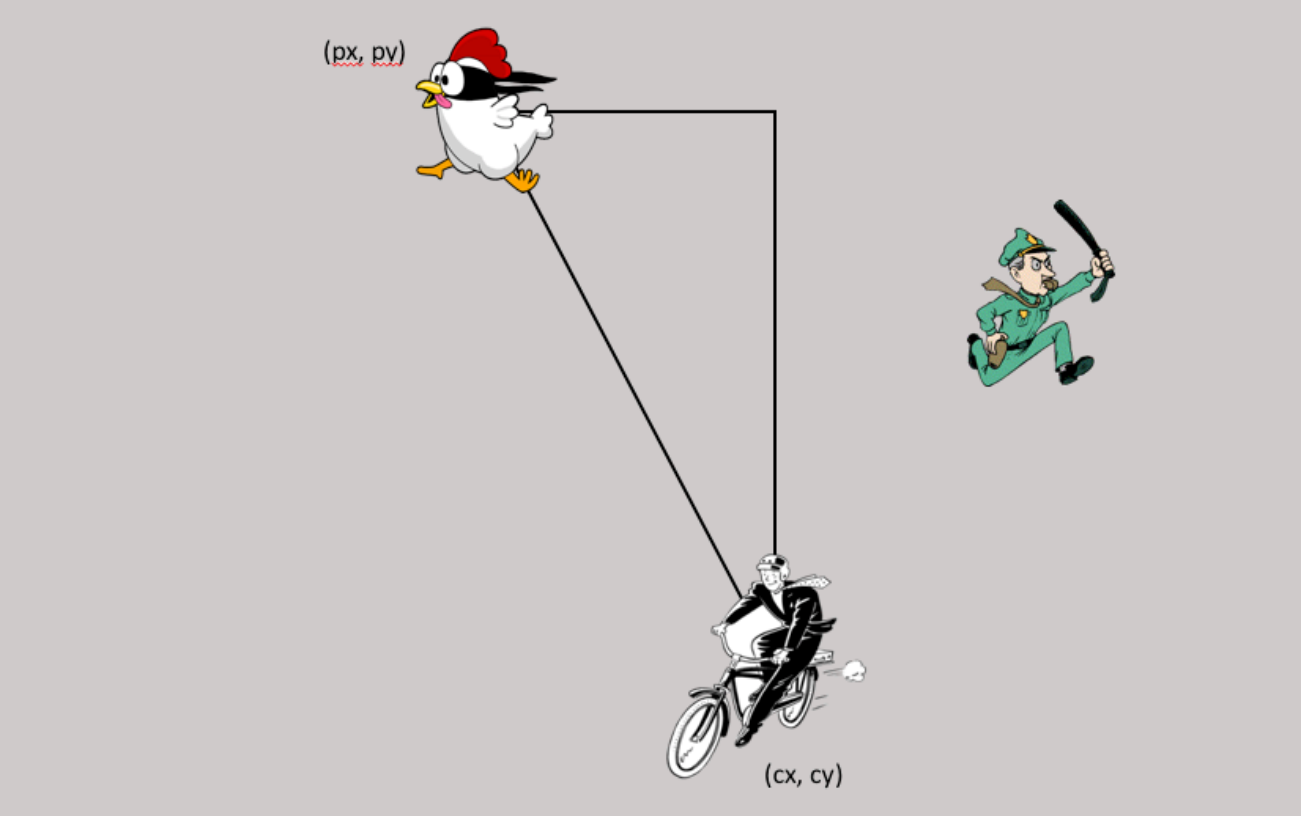
For each of the game screenshots, write the code to calculate the distance between the indicated characters. *The first one has been done for you.*




```
sqrt(sqr(line-length(600, 150)) + sqr(line-length(110, 300)))
```



Distance (px, py) to (cx, cy)



Directions: Use the Design Recipe to write a function `distance`, which takes in FOUR inputs: `px` and `py` (the x- and y-coordinate of the Player) and `cx` and `cy` (the x- and y-coordinates of another character), 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:

