Name: _____


# Algebra

Fall 2025 Student Workbook - WeScheme Edition

Name: _____


# Algebra

Fall 2025 Student Workbook - WeScheme Edition

# Table of Contents

# 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?"

# The Math Inside video games

- Video games are all about *change!* How fast is this character moving? How does the score change if the player collects a coin? Where on the screen should we draw a castle?
- We can break down a game into parts, and figure out which parts change and which ones stay the same. For example:
  - Computers use **coordinates** to position a character on the screen. These coordinates specify how far from the left (x-coordinate) and the bottom (y-coordinate) a character should be. Negative values can be used to "hide" a character, by positioning them somewhere off the screen.
  - When a character moves, those coordinates change by some amount. When the score goes up or down, it *also* changes by some amount.
- From the computer's point of view, the whole game is just a bunch of numbers that are changing according to some equations. We might not be able to see those equations, but we can definitely see the effect they have when a character jumps on a mushroom, flies on a dragon, or mines for rocks!
- Modern video games are *incredibly* complex, costing millions of dollars and several years to make, and relying on hundreds of programmers and digital artists to build them. But building even a simple game can give us a good idea of how the complex ones work!

# Notice and Wonder

Write down what you Notice and Wonder about the Ninja Cat Game.
"Notices" should be statements, not questions. What stood out to you? What do you remember?

| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

# Reverse Engineer a video game

*This page is designed to be used with the Ninja Cat Game.*

What is changing in the game? What variables is the program keeping track of? The first example is filled in for you.


| Thing in the Game | What Changes About It? | More Specifically... what variable(s) are being tracked? |
|---|---|---|
| *Dog* | *Position* | *x-coordinate* |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# Estimating Coordinates


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

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

The coordinates for the TARGET (Ruby) are: (_____x, _____y)

# Brainstorm Your Own Game

Created by: _____

## Background

Our game takes place: _____
In space? The desert? A mall?

## Player

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

## Target

Your Player GAINS points when they hit The Target.

The Target is a _____
The Target moves only to the left or right.

## Danger

Your Player LOSES points when they hit The Danger.

The Danger is a _____
The Danger moves only to the left or right.

## Artwork/Sketches/Proof of Concept

Below is a **640x480 rectangle**, representing your game screen.

- Label the bottom-left corner (0,0).
- Label the other three corners with their corresponding coordinates.
- In the rectangle, sketch a picture of your game!

# Order of Operations

If you were to write instructions for getting ready for school, it would matter very much which instruction came first!

Imagine what might happen if someone tried to follow these steps:

1. Put on your sneakers.
2. Tie your sneakers.
3. Put on your socks.

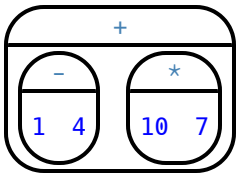
Sometimes we need multiple expressions in mathematics, and the order matters there, too!
Mathematicians didn't always agree on the **Order of Operations**, but at some point it became important to establish conventions that would allow them to work together.

To help us organize our math thinking into something we can trust, we can *diagram* an expression using the **Circles of Evaluation**.

For example, this expression:

$1 - 4 + 10 \times 7$

can be diagrammed as:




**Order of Operations** is important when programming, too!

To convert a **Circle of Evaluation** into Code, we walk through the circle from outside-in, moving left-to-right.

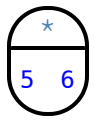
1. Type an open parenthesis when we *start* a circle.
2. Once we're in a circle, we first write the **function** at the top, then write the inputs from left to right.
3. Type a close parenthesis when we *end* a circle.

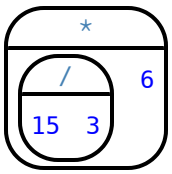
So, the Circle of Evaluation above would be programmed as:

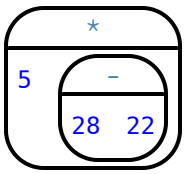
```
(+ (- 1 4) (* 10 7))
```

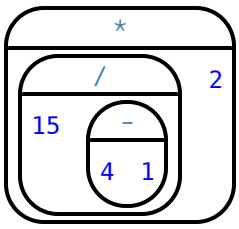
# Circles of Evaluation - Notice and Wonder

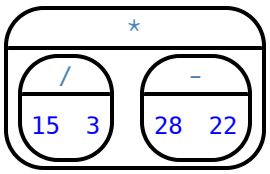
Let's take a look at a few ***Circles of Evaluation*** before we learn to draw them ourselves.



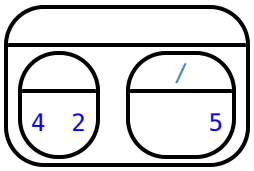
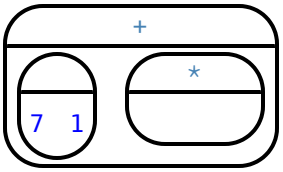
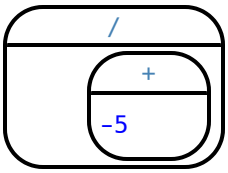
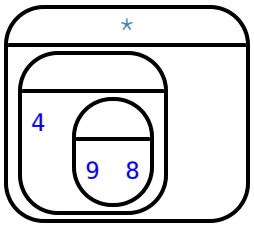
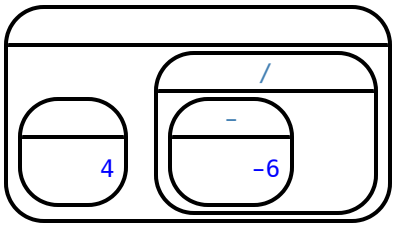
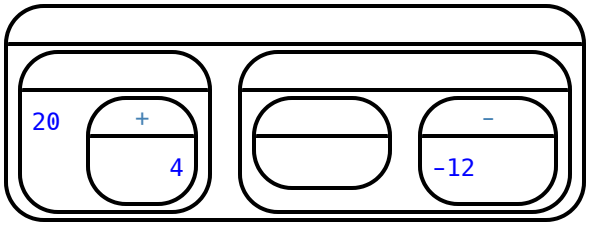




| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

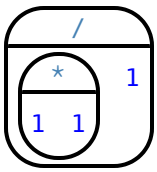
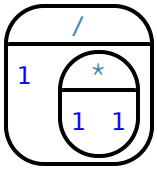
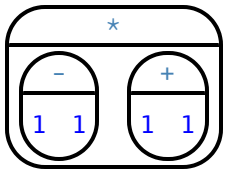
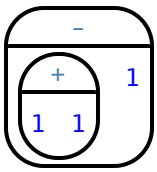
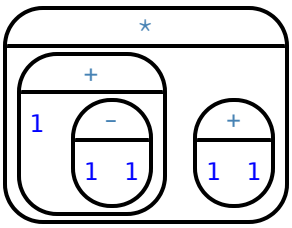
# Complete the Circles of Evaluation

For each expression on the left, finish the Circle of Evaluation on the right by filling in the blanks.

| | Arithmetic Expression | Circle of Evaluation |
|---|---|---|
| 1 | $4 + 2 - \frac{10}{5}$ |   |
| 2 | $7 - 1 + 5 \times 8$ |   |
| 3 | $\frac{-15}{-5 + 8}$ |   |
| 4 | $(4 + (9 - 8)) \times 5$ |   |
| 5 | $6 \times 4 + \frac{9 - -6}{5}$ |   |
| ★ | $\frac{20}{6 + 4} - \frac{5 \times 9}{-12 - 3}$ |   |

# Matching Expressions to Diagrams

Draw a line from each Circle of Evaluation on the left to the corresponding arithmetic expression on the right.

| Circle of Evaluation | | | Arithmetic Expression |
|---|---|---|---|
|  | 1 | A | $1 \div (1 \times 1)$ |
|  | 2 | B | $(1 + 1) - 1$ |
|  | 3 | C | $(1 \times 1) \div 1$ |
|  | 4 | D | $(1 + (1 - 1)) \times (1 + 1)$ |
|  | 5 | E | $(1 - 1) \times (1 + 1)$ |

# Expressions -> Circles of Evaluation

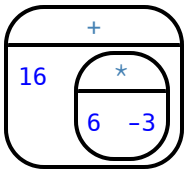
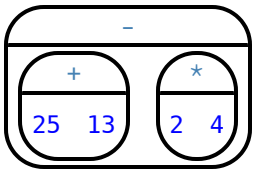
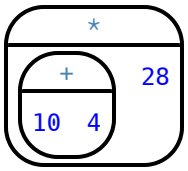
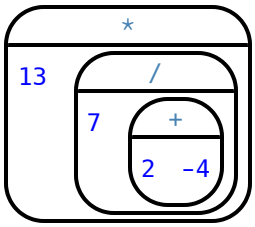
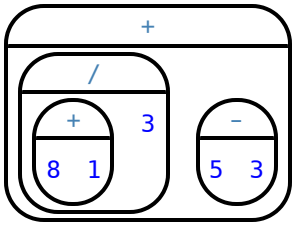
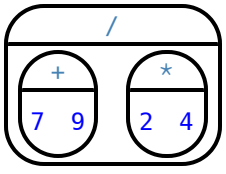
Translate each of the arithmetic expressions below into Circles of Evaluation.

| | Arithmetic Expression | Circle of Evaluation |
|---|---|---|
| 1 | $(6 \div 2) - (5 - 3)$ | |
| 2 | $9 - (2 \times 4)$ | |
| 3 | $8 - (1 + (2 \times 3))$ | |
| 4 | $(1 + (4 \times 7)) - 3$ | |

★ Rewrite each of these expressions with one less pair of parentheses without changing its Order of Operations.

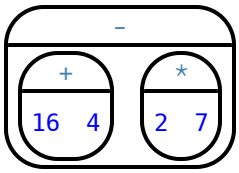
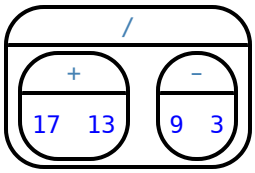
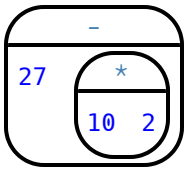
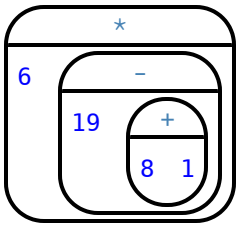
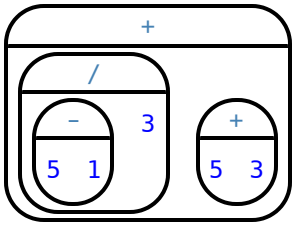
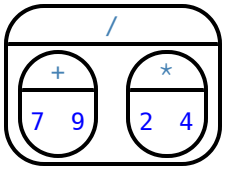
# Complete the Code!

For each Circle of Evaluation on the left, finish the Code on the right by filling in the blanks.

| | Circle of Evaluation | Code |
|---|---|---|
| 1 |  | `(+ _____ (* _____ -3))` |
| 2 |  | `(_____ (+ _____ 13) (_____ _____ 4))` |
| 3 |  | `(_____ (+ _____ 4) _____)` |
| 4 |  | `(_____ 13 (_____ 7 (_____ 2 -4)))` |
| 5 |  | `(_____ (_____ (_____ 8 1) 3) (_____ 5 3))` |
| 6 |  | `(/ (+ _____ _____) (* _____ _____))` |

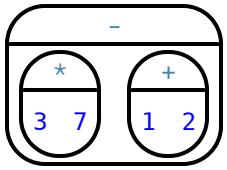
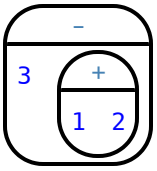
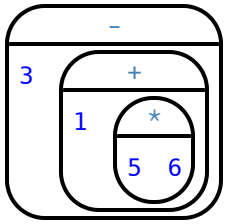
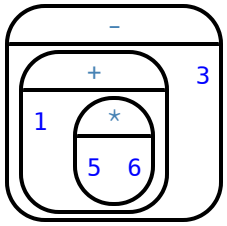
# Complete the Code by adding Parentheses!

For each Circle of Evaluation on the left, finish the Code on the right by adding parentheses.

| | Circle of Evaluation | Code |
|---|---|---|
| **1** |  | - + 16 4 * 2 7 |
| **2** |  | / + 17 13 - 9 3 |
| **3** |  | - 27 * 10 2 |
| **4** |  | 6 - 19 + 8 1 |
| **5** |  | + / - 5 1 3 + 5 3 |
| **6** |  | / + 7 9 * 2 4 |

# Expressions -> Circles of Evaluation -> Code 1

Complete the table by translating each of the arithmetic expressions below to code using the provided Circle of Evaluation.

| | Arithmetic Expression | Circle of Evaluation | Code |
|---|---|---|---|
| 1 | $3 \times 7 - (1 + 2)$ |   | |
| 2 | $3 - (1 + 2)$ |   | |
| 3 | $3 - (1 + 5 \times 6)$ |   | |
| 4 | $1 + 5 \times 6 - 3$ |   | |

# Expressions -> Circles of Evaluation -> Code 2

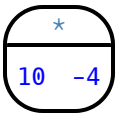
Translate each of the arithmetic expressions below into Circles of Evaluation, then translate them to Code.

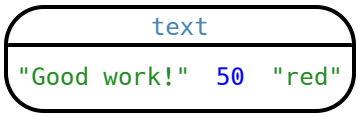
| | Arithmetic Expression | Circle of Evaluation | Code |
|---|---|---|---|
| **1** | $6 \times 8 + (7 - 23)$ | | |
| **2** | $18 \div 2 + 24 \times 4 - 2$ | | |
| **3** | $(22 - 7) \div (3 + 2)$ | | |
| **4** | $24 \div 4 \times 2 - 6 + 20 \times 2$ | | |

# Notice and Wonder - More than +, - , ÷, ×

## Part A

Here are two Circles of Evaluation . One of them is familiar, but the other is very different from what we've been working with.




1) **Focus on the Circles of Evaluation.** What do you Notice is different about the one on the right?

2) What do you Wonder about the Circle of Evaluation on the Right?

3) Can you figure out the Name for the function in the second Circle of Evaluation?

4) What do you think this expression will evaluate to? _____

## Part B

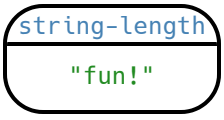
5) Convert this Circle of Evaluation to Code:

6) Test the code out in WeScheme!

7) What does the `50` mean to the computer? *Try replacing it with different values, and see what you get.*

8) What does the `"red"` mean to the computer? *Try replacing it with different values, and see what you get.*

**Here is another Circle of Evaluation to explore.**




9) Convert this Circle of Evaluation to code: _____

10) What do you think this expression will evaluate to? _____

# Expressions -> Circles of Evaluation -> Code - Challenge

Translate each of the arithmetic expressions below into Circles of Evaluation, then translate them to Code. *Hint: Two useful functions are* `sqr` *and* `sqrt`.

| | Arithmetic Expression | Circle of Evaluation | Code |
|---|---|---|---|
| 1 | $45 - 9 \times (3 + (2 - 4)) - 7$ | | |
| 2 | $50 \div 5 \times 2 - ((3 + 4) \times 2 - 5)$ | | |
| 3 | $\frac{16 + 3^2}{\sqrt{49} - 2}$ | | |

# 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.
- 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.) are treated the same way as functions: after all, they have inputs and outputs and obey the same rules!

## Applying Functions

Functions (and operators!) 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 math this could look like $f(5)$ or $g(10, 4)$.
- In WeScheme, these examples would be written as `(f 5)` and `(g 10 4)`.
- Applying the operator `+` to the inputs 1 and 2 would look like `(+ 1 2)`.
- Applying a function to make images would look like `(star 50 "solid" "red")`.
- There are many other functions in WeScheme, 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 WeScheme, 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? _____

_____

_____

# 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) `(string>? "a" "b")` | | |
| 3) `(= 3 2)` | | | 4) `(string<? "a" "b")` | | |
| 5) `(< 2 4)` | | | 6) `(string=? "a" "b")` | | |
| 7) `(>= 5 5)` | | | 8) `(string<>? "a" "a")` | | |
| 9) `(>= 4 6)` | | | 10) `(string>=? "a" "a")` | | |
| 11) `(<> 3 3)` | | | 12) `(string<>? "a" "b")` | | |
| 13) `(<> 4 3)` | | | 14) `(string>=? "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=? "a tree" "trees")` | | |
| 19) `(string=? "tree" "tree")` | | |
| 20) `(string-contains? "catnap" "cat")` | | |
| 21) `(string-contains? "cat" "catnap")` | | |

22) 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 WeScheme 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 WeScheme 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 WeScheme and click "Run". Experiment with each of the functions listed below, trying to find an expression that will build. Record the contract and example code for each function you are able to successfully build!*

| Name | | Domain | | Range |
|---|---|---|---|---|
| `; triangle` | `::` | `Number, String, Sting` | `->` | `Image` |
| `(triangle 80 "solid" "green")` | | | | |
| `; star` | `::` | | `->` | |
| | | | | |
| `; circle` | `::` | | `->` | |
| | | | | |
| `; rectangle` | `::` | | `->` | |
| | | | | |
| `; text` | `::` | | `->` | |
| | | | | |
| `; square` | `::` | | `->` | |
| | | | | |
| `; ellipse` | `::` | | `->` | |
| | | | | |
| `; regular-polygon` | `::` | | `->` | |
| | | | | |
| `; rhombus` | `::` | | `->` | |
| | | | | |
| `; 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 WeScheme 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`?

_____

## What Kind of Error is it?

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

4) In your own words, 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 WeScheme.*

5) `(triangle 20 "solid")`

```
triangle: expects 3 arguments, but given 2: 20 solid at: line 1, column 0, in <interactions>
```

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

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

```
triangle: expects a non-negative number as 1st argument, but given: solid; other arguments were:
red 20 at: line 1, column 0, in <interactions>
```

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

7) `(triangle 20 40 "solid" "red")`

```
triangle: expects 3 arguments, but given 4: 20 40 solid red at: line 1, column 0, in
<interactions>
```

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

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

```
solid: this variable is not defined at: line 1, column 0, in <interactions>
```

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

★ `(triangle 20 "striped" "red")`

```
triangle: expects a style ("solid" / "outline") or an opacity value [0-255]) as 2nd argument,
but given: "striped"; other arguments were: 20 "red" at: line 1, column 0, in <interactions>
```

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

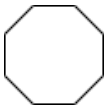
# Using Contracts

*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 [WeScheme](WeScheme) before recording it.*

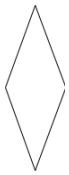
```
; ellipse :: ( Number , Number , String , String ) -> Image
              width    height   fill-style  color
```

| | | |
|---|---|---|
| 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 WeScheme to test your code.

1) What kind of triangle does the `triangle` function produce? _____
There are lots of other kinds of triangles! And WeScheme 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.

_____

_____

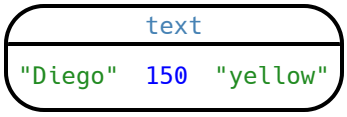
6) Which do you like better? Why? _____

# Composing with Circles of Evaluation

## Notice and Wonder

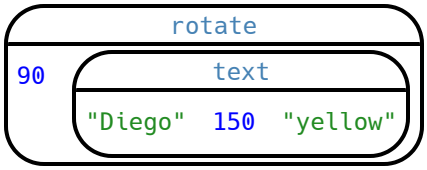
*Suppose we want to see the* `text` *"Diego" written vertically in yellow letters of size 150. Let's use Circles of Evaluation to look at the structure:*

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…*

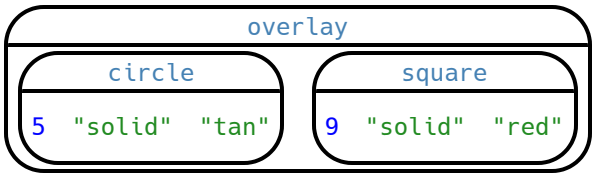
| **Write your name (any size), in your favorite color** | **`rotate` the image so that it's diagonal** |
|---|---|
| 3) Draw the circle of evaluation: | 4) Draw the circle of evaluation: |
| 5) Convert the Circle of Evaluation to code: | 6) Convert the Circle of Evaluation to code: |

# Circle of Evaluation to Code (Scaffolded)

## Complete the Code by Filling in the Blanks!

Finish the Code by filling in the blanks.

1)

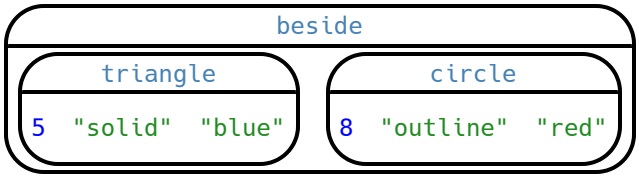



```
(overlay (circle ____ "solid" _____________________) (_____________________ 9 _____________________ "red"))
```

## Complete the Code by adding Parentheses

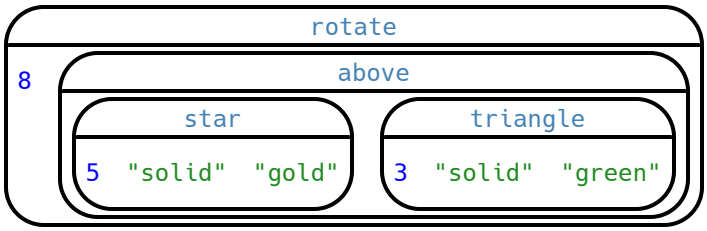
For each Circle of Evaluation, finish the Code by adding parentheses.

2)



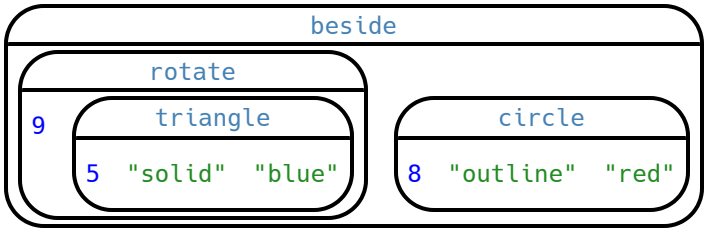

```
beside    triangle    5    "solid"    "blue"    circle    8    "outline" "red"
```

3)




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

4)




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

# Function Composition — Green Star

1) Draw a Circle of Evaluation and write the Code for a **solid, green star, size 50**. *Then go to WeScheme 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<br>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 WeScheme and type them into the Interactions Area and see if you can figure out how the code constructs its image.*

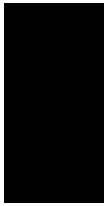
```
(beside (rectangle 200 100 "solid" "black")(square 100 "solid" "black"))

(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 |  | |
| ★ |  | |

# 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: $1 \times 3$ $\sqrt{16}$ $5$ - $2$
  - 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 WeScheme, these definitions are written a little differently, making it clear that we're talking about definitions:

- Try typing these definitions into the Definitions Area on the left, clicking "Run", and then *using* them in the Interactions Area on the right.
  - `(define x 4)`
  - `(define 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.
  - `(define x (+ 5 1))`
  - `(define y (* x 7))`
  - `(define food "Pizza!")`
  - `(define 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 WeScheme 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` | | |

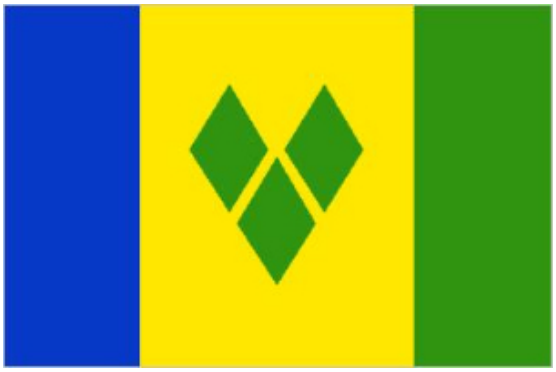
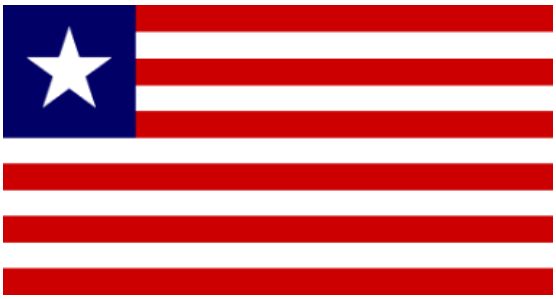
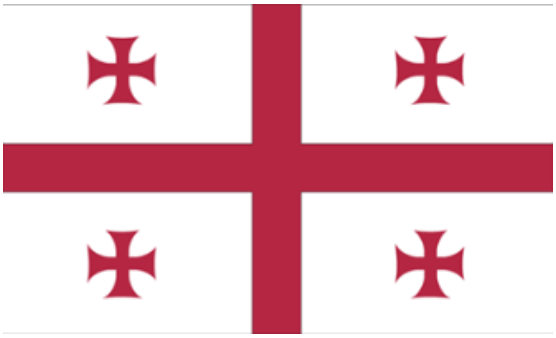
13) In the code, find the definitions of `exampleA`, `exampleB`, and `exampleC`. These all define the same shape, but their definitions are split across several lines. Suppose you *had* to split your code across multiple lines like this. Which one of these is the easiest to read, and why?

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?

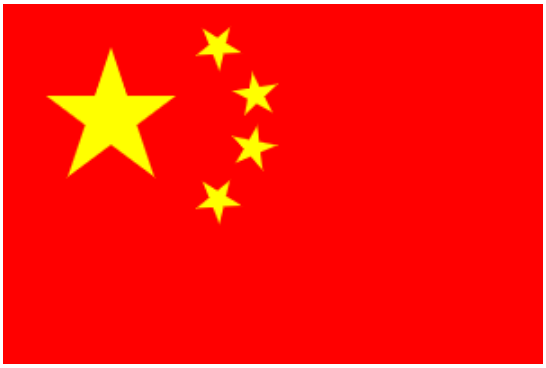
# Which Value(s) Would it Make Sense to Define?

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.



```
(define 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.

```
(define 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? _____.

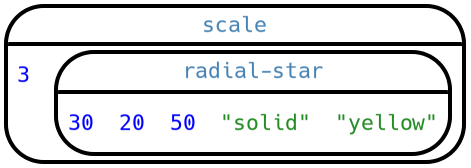
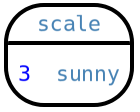
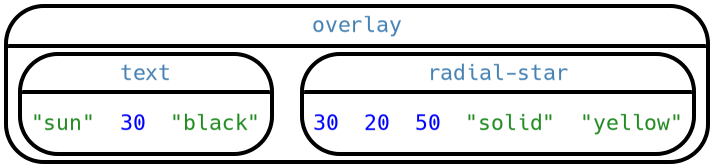
How many inputs are in its domain? _____. What are the types of those inputs? _____

# 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.

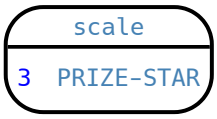
# Writing Code using Defined Values

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 WeScheme 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?

---

---

---

# Surface Area of a Rectangular Prism - Explore

1) What do you picture in your mind when you hear *rectangular prism*?

2) What do you picture in your mind when you hear *surface area*?

**Open the Surface Area of a Rectangular Prism Starter File and click "Run".**
**Type `prism` into the Interactions Area (on the right) and hit "enter" to see an image of a rectangular prism.**

3) How many faces does this prism have? _____

## Defining Faces

**Find PART 1 in the Definitions Area of the starter file (on the left). You will see a definition for `front` and `back`.**

4) How did the author know to use width and height as the dimensions for `front`? _____

5) Why are `front` and `back` defined to be the same thing? _____

6) Using these definitions as a model, add definitions for the other faces of this prism to the Definitions Area (on the left).

## Completing the List

**Find PART 2 in the starter file. You'll see `(list front back)` ... so far the list only includes `front` and `back`.**

7) Complete the faces list, then type `(print-imgs faces)` into the Interactions Area. What do you see?

## Printing Your Paper Model

**We're going to print the faces following directions in PART 3 and build a paper model of a rectangular prism.**
*Before you print and build your prism, you can change the length, width, and height of your prism at the top of the starter file. Be sure that all 3 dimensions are different, and that they are all small enough to fit on a sheet of paper. If you change them, record your new dimensions here.*

LENGTH: _____ WIDTH: _____ HEIGHT: _____

10) Calculate the surface area of your prism, by adding the area of each face. _____ Show your work below.

## Code for Calculating the Surface Area of a Prism

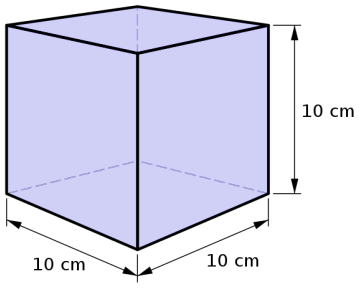
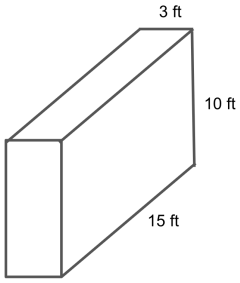
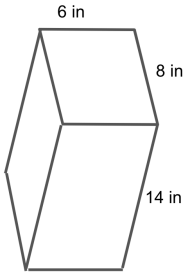
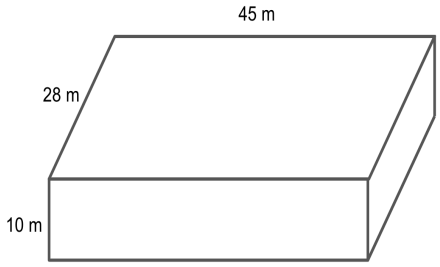
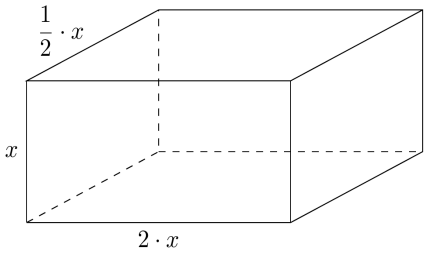
**Follow the directions in PART 4 of the starter file to write code to calculate the surface area.**

11) How many definitions did you write? _____

12) How does the surface area that the computer returns compare to the surface area you calculated by hand?

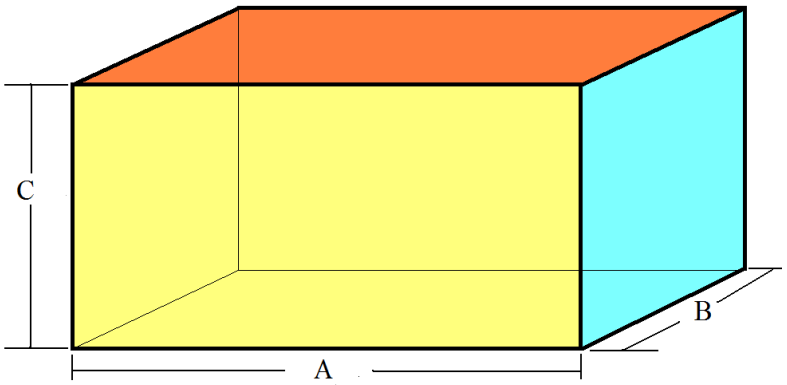
# Surface Area of a Prism - Practice

Find the Surface Area of each rectangular prism below. Show your work in the right-hand column, and write your final answer in the blank.

| | | |
|---|---|---|
| 1 |  | Surface Area: _____ |
| 2 |  | Surface Area: _____ |
| 3 |  | Surface Area: _____ |
| 4 |  | Surface Area: _____ |
| 5 |  | Surface Area: _____ |

# Surface Area of a Prism - More than One Way

Students in Mr. Grattan's class were asked to write code that would calculate the surface area of this rectangular prism. Help them convert their strategies into algebraic expressions and code, and double check that each strategy works.




1) Della says, "Just find the area of the top, bottom, left, right, front and back and add them all together!" **Will it work?** _____

- Algebraic Expression: $AB + AB + BC + BC + AC + AC = 2AB + 2BC + 2AC$
- Code: _____

2) Orion says, "Just find the area of the front, top and right faces, add them together, and double the sum." **Will it work?** _____

- Algebraic Expression: _____
- Code: _____

3) Jules says, "Double the area of the front, double the area of the top, double the area of the side. Then add them up." **Will it work?** _____

- Algebraic Expression: _____
- Code: _____

4) Tate says, "Just multiply the length times the width times the height and double their product." **Will it work?** _____

- Algebraic Expression: _____
- Code: _____

5) Can you think of one other way to find the surface area of the prism?

- Description: _____
- Algebraic Expression: _____
- Code: _____

6) Whose strategy do you like best? _____

Why? _____

# Making Sense of Coordinates

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

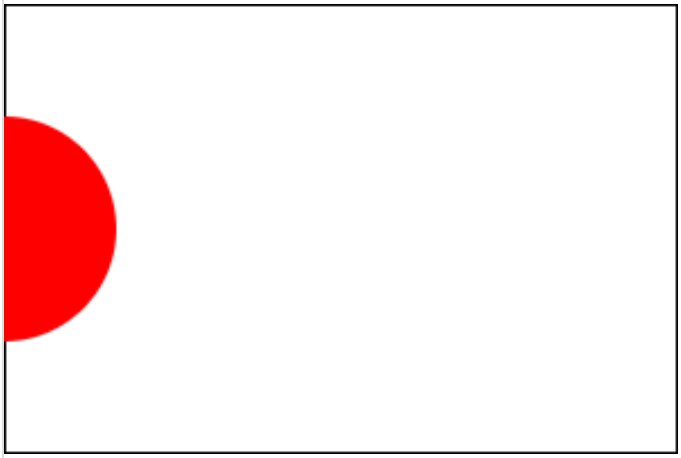
Think of the background image as a sheet of graph paper with the origin (0,0) in the bottom left corner. The width of the rectangle is 300 and the height is 200. The numbers in `translate` specify a point on that graph paper, where the center of the top image (in this case `dot`) should be placed.

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

1)

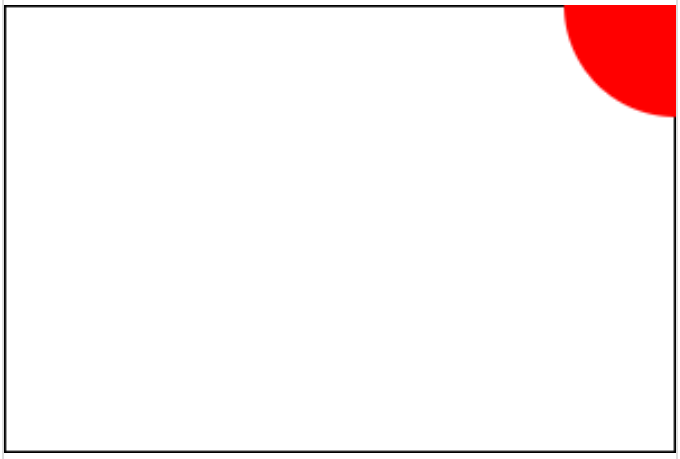
(translate dot _____ _____ background)

2)



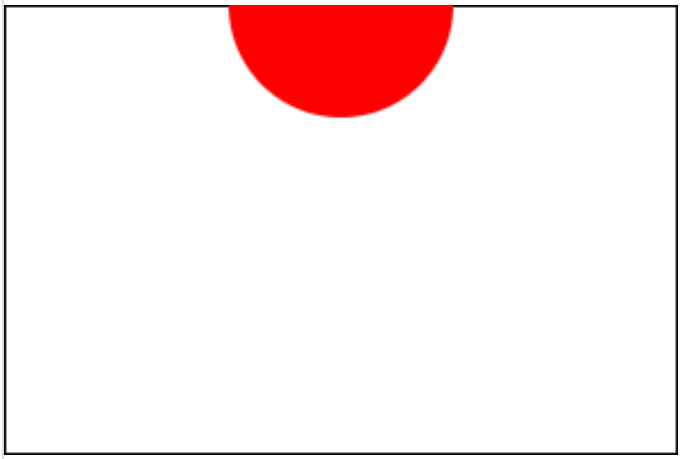
(translate dot _____ _____ background)

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 WeScheme, we use the semicolon ( `;` ). What color are comments in WeScheme? _____

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?

_____

_____

# Notice and Wonder

As you investigate the Blank Game Starter File with your partner, record what you Notice, and then what you Wonder.
*Remember, "Notices" are statements, not questions.*

| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

# 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) | $4 + 2$ |