_____ (_____) **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…

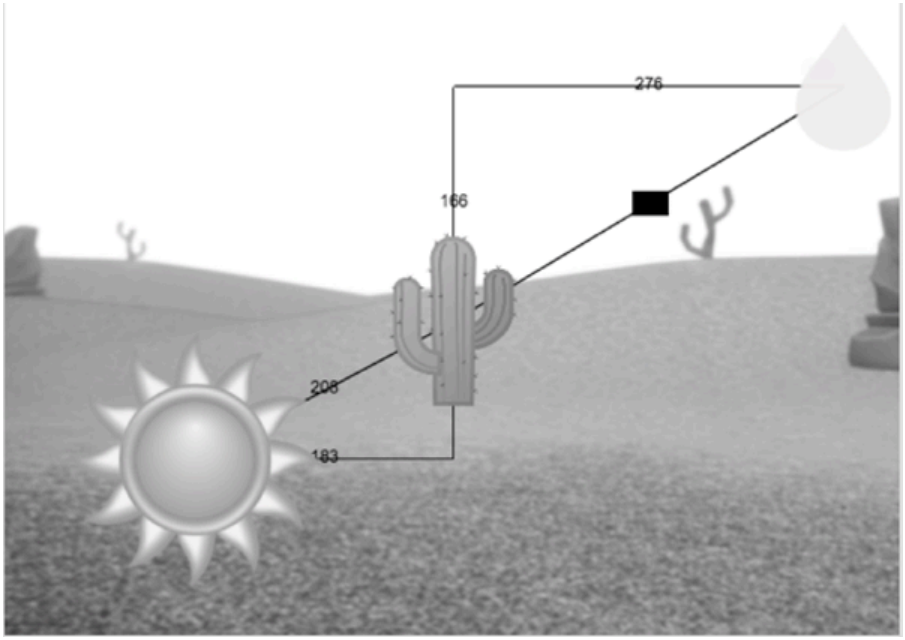
fun _____ (_____):
function name | variable(s)

_____
what the function does with those variable(s)

end

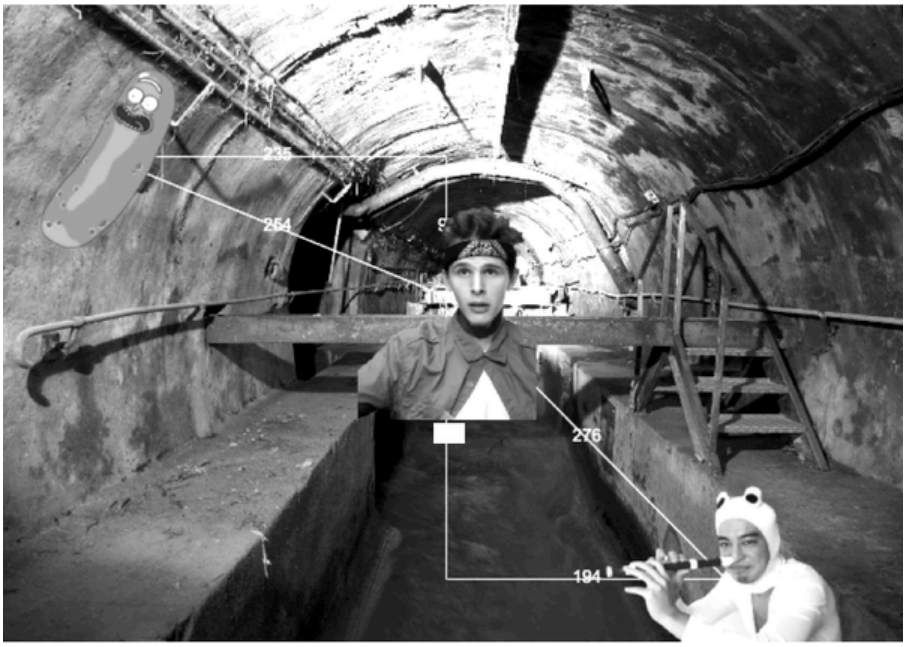
Comparing Code: Finding Missing Distances

For each of the game screenshots below, the math and the code for computing the covered distance is shown. Notice what is similar and what is different about how the top and bottom distances are calculated. Think about why those similarities and differences exist and record your thinking.