```
(EXAMPLE (f 1) (+ 1 2))
(EXAMPLE (f 2) (+ 2 2))
(EXAMPLE (f 3) (+ 3 2))
(EXAMPLE (f 4) (+ 4 2))
```

## 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:

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

```
(define (f x) (+ x 2))
```

## 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:

`(define (rs _____)` _____`)`

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:

`(define (bigc _____)` _____`)`

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:

`(define (ps _____)` _____`)`

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):

`(define (sun` _____ _____`)` _____`)`

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):

`(define (me` _____ _____`)` _____`)`

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):

`(define (gr` _____ _____`)` _____`)`

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.

```
(define (_____ _____) _____)
```

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.

```
(define (_____ _____) _____)
```

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.

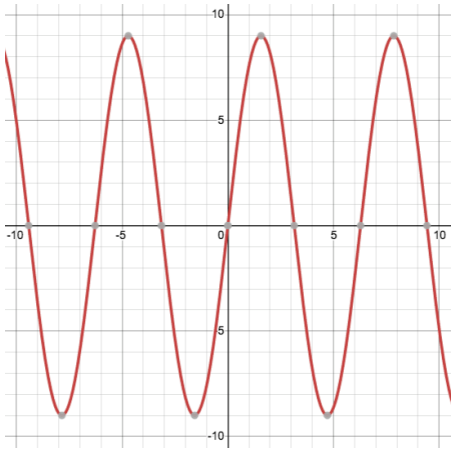
```
(define (_____ _____) _____)
```

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

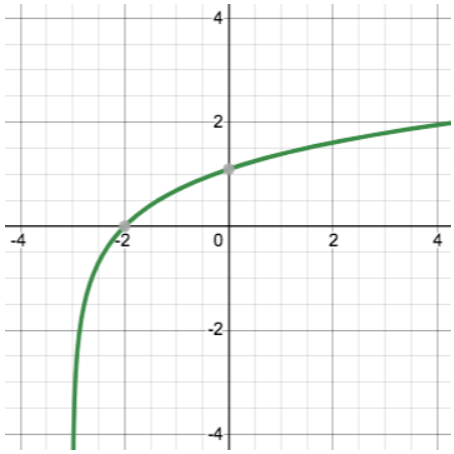
# Identifying Functions from Graphs

Decide whether each graph below is a function. If it's not, prove it by drawing a vertical line that crosses the graph at more than one point.

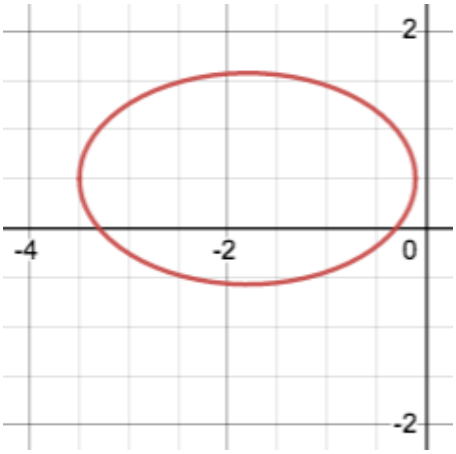
1) **Function** or **Not a Function**?



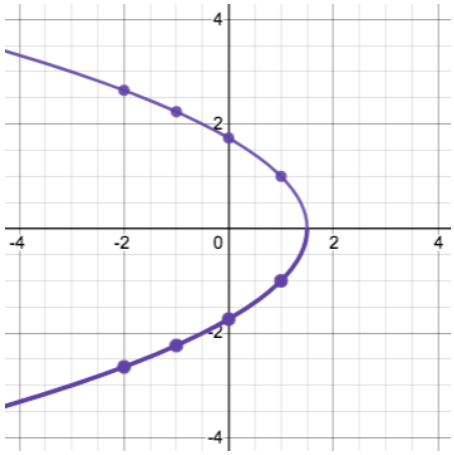

2) **Function** or **Not a Function**?



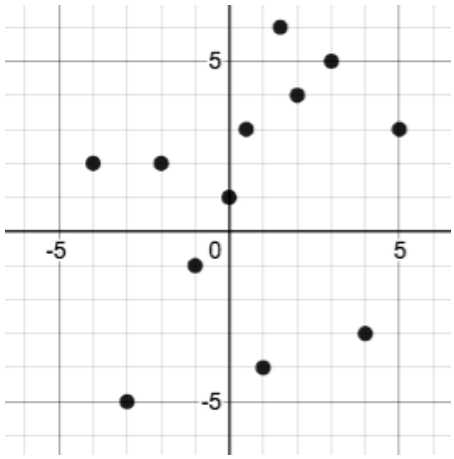

3) **Function** or **Not a Function**?



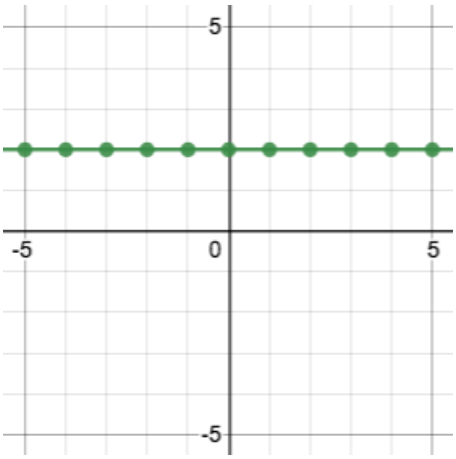

4) **Function** or **Not a Function**?



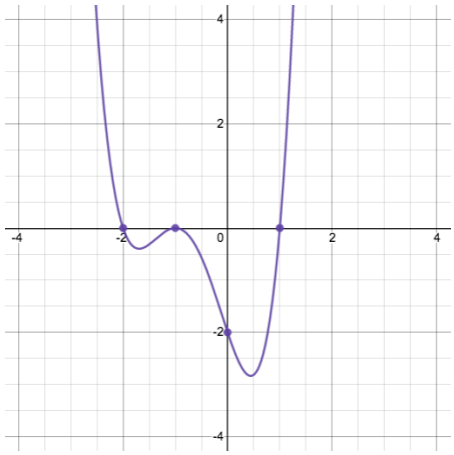

5) **Function** or **Not a Function**?



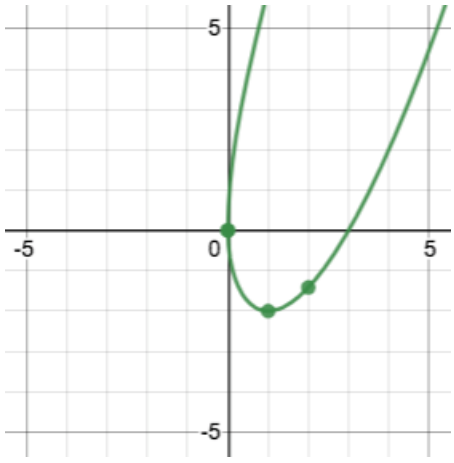

6) **Function** or **Not a Function**?



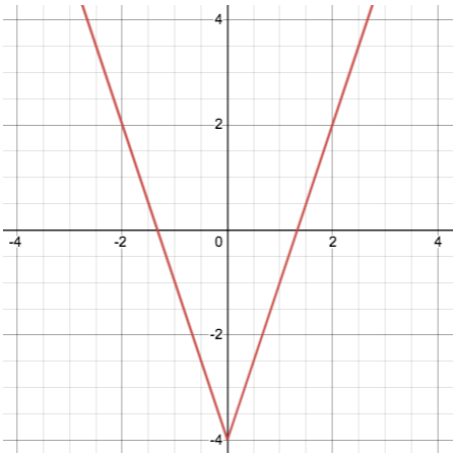

7) **Function** or **Not a Function**?




8) **Function** or **Not a Function**?



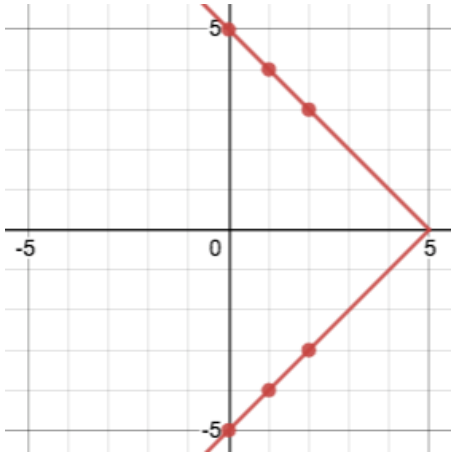

9) **Function** or **Not a Function**?



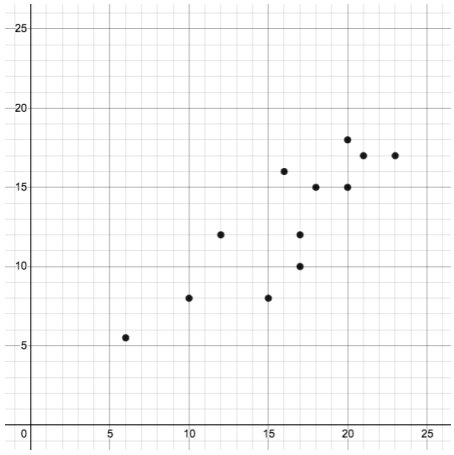
# Identifying Functions from Graphs (2)

Decide whether each graph below is a function. If it's not, prove it by drawing a vertical line that crosses the graph at more than one point.

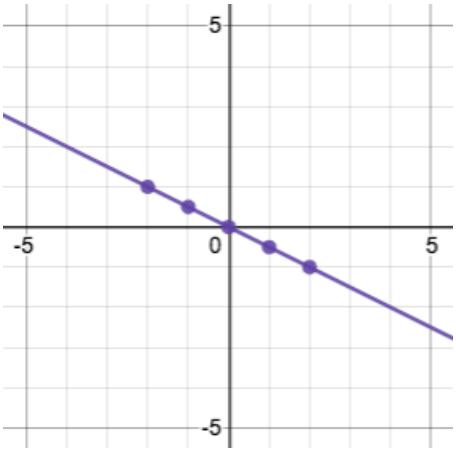
1) **Function** or **Not a Function**?



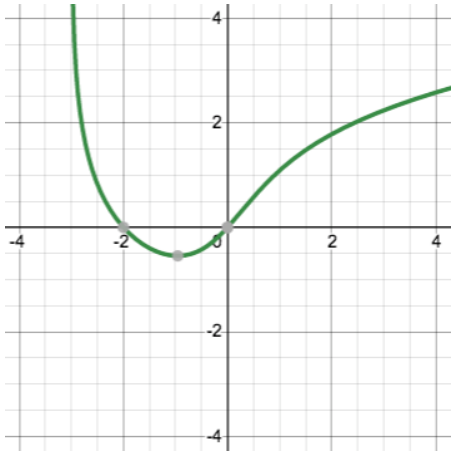

2) **Function** or **Not a Function**?



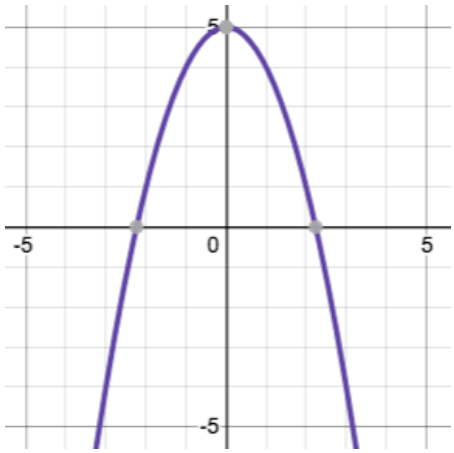

3) **Function** or **Not a Function**?



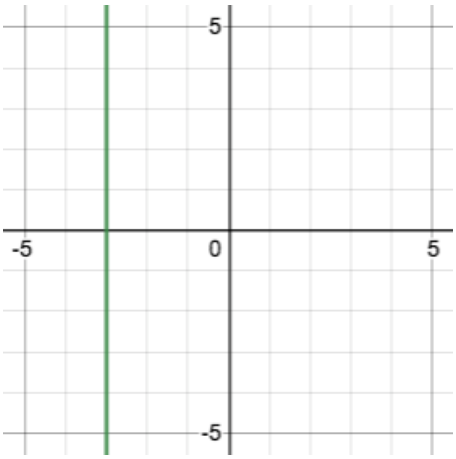

4) **Function** or **Not a Function**?



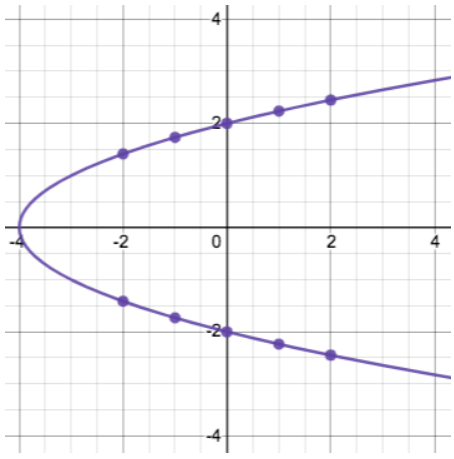

5) **Function** or **Not a Function**?



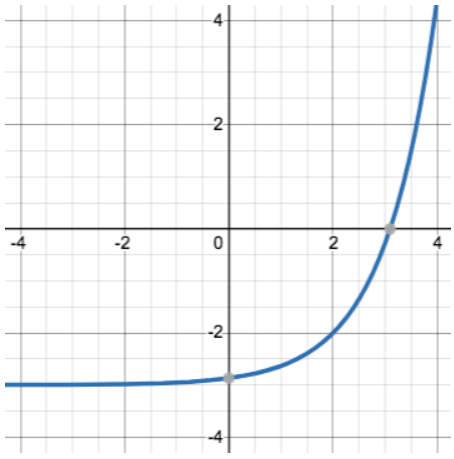

6) **Function** or **Not a Function**?



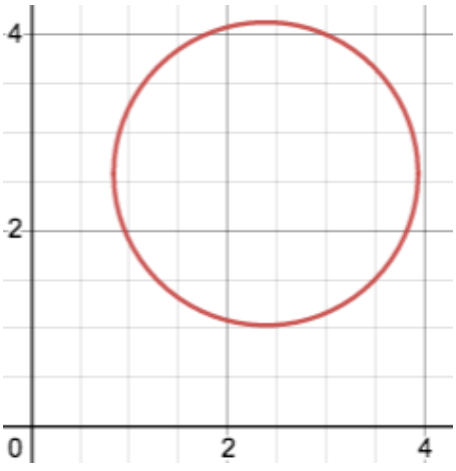

7) **Function** or **Not a Function**?




8) **Function** or **Not a Function**?




9) **Function** or **Not a Function**?



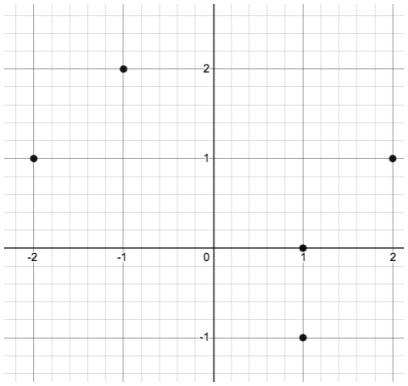
# Notice and Wonder - Functions

Write down what you Notice and Wonder about the graphs you've just seen. At a later point you will *also* use this page to record what you Notice and Wonder about the tables you'll see. *Remember: "Notices" should be statements, not questions!*

| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

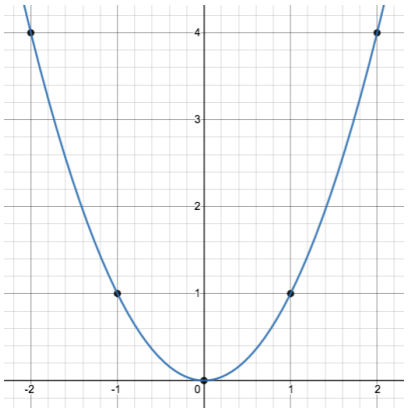
# How Tables Fail the Vertical Line Test

1) Each of the graphs below is also represented by a table. Use the vertical line test to determine whether or not each graph represents a function.



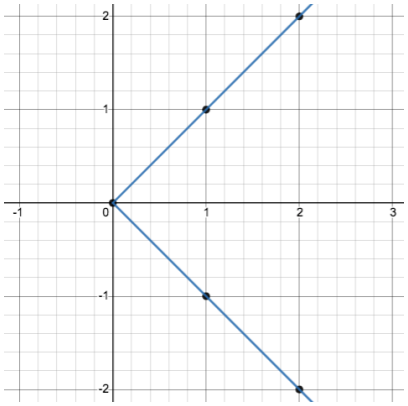
**Function** or **Not a Function**?

| x | -2 | -1 | 1 | 1 | 2 |
|---|---|---|---|---|---|
| y | 1 | 2 | 0 | -1 | 1 |



**Function** or **Not a Function**?

| x | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| y | 4 | 1 | 0 | 1 | 4 |



**Function** or **Not a Function**?

| x | 2 | 1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| y | 2 | 1 | 0 | -1 | -2 |

2) For each graph that failed the vertical line test, label the offending points with their coordinates.

3) Find the same coordinates in the table below the graph and circle or highlight them.

4) What do the tables of the non-functions have in common? What could you look for in other tables to identify whether or not they could represent a function?

5) Use the process you just described to determine whether each table below could represent a function. Circle or highlight the points that would end up on the same vertical line.

| x | y |
|---|---|
| 0 | -2 |
| 1 | -2 |
| 2 | -2 |
| 3 | -2 |
| 4 | -2 |

**Function** or **Not**?

| x | y |
|---|---|
| 0 | -2 |
| 1 | 1 |
| 2 | 4 |
| 3 | 7 |
| 3 | 10 |

**Function** or **Not**?

| x | y |
|---|---|
| 0 | 3 |
| 1 | 4 |
| -1 | 5 |
| 2 | 6 |
| -2 | 7 |

**Function** or **Not**?

| x | y |
|---|---|
| 1 | 0 |
| 0 | 1 |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |

**Function** or **Not**?

# Identifying Functions from Tables

Decide whether or not each table below could represent a function. If not, circle what you see that tells you it's not a function.
*In a function, there is exactly one y-value (or output) for each x-value (or input). If a table has more than one y-value (or output) for the same x-value (or input), it can't represent a function.*

1) **Function** or **Not**?

| x | y |
|---|---|
| 0 | 3 |
| 1 | 2 |
| 2 | 5 |
| 3 | 6 |
| 4 | 5 |

2) **Function** or **Not**?

| ind | dep |
|---|---|
| 5 | 3 |
| 1 | 4 |
| -3 | 5 |
| 3 | 6 |
| 2 | 7 |

3) **Function** or **Not**?

| input | output |
|---|---|
| 0 | 2 |
| 5 | 2 |
| 2 | 2 |
| 6 | 2 |
| 3 | 2 |

4) **Function** or **Not**?

| x | y |
|---|---|
| 1 | 0 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |

5) **Function** or **Not**?

| tickets | $ |
|---|---|
| 2 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |

6) **Function** or **Not**?

| input | output |
|---|---|
| -4 | -2 |
| -3 | -1 |
| -2 | 0 |
| -1 | 1 |
| 0 | 2 |

7) **Function** or **Not**?

| ind | dep |
|---|---|
| 10 | 9 |
| 3 | 2 |
| 9 | 8 |
| 17 | 16 |
| 3 | 5 |

8) **Function** or **Not**?

| C | F |
|---|---|
| -40 | -40 |
| 0 | 32 |
| 10 | 50 |
| 37 | 98.6 |
| 100 | 212 |

9) **Function** or **Not**?

| input | output |
|---|---|
| 0 | 7 |
| -1 | 2 |
| 4 | 3 |
| 8 | 6 |
| -5 | -8 |

10) **Function** or **Not**?

| $ | games |
|---|---|
| 10 | 5 |
| 11 | 25 |
| 12 | 45 |
| 13 | 65 |
| 14 | 85 |

11) **Function** or **Not**?

| x | y |
|---|---|
| 8 | 10 |
| 6 | 5 |
| 4 | 0 |
| 6 | -5 |
| 8 | -10 |

12) **Function** or **Not**?

| miles | minutes |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |

# Identifying Functions from Tables & Graphs

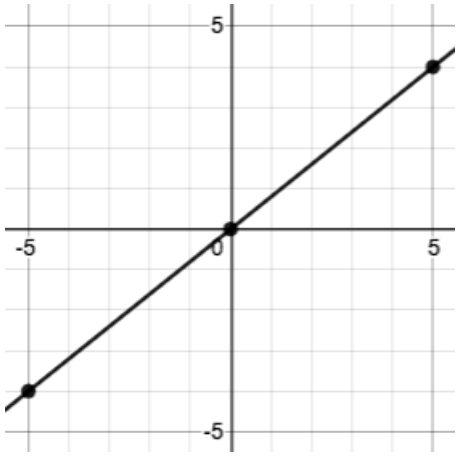
Decide whether or not each table or graph below could represent a function. If not, circle what tells you it's not a function.

*In a function, there's exactly one y-value for each x-value. Any table or graph with more than one y-value for the same x-value, can't represent a function.*

1) **Function** or **Not a Function**?

| x | y |
|---|---|
| -2 | 5 |
| 0 | 2 |
| 2 | 4 |
| 4 | 7 |
| 6 | 8 |

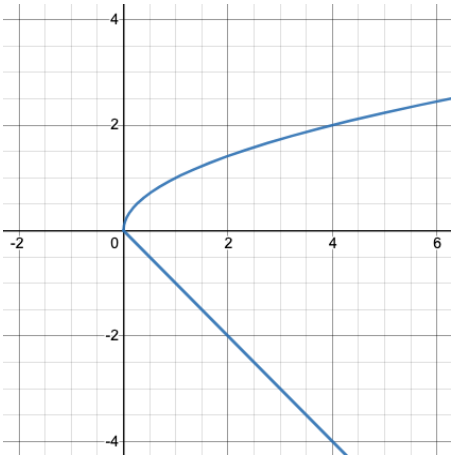
2) **Function** or **Not a Function**?




3) **Function** or **Not a Function**?

| x | y |
|---|---|
| 0 | 7 |
| 1 | 2 |
| 1 | 3 |
| 2 | 6 |
| 3 | -8 |

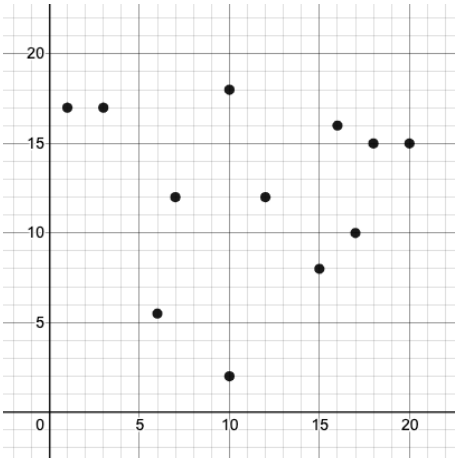
4) **Function** or **Not a Function**?




5) **Function** or **Not a Function**?

| x | y |
|---|---|
| -1.5 | -2 |
| -1 | -1 |
| -0.5 | 0 |
| 0 | 1 |
| 0.5 | 2 |

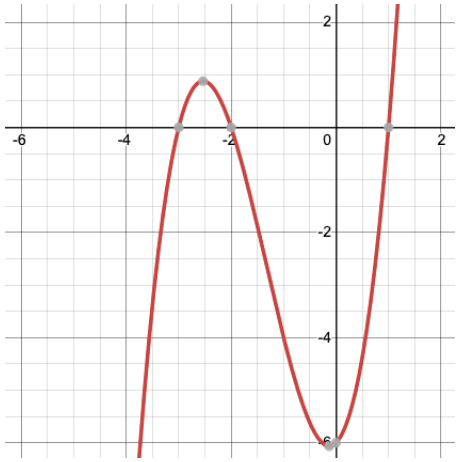
6) **Function** or **Not a Function**?




7) **Function** or **Not a Function**?

| x | y |
|---|---|
| -1 | 1.5 |
| 0 | 1.5 |
| 1 | 1.5 |
| 2 | 1.5 |
| 3 | 1.5 |

8) **Function** or **Not a Function**?




9) **Function** or **Not a Function**?

| x | y |
|---|---|
| 8 | 1 |
| 5 | 2 |
| 4 | 3 |
| 5 | 4 |
| 8 | 5 |

# Matching Examples and Definitions (Math)

Match each of the function definitions on the left with the corresponding table on the right.
*It may help to circle or highlight what's changing in the $f(x)$ column of the table!*

| Function Definitions | | | Example Tables |
|---|---|---|---|
| $f(x) = x - 2$ | **1** | **A** | |
| $f(x) = 2x$ | **2** | **B** | |
| $f(x) = 2x + 1$ | **3** | **C** | |
| $f(x) = 1 - 2x$ | **4** | **D** | |
| $f(x) = 2 + x$ | **5** | **E** | |

| $x$ | $f(x)$ |
|---|---|
| 1 | $2 \times 1$ |
| 2 | $2 \times 2$ |
| 3 | $2 \times 3$ |

| $x$ | $f(x)$ |
|---|---|
| 15 | 15 - 2 |
| 25 | 25 - 2 |
| 35 | 35 - 2 |

| $x$ | $f(x)$ |
|---|---|
| 10 | 2 + 10 |
| 15 | 2 + 15 |
| 20 | 2 + 20 |

| $x$ | $f(x)$ |
|---|---|
| 0 | 1 - 2(0) |
| 1 | 1 - 2(1) |
| 2 | 1 - 2(2) |

| $x$ | $f(x)$ |
|---|---|
| 10 | 2(10) + 1 |
| 20 | 2(20) + 1 |
| 30 | 2(30) + 1 |

# Function Notation - Substitution

## Part 1

Complete each **row** of the table below, substituting the given value into the expression and evaluating.

| | Function Definition | Expression | Substitution | Evaluates to |
|---|---|---|---|---|
| 1) | $f(x) = x + 2$ | $f(3)$ | $3 + 2$ | 5 |
| 2) | $g(x) = x - 1$ | $g(6)$ | | |
| 3) | $h(x) = 3x$ | $h(4)$ | | |
| 4) | $k(x) = 2x - 1$ | $k(5)$ | | |

## Part 2

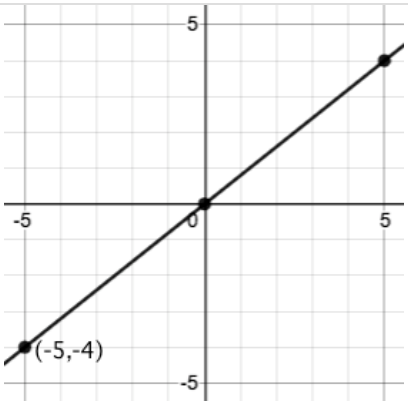
Each **column** below includes four different functions. Beneath each of them are a collection of different expressions for you to evaluate.

| **5)** $m(x) = -2x + 3$ | **6)** $n(x) = -x + 7$ | **7)** $v(x) = 10x - 8$ | **8)** $w(x) = x^2$ |
|---|---|---|---|
| $m(3) = -2(3) + 3$ | $n(5) =$ | $v(7) =$ | $w(-2) =$ |
| $-3$ | | | |
| $m(-4) =$ | $n(-2) =$ | $v(0) =$ | $w(10) =$ |
| | | | |
| $m(0) =$ | $n(3.5) =$ | $v(-10) =$ | $w(0) =$ |
| | | | |
| $m(0.5) =$ | $n(0) =$ | $v(2.5) =$ | $w(1.5) =$ |
| | | | |

| What do you Notice? | What do you Wonder? |
|---|---|
| | |

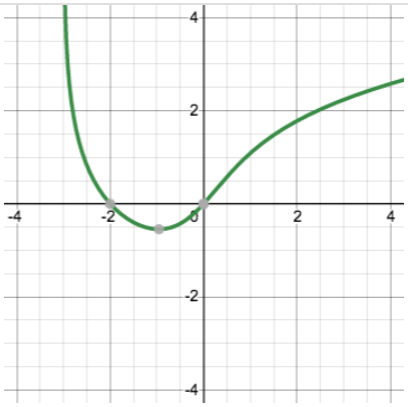
# Function Notation - Graphs

For each graph, find the values described by the expressions below. *The first one has been done for you.*



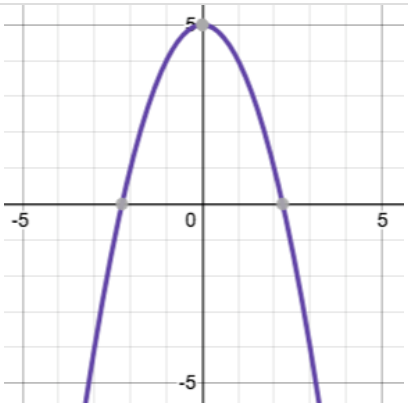

1) $f(-5) =$ -4

2) $f(5) =$ _____



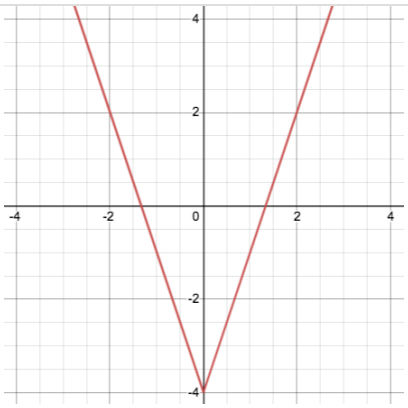

3) $g(-2) =$ _____

4) $g(0) =$ _____



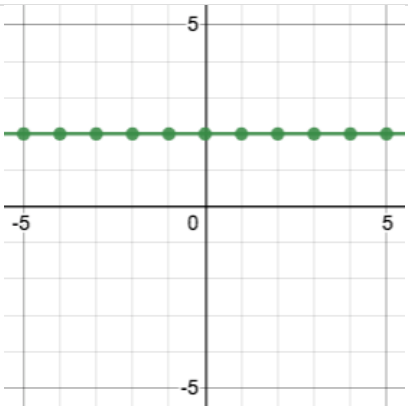

5) $h(0) =$ _____

6) $h(1) =$ _____



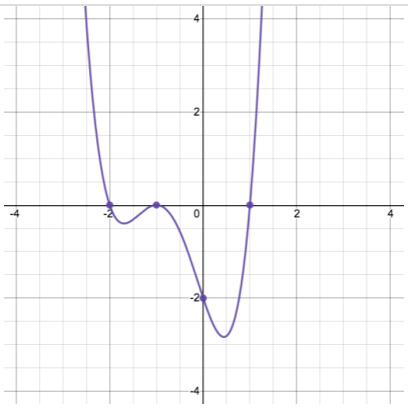

7) $j(-2) =$ _____

8) $j(0) =$ _____



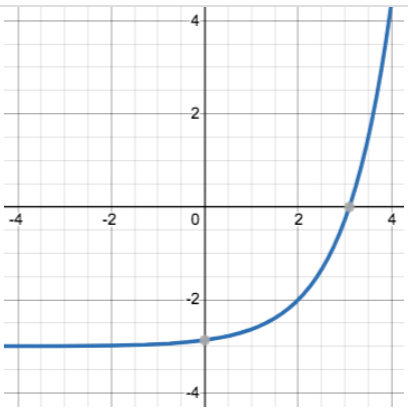

9) $k(3) =$ _____

10) $k(-2.5) =$ _____



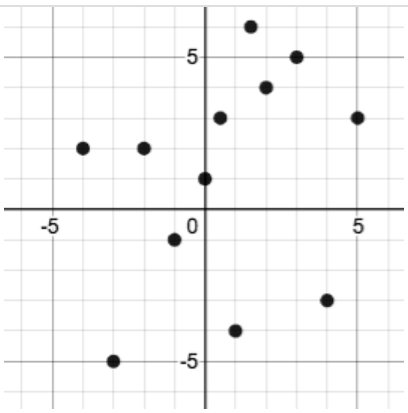

11) $m(0) =$ _____

12) $m(1) =$ _____



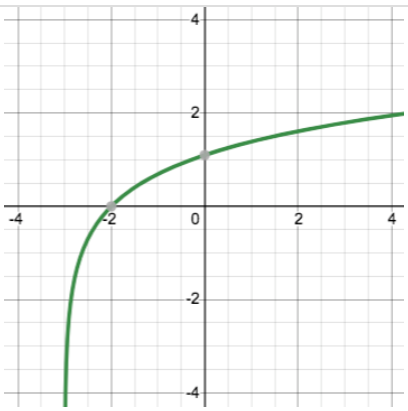

13) $n(2) =$ _____

14) $n(-\infty) \approx$ _____




15) $v(5) =$ _____

16) $v(2) =$ _____




17) $w(-2) =$ _____

18) $w(0) =$ _____

# Function Notation - Tables

Find the values described by the expressions below each table.

*Note: Not all of the relationships here are actually functions, which means that not all of these expressions can be evaluated!*

| $x$ | $f(x)$ |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |

| $x$ | $g(x)$ |
|---|---|
| 5 | 3 |
| 1 | 4 |
| -3 | 5 |
| 3 | 6 |
| 2 | 7 |

| $x$ | $h(x)$ |
|---|---|
| 0 | 2 |
| 5 | 2 |
| 2 | 2 |
| 6 | 2 |
| 3 | 2 |

| $x$ | $y(x)$ |
|---|---|
| 1 | 0 |
| 1 | 1 |
| 1 | 2 |
| 1 | 3 |
| 1 | 4 |

1) $f(3) =$ 6

2) $f(4) =$ _____

3) $g(1) =$ _____

4) $g(3) =$ _____

5) $h(0) =$ _____

6) $h(3) =$ _____

7) $y(1) =$ _____

8) $y(8) =$ _____

| $a$ | $b(a)$ |
|---|---|
| -4 | -2 |
| -3 | -1 |
| -2 | 0 |
| -1 | 1 |
| 0 | 2 |

| $c$ | $d(c)$ |
|---|---|
| 0 | 3 |
| 1 | 2 |
| 2 | 5 |
| 3 | 6 |
| 4 | 5 |

| $n$ | $m(n)$ |
|---|---|
| 0 | 0 |
| -1 | -1 |
| -2 | -2 |
| -3 | -3 |
| -4 | -4 |

| $q$ | $p(q)$ |
|---|---|
| 2 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |

9) $b(-1) =$ _____

10) $b(0) =$ _____

11) $d(2) =$ _____

12) $d(4) =$ _____

13) $m(0) =$ _____

14) $m(-3) =$ _____

15) $p(1) =$ _____

16) $p(2) =$ _____

| $s$ | $r(s)$ |
|---|---|
| 0 | 7 |
| -1 | 2 |
| 4 | 3 |
| 8 | 6 |
| -5 | -8 |

| $w$ | $v(w)$ |
|---|---|
| 10 | 5 |
| 11 | 25 |
| 12 | 45 |
| 13 | 65 |
| 14 | 85 |

| $y$ | $z(y)$ |
|---|---|
| 8 | 10 |
| 6 | 5 |
| 4 | 0 |
| 5 | -5 |
| 8 | -10 |

| $time$ | $l(time)$ |
|---|---|
| 10 | 9 |
| 3 | 2 |
| 9 | 8 |
| 17 | 16 |
| 5 | 5 |

17) $r(-1) =$ _____

18) $r(8) =$ _____

19) $v(11) =$ _____

20) $v(14) =$ _____

21) $z(6) =$ _____

22) $z(2) =$ _____

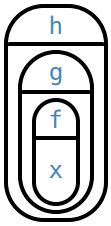
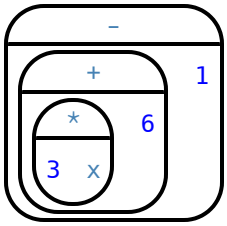
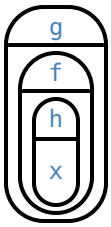
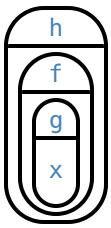
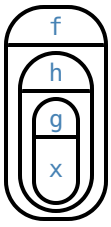
23) $l(10) =$ _____

24) $l(3) =$ _____

# Diagramming Function Composition

| | | |
|---|---|---|
| `f :: Number -> Number`<br>`Consumes a number,`<br>`multiplies by 3 to produce the result` | `g :: Number -> Number`<br>`Consumes a number, adds six to produce the result` | `h :: Number -> Number`<br>`Consumes a number,`<br>`subtracts one to produce the result` |
| $f(x) = 3x$ | $g(x) = x + 6$ | $h(x) = x - 1$ |

For each function composition diagrammed below, translate it into the equivalent Circle of Evaluation for Order of Operations.
Then write expressions for *both* versions of the Circles of Evaluation, and evaluate them for $x = 4$. The first one has been completed for you.

| | Function Composition | Order of Operations | Translate & Evaluate | |
|---|---|---|---|---|
| 1 |  |  | Composition: | `(h (g (f x)))` |
| | | | Operations: | `(- (+ (* 3 x) 6) 1)` |
| | | | Evaluate for x = 4 | $h(g(f(4))) = ((3 \times 4) + 6) - 1 = 17$ |
| 2 |  | | Composition: | |
| | | | Operations: | |
| | | | Evaluate for x = 4 | |
| 3 |  | | Composition: | |
| | | | Operations: | |
| | | | Evaluate for x = 4 | |
| 4 |  | | Composition: | |
| | | | Operations: | |
| | | | Evaluate for x = 4 | |

# Matching Examples and Contracts

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

| Examples | | | Contract |
|---|---|---|---|
| `(EXAMPLE (f 5) (/ 5 2))`<br>`(EXAMPLE (f 9) (/ 9 2))`<br>`(EXAMPLE (f 24) (/ 24 2))` | 1 | A | `; f :: Number -> Number` |
| `(EXAMPLE (f 1)`<br>`  (rectangle 1 1 "outline" "red"))`<br>`(EXAMPLE (f 6)`<br>`  (rectangle 6 6 "outline" "red"))` | 2 | B | `; f :: String -> Image` |
| `(EXAMPLE (f "pink" 5)`<br>`  (star 5 "solid" "pink"))`<br>`(EXAMPLE (f "blue" 8)`<br>`  (star 8 "solid" "blue"))` | 3 | C | `; f :: Number -> Image` |
| `(EXAMPLE (f "Hi!") (text "Hi!" 50 "red"))`<br>`(EXAMPLE (f "Ciao!") (text "Ciao!" 50 "red"))` | 4 | D | `; f :: Number, String -> Image` |
| `(EXAMPLE (f 5 "outline")`<br>`  (star 5 "outline" "yellow"))`<br>`(EXAMPLE (f 5 "solid")`<br>`  (star 5 "solid" "yellow"))` | 5 | E | `; f :: String, Number -> Image` |

# Matching Examples and Function Definitions

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