$\sqrt{166^2 + 276^2}$

```
sqrt(sqr(166) + sqr(276))
```



$\sqrt{276^2 - 194^2}$

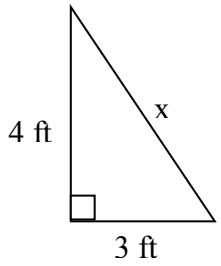
```
sqrt(sqr(276) - sqr(194))
```

Name:_____ Date: _____ Pythagorean Theorem Practice

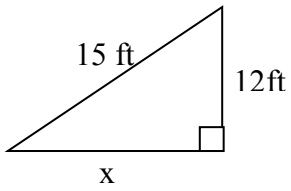
Label the hypotenuse of the triangle c. In each triangle find the length of the side marked *x* to the nearest unit (foot, cm, etc.). Show your work.

$$a^2 + b^2 = c^2$$

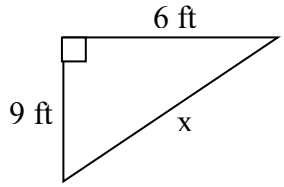
1.



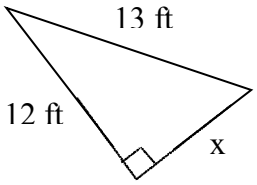
2.



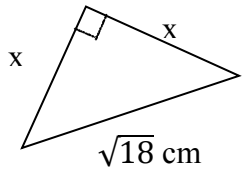
3.



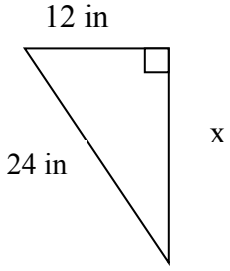
4.



5.

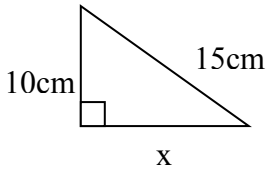


6.

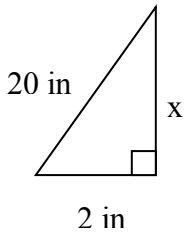


(optional)

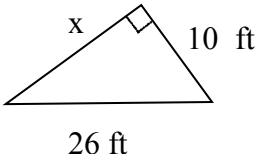
7.



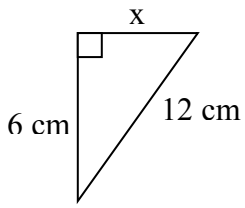
8.



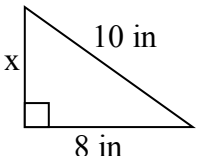
9.



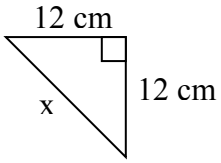
10.



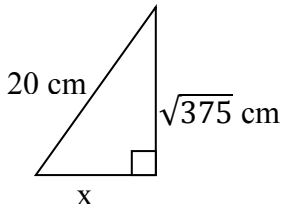
11.



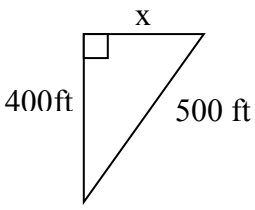
12.



13.