```
(EXAMPLE (gt 20) (triangle 20 "solid" "green"))
(EXAMPLE (gt 50) (triangle 50 "solid" "green"))
(define (gt size) (gt size "solid" "green"))
```

(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 |
|---|---|---|
| `(EXAMPLE (f "solid")`<br>`  (circle 8 "solid" "red"))`<br>`(EXAMPLE (f "outline")`<br>`  (circle 8 "outline" "red"))` | 1 | A `(define (f s) (star s "outline" "red"))` |
| `(EXAMPLE (f 2) (+ 2 2))`<br>`(EXAMPLE (f 4) (+ 4 4))`<br>`(EXAMPLE (f 5) (+ 5 5))` | 2 | B `(define (f num) (+ num num))` |
| `(EXAMPLE (f "red") (circle 7 "solid" "red"))`<br>`(EXAMPLE (f "teal")`<br>`  (circle 7 "solid" "teal"))` | 3 | C `(define (f c) (star 9 "solid" c))` |
| `(EXAMPLE (f "red") (star 9 "solid" "red"))`<br>`(EXAMPLE (f "grey") (star 9 "solid" "grey"))`<br>`(EXAMPLE (f "pink") (star 9 "solid" "pink"))` | 4 | D `(define (f s) (circle 8 s "red"))` |
| `(EXAMPLE (f 3) (star 3 "outline" "red"))`<br>`(EXAMPLE (f 8) (star 8 "outline" "red"))` | 5 | E `(define (f c) (circle 7 "solid" c))` |

# 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
(EXAMPLE (big-triangle 100 "red")
  (triangle 100 "solid" "red"))
(EXAMPLE (big-triangle 200 "orange")
  (triangle 200 "solid" "orange"))
```

2)
```
(EXAMPLE (purple-square 15)
  (rectangle 15 15 "outline" "purple"))
(EXAMPLE (purple-square 6)
  (rectangle 6 6 "outline" "purple"))
```

3)
```
(EXAMPLE (sum 5 8) (+ 5 8))
(EXAMPLE (sum 9 6) (+ 9 6))
(EXAMPLE (sum 120 11) (+ 120 11))
```

4)
```
(EXAMPLE (banner "Game Today!")
  (text "Game Today!" 50 "red"))
(EXAMPLE (banner "Go Team!")
  (text "Go Team!" 50 "red"))
(EXAMPLE (banner "Exit")
  (text "Exit" 50 "red"))
```

5)
```
(EXAMPLE (twinkle "outline" "red")
  (star 5 "outline" "red"))
(EXAMPLE (twinkle "solid" "pink")
  (star 5 "solid" "pink"))
(EXAMPLE (twinkle "outline" "grey")
  (star 5 "outline" "grey"))
```

6)
```
(EXAMPLE (half 5) (/ 5 2))
(EXAMPLE (half 8) (/ 8 2))
(EXAMPLE (half 900) (/ 900 2))
```

7)
```
(EXAMPLE (Spanish 5) "cinco")
(EXAMPLE (Spanish 30) "treinta")
(EXAMPLE (Spanish 12) "doce")
```

# 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…

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

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

**Definition**

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

(define (`gt size`)
function name | variable(s)

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

---

*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…

*(EXAMPLE (* *)* )
function name | input(s) | what the function produces

*(EXAMPLE (* *)* )
function name | input(s) | what the function produces

**Definition**

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

(define ( )
function name | variable(s)

)
what the function does with those variable(s)

# 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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

---

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

# 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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

**Definition**

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

(define ( _____ )
function name | variable(s)

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

---

**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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

**Definition**

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

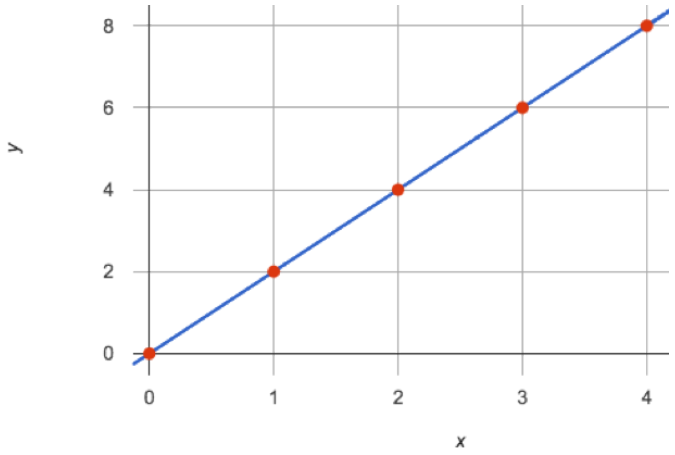
(define ( _____ )
function name | variable(s)

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

# Notice and Wonder (Linearity)

## Part 1

| x | y |
|---|---|
| 0 | 0 |
| 1 | 2 |
| 2 | 4 |
| 3 | 6 |
| 4 | 8 |




| What do you Notice? | What do you Wonder? |
|---|---|
| | |

## Part 2

- What is the y-value for each table when x is 0?
- What is the next pair for each of these tables?

| x | y |
|---|---|
| 0 | |
| 1 | 2 |
| 2 | 3 |
| 3 | 4 |
| 4 | 5 |
| 5 | 6 |
| | |

| independent | dependent |
|---|---|
| 0 | |
| 1 | 20 |
| 2 | 17 |
| 3 | 14 |
| 4 | 11 |
| 5 | 8 |
| | |

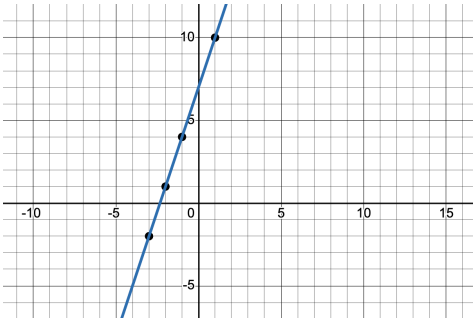
# Matching Tables to Graphs

For each of the tables below, find the graph that matches.
**Note:** The scales on the graphs are not the same! Look at the axes to help you find the right match!

| x | -1 | 0 | 1 | 2 | 2 |
|---|---|---|---|---|---|
| y | 4 | 7 | 10 | 13 | 16 |

**1**

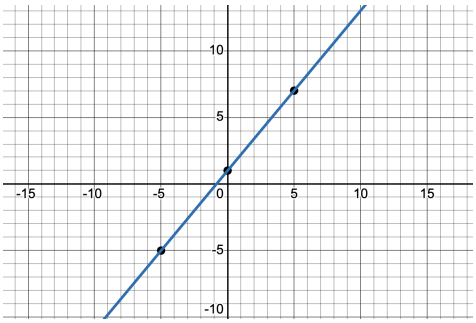
**A**




| x | -5 | -4 | -3 | -2 | -1 |
|---|---|---|---|---|---|
| y | 9 | 8 | 7 | 5 | 5 |

**2**

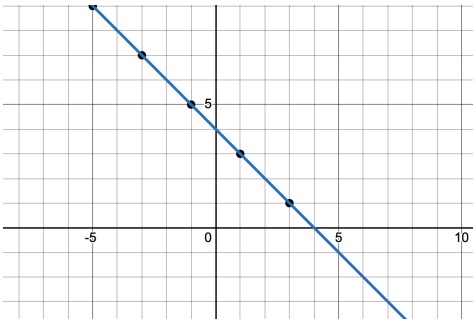
**B**




| x | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|
| y | -10 | -7 | -4 | -1 | 2 |

**3**

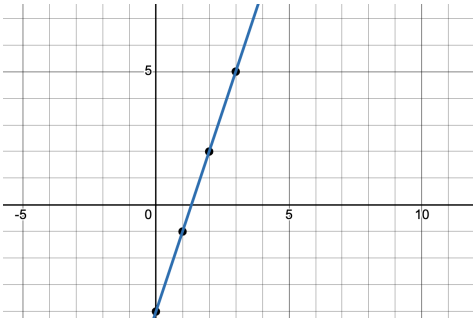
**C**




| x | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| y | 1 | 2.2 | 3.6 | 4.8 | 6 |

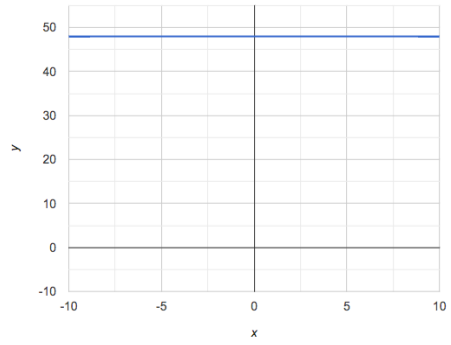
**4**

**D**

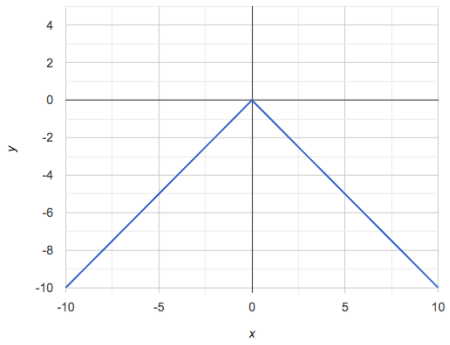


# Are All Graphs Linear?

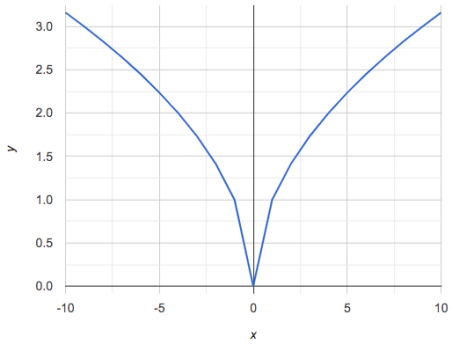
Beneath each graph circle **Linear** or **Not Linear**.



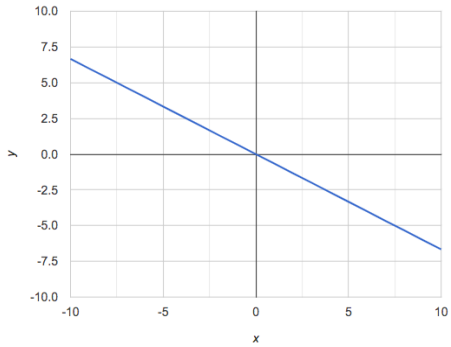

1) **Linear** or **Not Linear**?



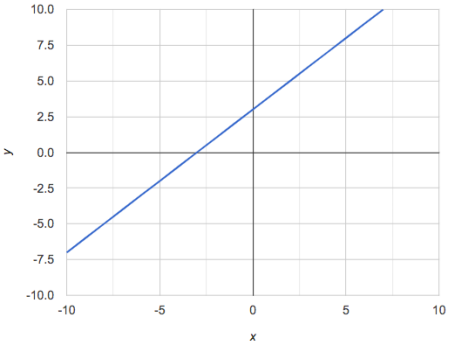

2) **Linear** or **Not Linear**?



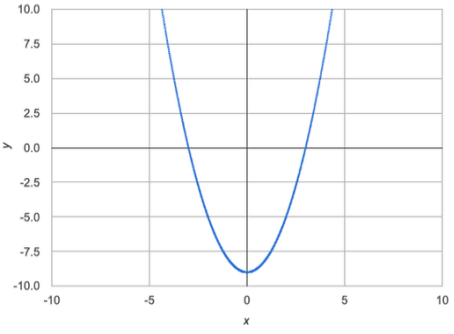

3) **Linear** or **Not Linear**?




4) **Linear** or **Not Linear**?




5) **Linear** or **Not Linear**?




6) **Linear** or **Not Linear**?

| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

# Are All Tables Linear?

Look at the six tables shown below.

1) Extend as many of the tables as you can by adding the next (x,y) pair in the sequence.

2) If the table is linear, write down your prediction of what the y-value will be when x = 0.

3) If the table is not linear, write **not linear** instead of an answer for y.

**A**

| x | -2 | -1 | 0 | 1 | 2 | |
|---|---|---|---|---|---|---|
| y | -2 | -3 | -4 | -5 | -6 | |

*when x=0, y will equal* _____

**B**

| x | 2 | 4 | 6 | 8 | 10 | |
|---|---|---|---|---|---|---|
| y | -12 | -16 | -20 | -24 | -28 | |

*when x=0, y will equal* _____

**C**

| x | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| y | 1 | 4 | 9 | 16 | 25 | |

*when x=0, y will equal* _____

**D**

| x | 5 | 6 | 7 | 8 | 9 | |
|---|---|---|---|---|---|---|
| y | 3 | 3 | 3 | 3 | 3 | |

*when x=0, y will equal* _____

**E**

| x | 1 | 2 | 3 | 4 | 5 | |
|---|---|---|---|---|---|---|
| y | 84 | 94 | 104 | 114 | 124 | |

*when x=0, y will equal* _____

**F**

| x | -10 | -9 | -8 | -7 | -6 | |
|---|---|---|---|---|---|---|
| y | $-1/10$ | $-1/9$ | $-1/8$ | $-1/7$ | $-1/6$ | |

*when x=0, y will equal* _____

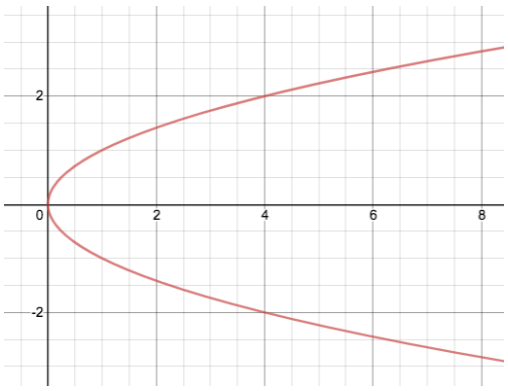
| What do you Notice? | What do you Wonder? |
|---|---|
| | |

# Linear, Non-linear, or Bust?

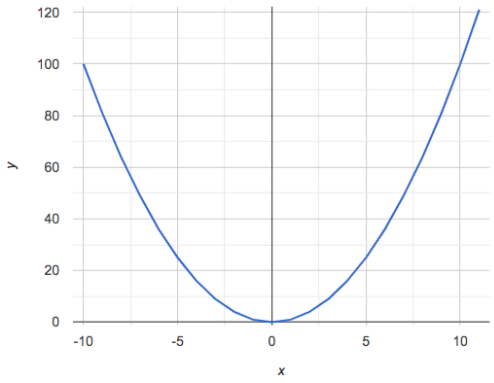
Circle whether each representation is of a linear function, a nonlinear function or is not a function at all!
**Remember: Functions will pass the Vertical Line Test, meaning they'll have exactly one y-value for each x-value!**

| x | y |
|---|---|
| 1 | 5 |
| 2 | 10 |
| 3 | 15 |
| 4 | 20 |
| 5 | 25 |
| 6 | 30 |
| 7 | 35 |

**1)** Linear Nonlinear Not a Function




**2)** Linear Nonlinear Not a Function



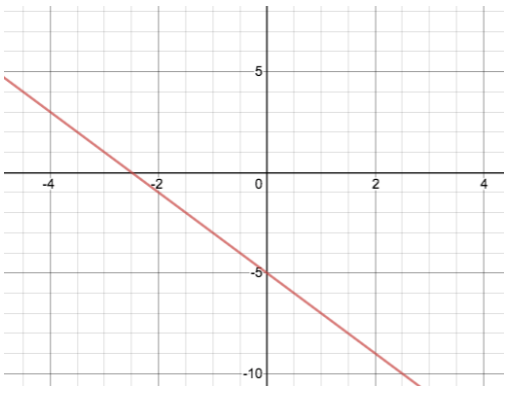

**3)** Linear Nonlinear Not a Function

| x | y |
|---|---|
| 1 | 1 |
| 2 | 4 |
| 3 | 9 |
| 4 | 16 |
| 5 | 25 |
| 6 | 36 |
| 7 | 49 |

**4)** Linear Nonlinear Not a Function

| x | y |
|---|---|
| 1 | 1 |
| 2 | 2 |
| 3 | 3 |
| 4 | 4 |
| 4 | 5 |
| 6 | 6 |
| 7 | 9 |

**5)** Linear Nonlinear Not a Function




**6)** Linear Nonlinear Not a Function

# Slope & y-Intercept from Tables (Intro)

**slope (rate)**: *how much y changes as x-increases by 1*
**y-intercept**: *the y-value when* $x = 0$

| x | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| y | -1 | 1 | 3 | 5 | 7 | 9 |

1) Compute the slope: _____

2) Compute the y-intercept: _____

3) What strategies did you use to compute the slope and y-intercept?

_____

_____

The slope and y-intercept in this table are harder to find, because the x-values don't go up by 1 and we can't see a value for $x = 0$.
**Try filling in the points that have been skipped to compute the slope and y-intercept.**

| x | 3 | 6 | 9 | 12 |
|---|---|---|---|---|
| y | 4 | 9 | 14 | 19 |

4) Compute the slope: _____

5) Compute the y-intercept: _____

The slope and y-intercept in this table are even harder to find, because the x-values are out of order!
**Calculate the slope and y-intercept from *any* two points!** Be sure to show your work.

| x | 3 | 20 | 5 | 9 | 1 |
|---|---|---|---|---|---|
| y | 5 | 56 | 11 | 23 | -1 |

6) Compute the slope: _____

7) Compute the y-intercept: _____

# Slope & y-Intercept from Tables (Practice)

| x | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| y | -1 | 2 | 5 | 8 | 11 | 14 |

1) slope: _____ y-intercept: _____

| x | -2 | -1 | 0 | 1 | 2 | 3 |
|---|---|---|---|---|---|---|
| y | 17 | 11 | 5 | -1 | -7 | -13 |

2) slope: _____ y-intercept: _____

| x | -3 | -2 | -1 | 0 | 1 | 2 |
|---|---|---|---|---|---|---|
| y | 0 | $\frac{2}{3}$ | $1\frac{1}{3}$ | 2 | $2\frac{2}{3}$ | $3\frac{1}{3}$ |

3) slope: _____ y-intercept: _____

| x | -1 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|
| y | -7 | -3 | 1 | 5 | 9 | 13 |

4) slope: _____ y-intercept: _____

| x | -5 | -4 | -3 | -2 | -1 | 0 |
|---|---|---|---|---|---|---|
| y | 1 | 2.5 | 4 | 5.5 | 7 | 8.5 |

5) slope: _____ y-intercept: _____

| x | -4 | -3 | - 2 | -1 | 0 | 1 |
|---|---|---|---|---|---|---|
| y | 0 | 0.6 | 1.2 | 1.8 | 2.4 | 3 |

6) slope: _____ y-intercept: _____

| x | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| y | 5 | 3 | 1 | -1 | -3 | -5 |

7) slope: _____ y-intercept: _____

| x | -4 | -2 | 0 | 2 | 4 | 6 |
|---|---|---|---|---|---|---|
| y | 0 | 4 | 8 | 12 | 16 | 20 |

★ slope: _____ y-intercept: _____

# Identifying Slope in Tables

$$slope = \frac{y_2 - y_1}{x_2 - x_1}$$

Can you identify the **slope** for the functions represented in each of these tables?
*Note: Some tables may have their rows out of order!*

**1**

| x | y |
|---|---|
| -1 | -3 |
| 4 | 12 |
| 8 | 21 |
| 9 | 24 |

slope/rate: _____

**2**

| x | y |
|---|---|
| -5 | 35 |
| -3 | 21 |
| 0 | 0 |
| 5 | -35 |

slope/rate: _____

**3**

| x | y |
|---|---|
| 12 | 15 |
| 17 | 17 |
| 13 | 15.4 |
| 20 | 18.2 |

slope/rate: _____

**4**

| x | y |
|---|---|
| 1 | 39 |
| 4 | 31.5 |
| 3 | 34 |
| 7 | 24 |

slope/rate: _____

**5**

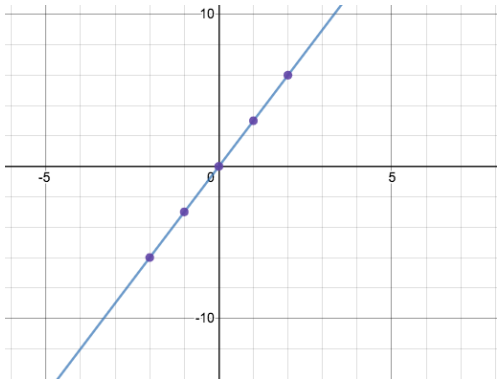
| x | y |
|---|---|
| 13 | 57 |
| 0 | 41.4 |
| 8 | 51 |
| -2 | 39 |

slope/rate: _____

# Identifying Slope and y-intercept in Graphs

Can you identify the **slope** and **y-intercept** for each of these graphs?

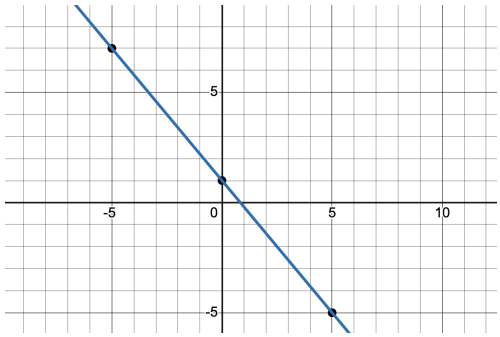
1




slope/rate: _____

y-intercept: _____

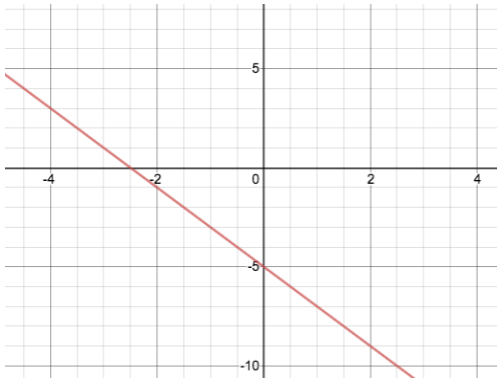
2




slope/rate: _____

y-intercept: _____

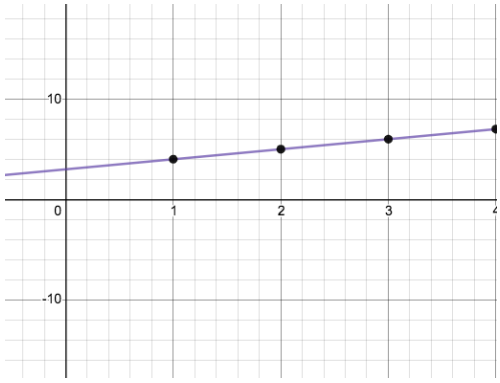
3




slope/rate: _____

y-intercept: _____

4

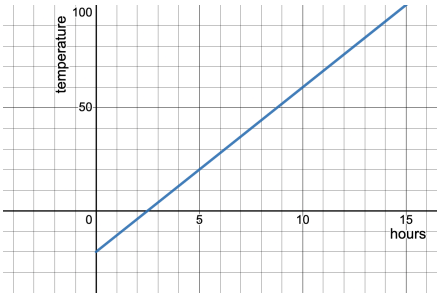
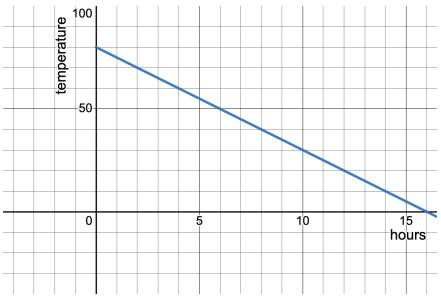
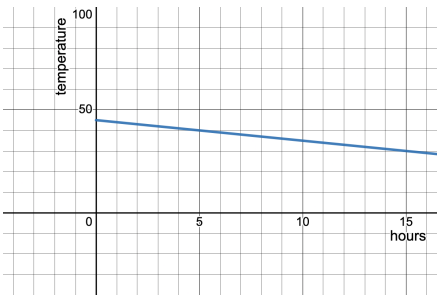
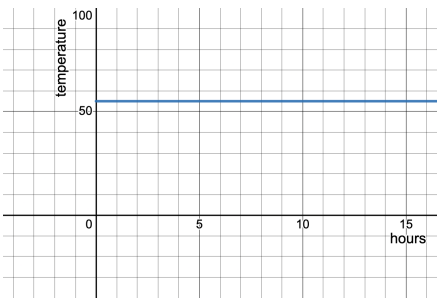



slope/rate: _____

y-intercept: _____

# What Story does the Graph tell?

For each of the Graphs below, write the story that it tells. *(The first one has been done for you.)*

| | | |
|---|---|---|
| 1 |  | The temperature started at -20 degrees and increased by 8 degrees per hour. |
| 2 |  | |
| 3 |  | |
| 4 |  | |

# What Story does the Table tell?

For each of the Tables below, write the story that it tells.