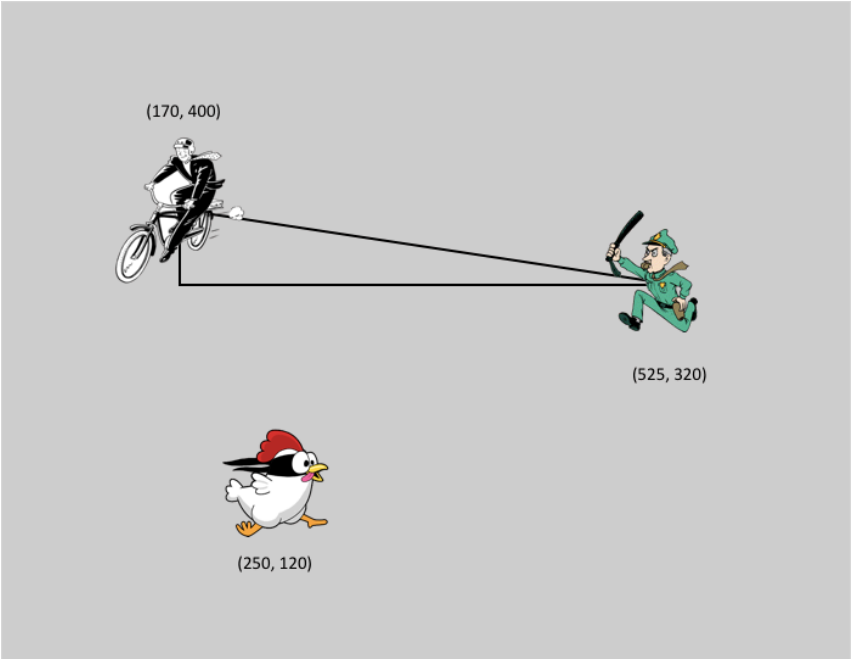
14.

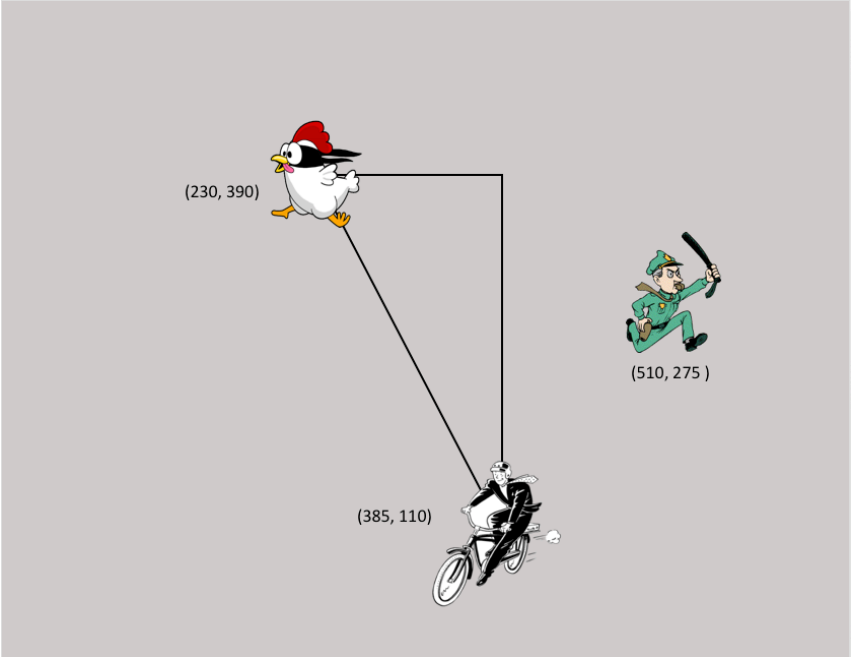


(optional)

Distance From Game Coordinates 2

For each of the game screenshots below, write the code to calculate the distance between the indicated characters. *Refer to Distance from Game Coordinates for an Example.*





(optional)

Word Problem: line-length

Directions: Write a function called `line-length`, which takes in two numbers and returns the **positive difference** between them. It should always subtract the smaller number from the bigger one. If they are equal, it should return zero.

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:

line-length(*10, 5*) **is** *10 - 5*
function name | input(s) | what the function produces

line-length(*2, 8*) **is** *8 - 2*
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)

if _____ **:** _____

else: _____ **:** _____

end

end

(optional)

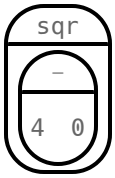
Collisions

Distance

The Player is at (4, 2) and the Target is at (0, 5).
Distance takes in the player's x, player's y, character's x and character's y. Use the formula below to fill in the EXAMPLE:

$$\sqrt{(4-0)^2+(2-5)^2}$$

Convert it into a Circle of Evaluation. (We've already gotten you started!)