1

| maple syrup produced (gallons) | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| gallons of sap boiled | 0 | 40 | 80 | 120 | 160 |

2

| seconds on stove | 0 | 10 | 20 | 30 | 40 | 50 |
|---|---|---|---|---|---|---|
| water temp in deg F | 50 | 59 | 68 | 77 | 86 | 95 |

3

| tickets sold | 0 | 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|
| profit in dollars | -560 | -360 | -160 | 40 | 240 |

4

| bowls served | 0 | 10 | 20 | 30 | 40 |
|---|---|---|---|---|---|
| gallons of gumbo in the pot | 19 | 18 | 17 | 16 | 15 |

5

| month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| hours of daylight in Berlin | 8.3 | 9.8 | 11.9 | 13.8 | 15.8 | 16.9 | 16.4 | 14.8 | 12.8 | 10.8 | 8.8 | 7.8 |

# 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…

*(EXAMPLE (* _____ *)* _____ *)*
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ *)*
function name | input(s) | what the function produces

---

### 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…

*(EXAMPLE (* _____ *)*
function name | input(s)

_____ *)*
what the function produces

*(EXAMPLE (* _____ *)* _____ *)*
function name | input(s) | what the function produces

# 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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

## Definition

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

(define ( _____ )
function name | variable(s)

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

# Danger and Target Movement

**Directions:** Use the Design Recipe to write a function `update-danger`, which takes in the danger's x-coordinate and produces the next x-coordinate, which is 50 pixels to the 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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

---

**Directions:** Use the Design Recipe to write a function `update-target`, which takes in the target's x-coordinate and produces the next x-coordinate, which is 50 pixels to the right.

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

# Problem Decomposition

**Sometimes a problem is too complicated to solve all at once:**

- Maybe there are too many variables.
- Maybe there is so much information that we can't get a handle on it!
- Maybe we'll be less likely to make mistakes if we think about the parts one at a time.

**Problem Decomposition** allows us to break complicated problems down into simpler pieces… and then solve by working with the pieces. There are two strategies:

- **Top-Down**:
  - Start with the "big picture", writing functions or equations that describe the connections between parts of the problem.
  - Then, work on defining those parts.
- **Bottom-Up**:
  - Start with the smaller parts, writing functions or equations that describe the parts we understand.
  - Then, connect those parts together to solve the whole problem.

You may find that one strategy works better for some types of problems than another, so make sure you're comfortable using both of them!

# Word Problems: revenue, cost

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

**Contract and Purpose Statement**

Every contract has three parts…

; _____ : _____ -> _____
function name Domain Range

; _____
what does the function do?

**Examples**

Write some examples, then circle and label what changes…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

---

**Directions:** Use the Design Recipe to write a function `cost`, which takes in the number of glasses sold and calculates the total cost of materials if each glass costs $.30 to make.

**Contract and Purpose Statement**

Every contract has three parts…

; _____ : _____ -> _____
function name Domain Range

; _____
what does the function do?

**Examples**

Write some examples, then circle and label what changes…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

# Word Problem: profit

**Directions:** Use the Design Recipe to write a function `profit` that calculates total profit from glasses sold, which is computed by subtracting the total cost from the total revenue.

## 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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

## Definition

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

(define ( _____ )
function name | variable(s)

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

# Profit - More than one Way!

Four students defined the same `revenue` and `cost` functions, shown below:

```
(define (revenue g) (* 1.75 g))
(define (cost g) (* 0.3 g))
```

They then came up with **four different definitions** for `profit`:

**Khalil:**
```
(define (profit g) (- (* 1.75 g) (* 0.3 g)))
```

**Samaria:**
```
(define (profit g) (* (- 1.75 0.3) g))
```

**Alenka:**
```
(define (profit g) (* 1.45 g))
```

**Fauzi:**
```
(define (profit g) (- (revenue g) (cost g)))
```

1) Which of these four definitions do you think is "best", and why?

2) If lemons get more expensive, which definitions of `profit` need to be changed?

3) If Sally raises her prices, which definitions of `profit` need to be changed?

4) Which definition of `profit` is the most flexible? Why?

# Top Down or Bottom Up

Jamal's trip requires him to drive 20 mi to the airport, fly 2,300 mi, and then take a bus 6 mi to his hotel. His average speed driving to the airport is 40 mph, the average speed of an airplane is 575 mph, and the average speed of his bus is 15 mph.
Aside from time waiting for the plane or bus, how long is Jamal in transit?

| **Bear's Strategy:** | **Lion's Strategy:** |
|---|---|
| $\text{Drive Time} = 20 \text{ miles} \times \frac{1 \text{ hour}}{40 \text{ miles}} = 0.5 \text{ hours}$<br>$\text{Fly Time} = 2300 \text{ miles} \times \frac{1 \text{ hour}}{575 \text{ miles}} = 4 \text{ hours}$<br>$\text{Bus Time} = 6 \text{ miles} \times \frac{1 \text{ hour}}{15 \text{ miles}} = 0.4 \text{ hours}$<br>$\text{In Transit Time} = \text{Drive Time} + \text{Fly Time} + \text{Bus Time}$<br>$0.5 + 4 + 0.4 = 4.9 \text{ hours}$ | $\text{In Transit Time} = \text{Drive Time} + \text{Fly Time} + \text{Bus Time}$<br>$\text{Drive Time} = 20 \text{ miles} \times \frac{1 \text{ hour}}{40 \text{ miles}} = 0.5 \text{ hours}$<br>$\text{Fly Time} = 2300 \text{ miles} \times \frac{1 \text{ hour}}{575 \text{ miles}} = 4 \text{ hours}$<br>$\text{Bus Time} = 6 \text{ miles} \times \frac{1 \text{ hour}}{15 \text{ miles}} = 0.4 \text{ hours}$<br>$0.5 + 4 + 0.4 = 4.9 \text{ hours}$ |

1) Whose Strategy was Top Down? How do you know?

2) Whose Strategy was Bottom Up? How do you know?

3) Which way of thinking about the problem makes more sense to you?

## What's happening with that Math?!

When calculating Jamal's drive time, we multiplied distance by speed. More specifically, we multiplied the starting value ( $20 \text{ miles}$ ) by $\frac{1 \text{ hour}}{40 \text{ miles}}$. Why? Why not reverse it, to use $\frac{40 \text{ miles}}{1 \text{ hour}}$, as stated in the problem?

$\text{Time}$ is the desired outcome. Looking at the units, we can see that speed must have $\text{miles}$ as its denominator to *cancel out* the $\text{miles}$ in the starting value.

$$\frac{20 \text{ miles}}{1} \times \frac{1 \text{ hour}}{40 \text{ miles}} = \frac{20 \cancel{\text{ miles}} \times 1 \text{ hour}}{40 \cancel{\text{ miles}}} = \frac{20}{40} \text{hour} = \frac{1}{2} \text{hour}$$

# Inequalities

**Sometimes we want to *ask questions* about data:**

- Is `x` greater than `y`?
- Is one string equal to another?

These questions are answered with a new data type called a ***Boolean***.

> Unlike Numbers, Strings, and Images, Booleans have only two possible values. A Boolean value is either `true` or `false`.
> You already know some functions that produce Booleans, such as `<` and `>`!
>
> Our programming language has them, too. We can evaluate:

| `(< 3 4)` | `(> 2 10)` | `(= -10 19)` |
|---|---|---|
| "3 is less than 4" is `true` | "2 is greater than 10" is `false` | "-10 is equal to 19" is `false` |

**We can also ask more complicated questions:**

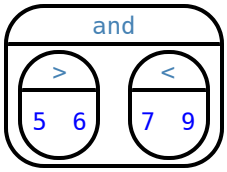
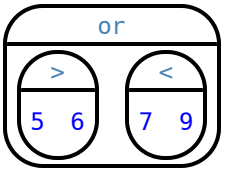
- Is the elephant small enough and light enough to ride in the boat?
- Do we have enough rice and enough time to make it for dinner?

Our programming language uses the `and` and `or` functions to combine to ***Simple Inequalities*** to make a ***Compound Inequality***.

- The `and` function will return true if **both** sub-expressions are `true`.
- The `or` function will return true if **at least one** sub-expression is `true`.

| `(and (> 5 6) (< 7 9))` | `(or (> 5 6) (< 7 9))` |
|---|---|
| "5 is greater than 6 *and* 7 is less than 9" | "5 is greater than 6 *or* 7 is less than 9" |
| This will evaluate to `false`, because the expressions aren't both `true`. | This will evaluate to `true`, because at least one of the expressions is `true`. |

The Circles of Evaluation work the same way with Booleans that they do with Numbers, Strings and Images.

| | |
|---|---|
|  |  |





Video games use Booleans for many things including:

- asking when a player's health is equal to zero
- determining whether two characters are close enough to bump into one another
- figuring out if a character's coordinates put it off the edge of the screen

# Boolean Functions

Make a prediction about what each function in the Boolean Starter File does.

Now, experiment with the functions. Fill in the blanks below so that each of the five functions returns `true`.

1) `(odd? _____)`

2) `(even? _____)`

3) `(less-than-one? _____)`

4) `(continent? _____)`

5) `(primary-color? _____)`

Fill in the blanks below so that each of the five functions returns `false`.

6) `(odd? _____)`

7) `(even? _____)`

8) `(less-than-one? _____)`

9) `(continent? _____)`

10) `(primary-color? _____)`

All 5 of these functions produce Booleans. How would you describe what a Boolean is?

# Simple Inequalities

Each inequality expression in the first column contains a number.
Decide whether or not that number is a solution to the expression and place it in the appropriate column.
Then identify 4 *solution* values and 4 *non-solution* values for `x`.

- **Solutions** will make the expression `true`.
- **Non-Solutions** will make the expression `false`.

You can see graphs of the solution sets of these inequalities and test out each of your lists in the Simple Inequalities Starter File.
*The comments in the starter file will help you learn how it works!*

★ Challenge yourself to use negatives, positives, fractions, decimals, etc. for your `x` values.

| | Expression | 4 solutions that evaluate to `true` | 4 non-solutions that evaluate to `false` |
|---|---|---|---|
| a | `(> x 2)` | | |
| b | `(<= x -2)` | | |
| c | `(< x 3.5)` | | |
| d | `(>= x -1)` | | |
| e | `(> x -4)` | | |
| f | `(<> x 2)` | | |

1) For which inequalities was the number from the expression part of the solution?

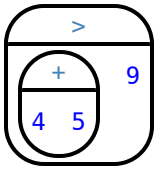
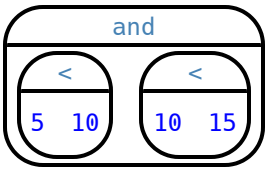
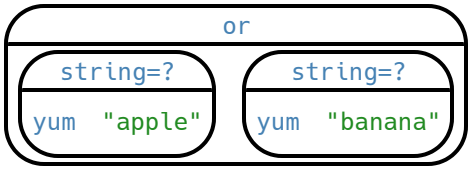
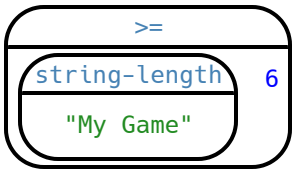
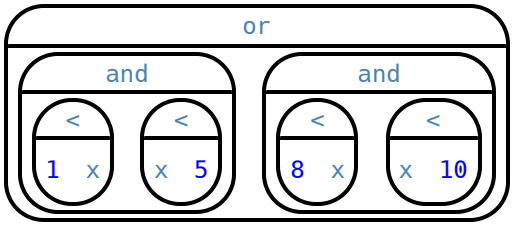
2) For which inequalities was the number from the expression not part of the solution?

3) For which inequalities were the solutions on the left end of the number line?

4) For which inequalities were the solutions on the right end of the number line?

# Converting Circles of Evaluation to Code

Convert each Circle of Evaluation on the left-hand side to Code.

| | Circle of Evaluation | Code |
|---|---|---|
| 1 |  | |
| 2 |  | |
| 3 |  | |
| 4 |  | |
| 5 |  | |

# Compound Inequalities - Practice

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

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

What will this evaluate to? Why? _____

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

What will this evaluate to? Why? _____

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

What will this evaluate to? Why? _____

4) Write the contracts for `and` & `or` in your Contracts page.

# Compound Inequality Warmup

1) What are 4 solutions for $x > 5$ ?

2) What are 4 non-solutions for $x > 5$ ?

3) What are 4 solutions for $x \leq 15$ ?

4) What are 4 non-solutions for $x \leq 15$ ?

5) What 4 numbers are in the solution set of $x > 5$ **and** $x \leq 15$ , making both of these inequalities true?

6) How would that be different from the solution set of $x > 5$ **or** $x \leq 15$ , making at least one of these inequalities true?

# Exploring Compound Inequalities

*This page is designed to accompany the Compound Inequalities Starter File. When you click "Run" you will see 4 graphs. The first two are simple inequalities and the second two are compound inequalities.*

1) What does `and-intersection` do?

2) Why is the dot on 5 red and the circle on 15 green?

3) Do you think every graph made with `and-intersection` will have a red dot at one end and a green dot at the other? Why or why not?

4) What does `or-union` do?

5) Why did the graph of this `or-union` result in the whole numberline being shaded blue?

6) Not all graphs of `or-union` will look like this. Can you think of a pair of inequalities whose union won't shade the whole graph?

**Change the function definition on *line 8* to** $x < 5$ **and the definition on *line 9* to** $x \geq 15$**.**
Before you click "Run", think about what the new graphs of `and-intersection` and `or-union` will look like. Then test them out.

7) What does the new `and-intersection` graph look like?

8) What does the new `or-union` graph look like?

9) Why is the dot for 5 still red and the dot for 15 still green?

10) Which of the 8 numbers from the list are part of the solution set? _____

How do you know? _____

11) Is 3 part of the solution set? _____ Explain. _____

12) Is 10 part of the solution set? _____ Explain. _____

# Compound Inequalities: Solutions & Non-Solutions

For each Compound Inequality listed below, identify 4 *solutions* and 4 *non-solutions*, unless the solution set includes **all real numbers** or there are **no solutions**.

- Solutions for ***intersections*** (which use **and**) will make both of the expressions `true`.
- Solutions for ***unions*** (which use **or**) will make at least one of the expressions `true`.

Pay special attention to the numbers in the sample expression! Challenge yourself to use negatives, positives, fractions, decimals, etc.

*The first two have been done for you - Answers will vary!*

| | Expression | 4 solutions that evaluate to `true` | 4 non-solutions that evaluate to `false` |
|---|---|---|---|
| a | `x > 5` and `x < 15` | **6, 9.5, 12, 14.9** | **-2, 5, 15, 16.1** |
| b | `x > 5` or `x < 15` | **All real numbers** | **No non-solutions** |
| c | `x <= -2` and `x > 7` | | |
| d | `x <= -2` or `x > 7` | | |
| e | `x < 3.5` and `x > -4` | | |
| f | `x < 3.5` or `x > -4` | | |
| g | `x >= -1` and `x > -5` | | |
| h | `x >= -1` or `x > -5` | | |
| i | `x < -4` and `x > 2` | | |

1) Could there ever be a union with *no solutions*? Explain your thinking.

2) Could there ever be an intersection whose solution is *all real numbers*? Explain your thinking.

# Compound Inequality Functions

Each of the plots below was generated using the code `(inequality comp-ineq (list -1 0 1.6 3 5.2 7 8.1 9))`. Using the numbers 3 and 7, write the code to define `comp-ineq` for each plot.
*Note: The example is defined using 0 and 8.1 rather than 3 and 7.*


**code:** `(define (comp-ineq x) (and (> x 0) (<= x 8.1)))`


**code:** _____


**code:** _____


**code:** _____


**code:** _____

# Sam the Butterfly

*Open the Sam the Butterfly Starter File starter file and click "Run". (Hi, Sam!)* Move Sam around the screen using the arrow keys.

1) What do you Notice about the program?