Convert it to Pyret code.

Word Problem: distance

Directions: Write a function `distance`, which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) cy: The y-coordinate of another game character. It should return the distance between the two, using the Distance formula: $\text{Distance}^2 = (px - cx)^2 + (py - cy)^2$

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: is-collision

Directions: Write a function `is-collision`, which takes FOUR inputs: (1) px: The x-coordinate of the player, (2) py: The y-coordinate of the player, (3) cx: The x-coordinate of another game character, (4) cy: The y-coordinate of another game character. It should return true if the coordinates of the player are within **50 pixels** of the coordinates of the other character. Otherwise, false.

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

Notes

Making Pong

Nested Structures

Non-Nested Pinwheels Code

```
# A PinwheelState is the angle of rotation for 4 pinwheels
data PinwheelState:
  | pinwheels(
      p1a :: Number,
      p2a :: Number,
      p3a :: Number,
      p4a :: Number)
end

STARTING-PINWHEELS = pinwheels(60, 3, 25, 70)

# update-pinwheel :: Number -> Number
fun update-pinwheel(angle):
  angle + 6
end

# next-state-tick :: PinwheelState -> PinwheelState
fun next-state-tick(ps):
  pinwheels(
    update-pinwheel(ps.p1a),
    update-pinwheel(ps.p2a),
    update-pinwheel(ps.p3a),
    update-pinwheel(ps.p4a))
end

# draw-pinwheel :: Number -> Image
fun draw-pinwheel(angle):
  rotate(angle, PINWHEEL-IMG)
end

# draw-state :: PinwheelState -> Image
fun draw-state(ps):
  translate(draw-pinwheel(ps.p1a),
    400, 100,
    translate(draw-pinwheel(ps.p2a),
      320, 240,
      translate(draw-pinwheel(ps.p3a),
        100, 400,
        translate(draw-pinwheel(ps.p4a),
          500, 350,
          empty-scene(640, 480)))))
end
```

Nested Pinwheels Code

```
# A Pinwheel is an angle of rotation
data Pinwheel:
  | pw(angle :: Number)
end

# A PinwheelState is 4 Pinwheels
data PinwheelState:
  | pinwheels(
      p1 :: Pinwheel,
      p2 :: Pinwheel,
      p3 :: Pinwheel,
      p4 :: Pinwheel)
end

STARTING-PINWHEELS = pinwheels(pw(60), pw(3), pw(25), pw(70))

# update-pinwheel :: Pinwheel -> Pinwheel
fun update-pinwheel(p):
  pw(p.angle + 6)
end

# next-state-tick :: PinwheelState -> PinwheelState
fun next-state-tick(ps):
  pinwheels(
    update-pinwheel(ps.p1),
    update-pinwheel(ps.p2),
    update-pinwheel(ps.p3),
    update-pinwheel(ps.p4))
end

# draw-pinwheel :: Pinwheel -> Image
fun draw-pinwheel(p):
  rotate(p.angle, PINWHEEL-IMG)
end

# draw-state :: PinwheelState -> Image
fun draw-state(ps):
  translate(draw-pinwheel(ps.p1),
    400, 100,
    translate(draw-pinwheel(ps.p2),
      320, 240,
      translate(draw-pinwheel(ps.p3),
        100, 400,
        translate(draw-pinwheel(ps.p4),
          500, 350,
          empty-scene(640, 480)))))
end
```

Nested Pinwheels Code (2)

```
# A Pinwheel is an angle of rotation and a speed
data Pinwheel:
  | pw(angle :: Number, speed :: Number)
end

# A PinwheelState is 4 Pinwheels
data PinwheelState:
  | pinwheels(
      p1 :: Pinwheel,
      p2 :: Pinwheel,
      p3 :: Pinwheel,
      p4 :: Pinwheel)
end

STARTING-PINWHEELS = pinwheels(
  pw(60, 6),
  pw(3, 12),
  pw(25, 24),
  pw(70, -48))

# update-pinwheel :: Pinwheel -> Pinwheel
fun update-pinwheel(p):
  pw(p.angle + p.speed, p.speed)
end

# next-state-tick :: PinwheelState -> PinwheelState
fun next-state-tick(ps):
  pinwheels(
    update-pinwheel(ps.p1),
    update-pinwheel(ps.p2),
    update-pinwheel(ps.p3),
    update-pinwheel(ps.p4))
end

# draw-pinwheel :: Pinwheel -> Image
fun draw-pinwheel(p):
  rotate(p.angle, PINWHEEL-IMG)
end

# draw-state :: PinwheelState -> Image
fun draw-state(ps):
  translate(draw-pinwheel(ps.p1),
    400, 100,
    translate(draw-pinwheel(ps.p2),
      320, 240,
      translate(draw-pinwheel(ps.p3),
        100, 400,
        translate(draw-pinwheel(ps.p4),
          500, 350,
          empty-scene(640, 480)))))
end
```

Timers

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

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

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Datatype (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

Define the Data Structure

a _____State is _____ data _____State: | _____ (

_____

_____

_____

_____) end

Make a sample instance for each sketch from the previous page

_____ = _____ _____ =

_____ _____ =

_____

Write an example for one of the functions on the previous page

_____

_____

_____

_____

_____

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Datatype (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

Define the Data Structure

a _____State is _____ data _____State: | _____ (

_____

_____

_____

_____) end

Make a sample instance for each sketch from the previous page

_____ = _____ _____ =

_____ _____ =

_____

Write an example for one of the functions on the previous page

_____

_____

_____

_____

_____

Animation Data Worksheet

Decrease the cat's hunger level by 2 and sleep level by 1 on each tick.

Draw a sketch for three distinct moments of the animation

Sketch A	**Sketch B**	**Sketch C**

What things are changing?

Thing	Describe how it changes

What fields do you need to represent the things that change?

Field name (dangerX, score, playerIMG ...)	Datatype (Number, String, Image, Boolean ...)

Make a To-Do List, and check off each as "Done" when you finish each one.

Component	***When is there work to be done?***	To-Do	Done
Data Structure	*If any new field(s) were added, changed, or removed*	☐	☐
draw-state	*If something is displayed in a new way or position*	☑	☐
next-state-tick	*If the Data Structure changed, or the animation happens automatically*	☐	☐
next-state-key	*If the Data Structure changed, or a keypress triggers the animation*	☐	☐
reactor	*If either next-state function is new*	☐	☐

Define the Data Structure

a _____State is _____ data _____State: | _____ (

_____

_____

_____

_____) end

Make a sample instance for each sketch from the previous page

_____ = _____ _____ =

_____ _____ =

_____

Write an example for one of the functions on the previous page

_____

_____

_____

_____

_____

Contracts for Reactive

Contracts tell us how to use a function, by telling us three important things:

1. The **Name**
2. The **Domain** of the function - what kinds of inputs do we need to give the function, and how many?
3. The **Range** of the function - what kind of output will the function give us back?

For example: The contract `triangle :: (Number, String, String) -> Image` tells us that the name of the function is `triangle`, it needs three inputs (a Number and two Strings), and it produces an Image.
With these three pieces of information, we know that typing `triangle(20, "solid", "green")` will evaluate to an Image.

Name	Domain	Range
`# above`	`::  ( Image , Image )` (above, below)	`-> Image`
`above(circle(10, "solid", "black"), square(50, "solid", "red"))`		
`# beside`	`::  ( Image , Image )` (left, right)	`-> Image`
`beside(circle(10, "solid", "black"), square(50, "solid", "red"))`		
`# circle`	`::  ( Number , String , String )` (radius, fill-style, color)	`-> Image`
`circle(50, "solid", "purple")`		
`# ellipse`	`::  ( Number , Number , String , String )` (width, height, fill-style, color)	`-> Image`
`ellipse(100, 50, "outline", "orange")`		
`# expt`	`::  ( Number , Number )` (base, power)	`-> Number`
`expt(3, 4) # three to the fourth power`		
`# flip-horizontal`	`::  ( Image )`	`-> Image`
`flip-horizontal(text("Lion", 50, "maroon"))`		
`# flip-vertical`	`::  ( Image )`	`-> Image`
`flip-vertical(text("Orion", 65, "teal"))`		
`# isosceles-triangle`	`::  ( Number , Number , String , String )` (size, vertex-angle, fill-style, color)	`-> Image`
`isosceles-triangle(50, 20, "solid", "grey")`		
`# overlay`	`::  ( Image , Image )` (top, bottom)	`-> Image`
`overlay(circle(10, "solid", "black"), square(50, "solid", "red"))`		
`# radial-star`	`::  ( Num , Num , Num , Str , Str )` (points, outer, inner, fill-style, color)	`-> Image`
`radial-star(6, 20, 50, "solid", "red")`		
`# rectangle`	`::  ( Number , Number , String , String )` (width, height, fill-style, color)	`-> Image`
`rectangle(100, 50, "outline", "green")`		
`# regular-polygon`	`::  ( Number , Number , String , String )` (size, vertices, fill-style, color)	`-> Image`
`regular-polygon(25,5, "solid", "purple")`		

Name	Domain	Range
# rhombus	:: (Number (size) , Number (top-angle) , String (fill-style) , String (color))	-> Image
rhombus(100, 45, "outline", "pink")		
# right-triangle	:: (Number (leg1) , Number (leg2) , String (fill-style) , String (color))	-> Image
right-triangle(50, 60, "outline", "blue")		
# rotate	:: (Number (degrees) , Image (img))	-> Image
rotate(45, star(50, "solid", "dark-blue"))		
# scale	:: (Number (factor) , Image (img))	-> Image
scale(1/2, star(50, "solid", "light-blue"))		
# sqr	:: (Number)	-> Number
sqr(4)		
# sqrt	:: (Number)	-> Number
sqrt(4)		
# square	:: (Number (size) , String (fill-style) , String (color))	-> Image
square(50, "solid", "red")		
# star	:: (Number (radius) , String (fill-style) , String (color))	-> Image
star(50, "solid", "red")		
# star-polygon	:: (Number (size) , Number (point-count) , Number (step-count) , String (fill-style) , String (color))	-> Image
star-polygon(100, 10, 3 ,"outline", "red")		
# string-contains	:: (String (haystack) , String (needle))	-> Boolean
string-contains("hotdog", "dog")		
# string-length	:: (String)	-> Number
string-length("rainbow")		
# sum	:: (Table (table-name) , String (column))	-> Number
sum(animals-table, "pounds")		
# text	:: (String (message) , Number (size) , String (color))	-> Image
text("Zari", 85, "orange")		
# translate	:: (Image (front) , Number (x-coordinate) , Number (y-coordinate) , Image (behind))	-> Image
translate(circle(10, "solid", "black"), 10, 10, square(50, "solid", "red"))		
# triangle	:: (Number (size) , String (fill-style) , String (color))	-> Image
triangle(50, "solid", "fuchsia")		
# triangle-asa	:: (Number (top-left-angle) , Number (left-side) , Number (bottom-angle) , String (fill-style) , String (color))	-> Image
triangle-asa(90, 200, 10, "solid", "purple")		

Name	Domain	Range
# triangle-sas	:: (Number bottom-R-side , Number top-R-angle , Number top-side , String fill-style , String color) ->	Image

```
triangle-sas(50, 20, 70, "outline", "dark-green")
```

	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->
	::	->

These materials were developed partly through support of the National Science Foundation, (awards 1042210, 1535276, 1648684, and 1738598),

These materials were developed partly through support of the National Science Foundation (awards 1042210, 1535276, 1648684, and 1738598) and are licensed under a Creative Commons 4.0 Unported License. Based on a work at www.BootstrapWorld.org. Permissions beyond the scope of this license may be available by contacting contact@BootstrapWorld.org.