2) What do you Wonder?

3) What do you see when Sam is at (0,0)? Why is that?

4) What changes as the butterfly moves left and right?

5) Sam is in a 640 × 480 yard. Sam's mom wants Sam to stay in sight… *How far to the left and right can Sam go and still remain visible?*

6) Write an inequality to complete each of the following statements:

*Sam hasn't gone off the left edge of the screen as long as…*

*Sam hasn't gone off the right edge of the screen as long as…*

7) Draw the Circle of Evaluation for each inequality you wrote above.

8) Translate each of the Circles of Evaluation into code.

code:

code:

# Left and Right

**Directions:** Use the Design Recipe to write a function `safe-left?`, which takes in an x-coordinate and checks to see if it's greater than -50.

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

---

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

**Contract and Purpose Statement**

Every contract has three parts…

; _____ : _____ -> _____
function name Domain Range

; _____
what does the function do?

**Examples**

Write some examples, then circle and label what changes…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

# Word Problem: onscreen?

**Directions:** Use the Design Recipe to write a function `onscreen?`, which takes in an x-coordinate and checks to see if Sam is safe on the left while also being safe on the right.

## 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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

## Definition

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

(define ( _____ )
function name | variable(s)

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

# Piecewise Functions in a Nutshell

- Sometimes we want to build functions that act differently for different inputs. For example, suppose a business charges $10/pizza, but only $5/pizza for orders of six or more. How could we write a function that computes the total price based on the number of pizzas?
- In math, **Piecewise Functions** are functions that can behave one way for part of their Domain, and another way for a different part. In our pizza example, our function would act like $cost(pizzas) = 10 * pizzas$ for anywhere from 1-5 pizzas. But after 5, it acts like $cost(pizzas) = 5 * pizzas$.
- Piecewise functions are divided into "pieces". Each piece is divided into two parts:
  1. How the function should behave
  2. The domain where it behaves that way
- Our programming language can be used to write piecewise functions, too! Just as in math, each piece has two parts:

```
(define (cost pizzas)
  (cond
    [(>= pizzas 6) (* 5 pizzas)]))
```

Piecewise functions are powerful, and let us solve more complex problems. We can use piecewise functions in a video game to add or subtract from a character's x-coordinate, moving it left or right depending on which key was pressed.

# Red Shape - Explore

1) Open the Red Shape Starter File, and read through the code you find there. This code contains new programming that you haven't seen yet! Take a moment to list everything you Notice, and then everything you Wonder…

| What do you Notice? | What do you Wonder? |
| --- | --- |
| | |

2) What happens if you click "Run" and type `(red-shape "ellipse")`?

---

3) Add **another example** for "triangle".

4) Add another line of code to the definition, to define what the function should do with the input "triangle".

5) Come up with some new shapes, and add them to the code. Make sure you include examples or you will get an error message!

6) In your own words, describe how *piecewise functions* work in this programming environment.

---

---

---

---

# Word Problem: red-shape

**Directions:** A friend loves red shapes so we've decided to write a program that makes it easy to generate them. Write a function called `red-shape` which takes in the name of a shape and makes a 20-pixel, solid, red image of the shape.

## Contract and Purpose Statement

Every contract has three parts…

```
; red-shape : String -> Image
```
function name | Domain | Range

```
; Given a shape name, produce a solid, red, 20-pixel image of the shape.
```
what does the function do?

## Examples

Write some examples, then circle and label what changes…

(*EXAMPLE* (`red-shape "circle"`) `(circle 20 "solid" "red")`)
function name | input(s) | what the function produces

(*EXAMPLE* (`red-shape "triangle"`) `(triangle 20 "solid" "red")`)
function name | input(s) | what the function produces

(*EXAMPLE* (`red-shape "rectangle"`) `(rectangle 20 20 "solid" "red")`)
function name | input(s) | what the function produces

(*EXAMPLE* (`red-shape "star"`) `(star 20 "solid" "red")`)
function name | input(s) | what the function produces

## Definition

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

(define ( ) )
function name | variable(s)

```
(cond
  [                                  ]
  [                                  ]
  [                                  ]
  [                                  ]
  [                                  ]))
```

# Word Problem: update-player

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

## Contract and Purpose Statement

Every contract has three parts…

; _____ : _____ -> _____
function name | Domain | Range

; _____
what does the function do?

## Examples

Write some examples, then circle and label what changes…

(*EXAMPLE* (*update-player* *300* *"up"*) _____)
function name | input(s) | what the function produces

(*EXAMPLE* (_____) _____)
function name | input(s) | what the function produces

(*EXAMPLE* (_____) _____)
function name | input(s) | what the function produces

(*EXAMPLE* (_____) _____)
function name | input(s) | what the function produces

## Definition

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

(define (_____)
function name | variable(s)

(*cond*

[_____ _____]

[_____ _____]

[_____ _____]))

# Challenges for update-player

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

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

```
(EXAMPLE (update-player _____ _____)
  _____)
(EXAMPLE (update-player _____ _____)
  _____)
```

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

```
(EXAMPLE (update-player _____ _____)
  _____)
(EXAMPLE (update-player _____ _____)
  _____)
```

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

```
(EXAMPLE (update-player _____ _____)
  _____)
(EXAMPLE (update-player _____ _____)
  _____)
```

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

```
(EXAMPLE (update-player _____ _____)
  _____)
(EXAMPLE (update-player _____ _____)
  _____)
```

# Line Length Explore

*Sign in to WeScheme 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.

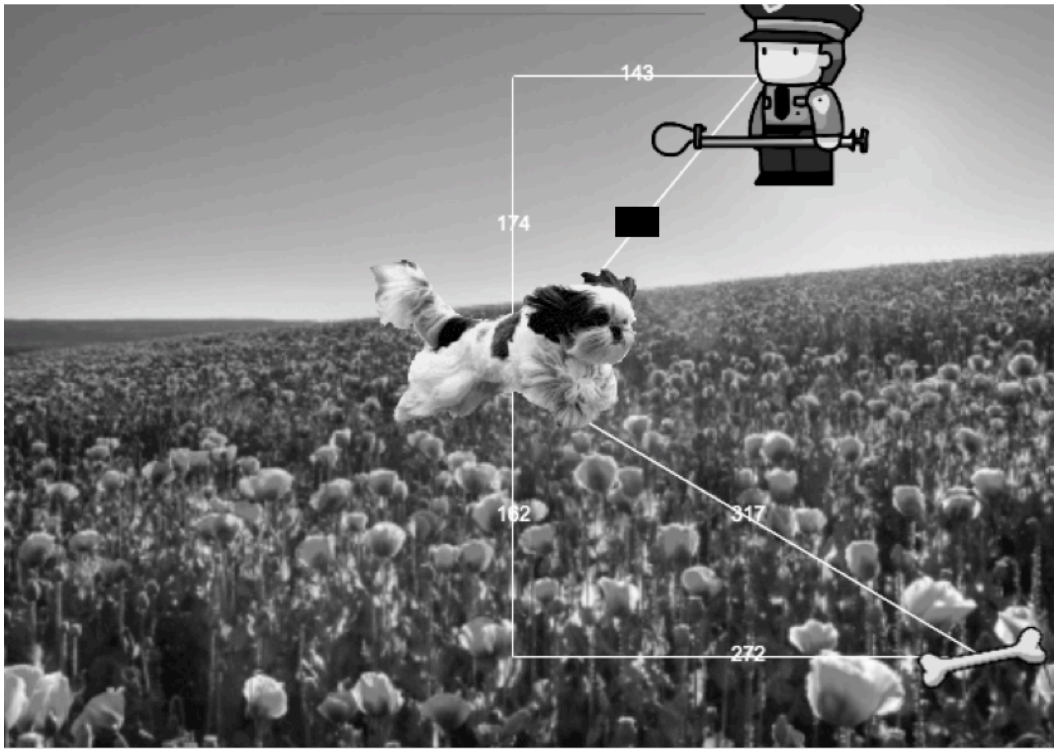
3) What does the `line-length` function *do?*

4) Why does it use conditionals?

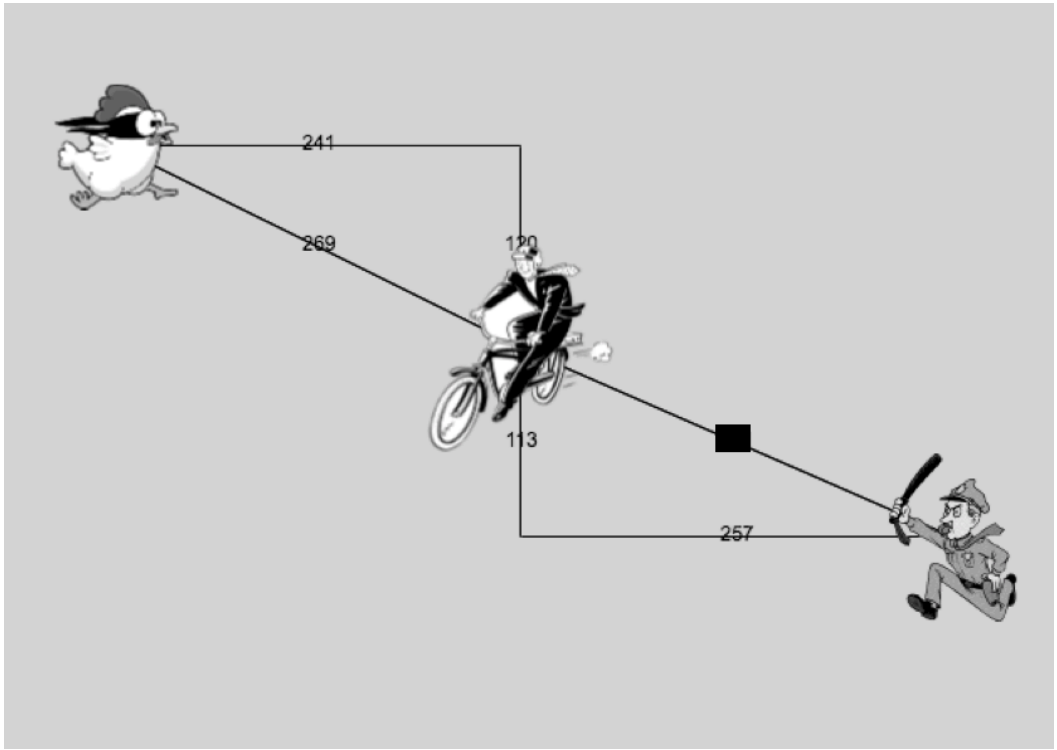
5) Why is the distance between two points always positive?

# 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!*



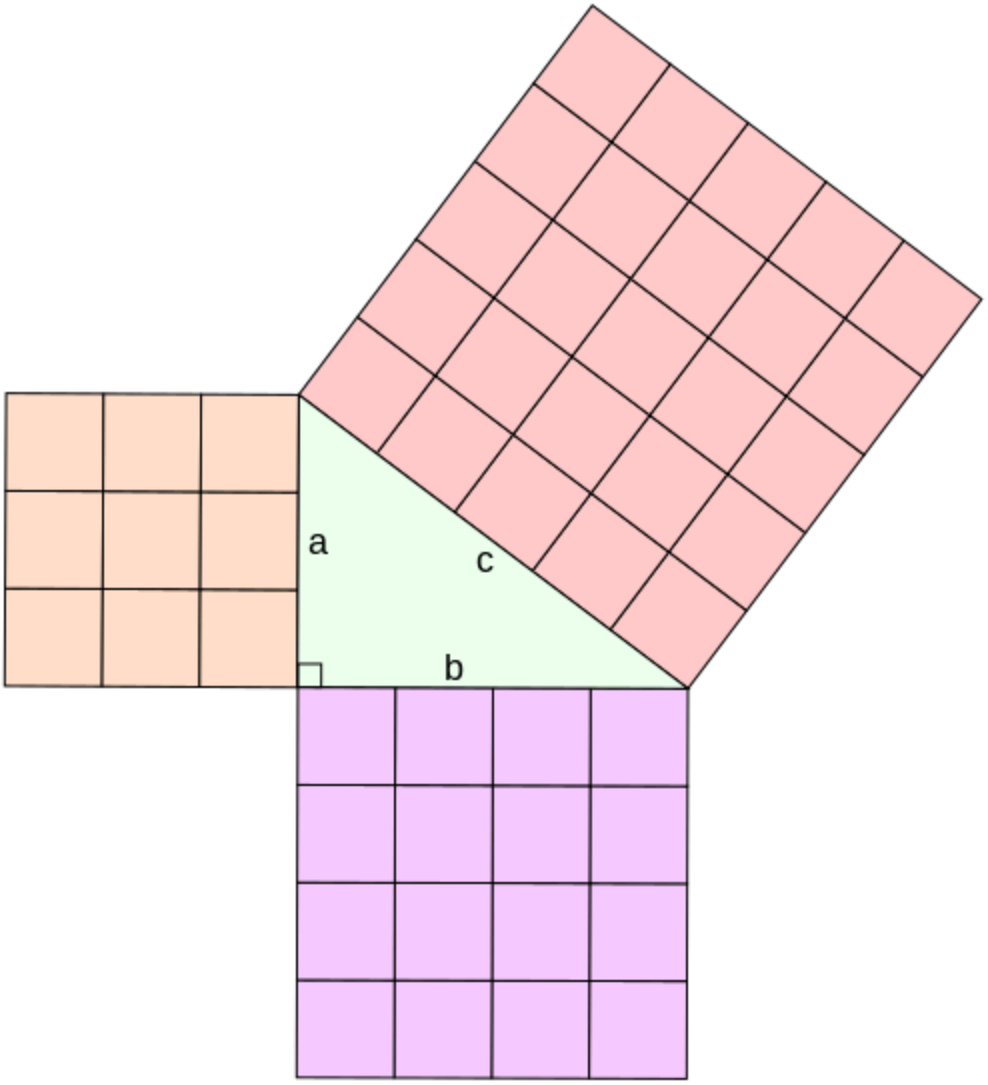

---




---

# 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.



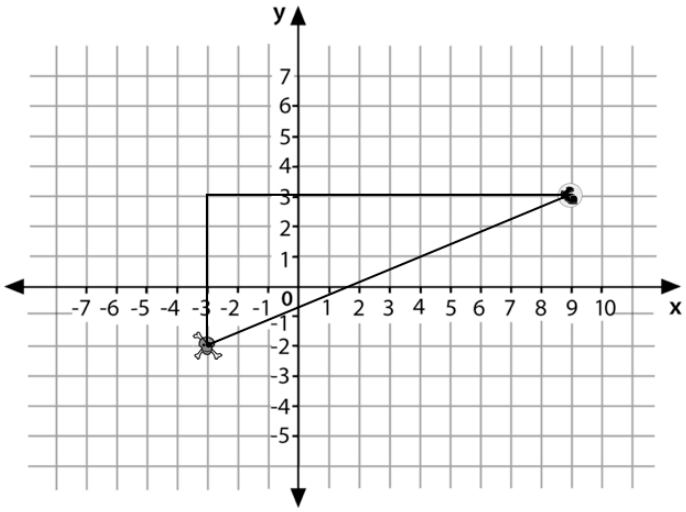

How would you describe the relationship you've observed between the three squares whose side-lengths are determined by the lengths of the sides of a right triangle?

# 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))))
```



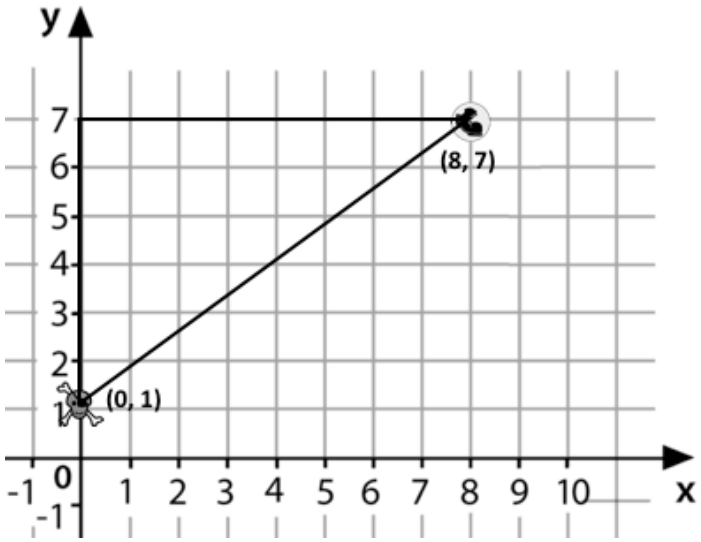

1) Where do the 9 and -3 come from? _____

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?

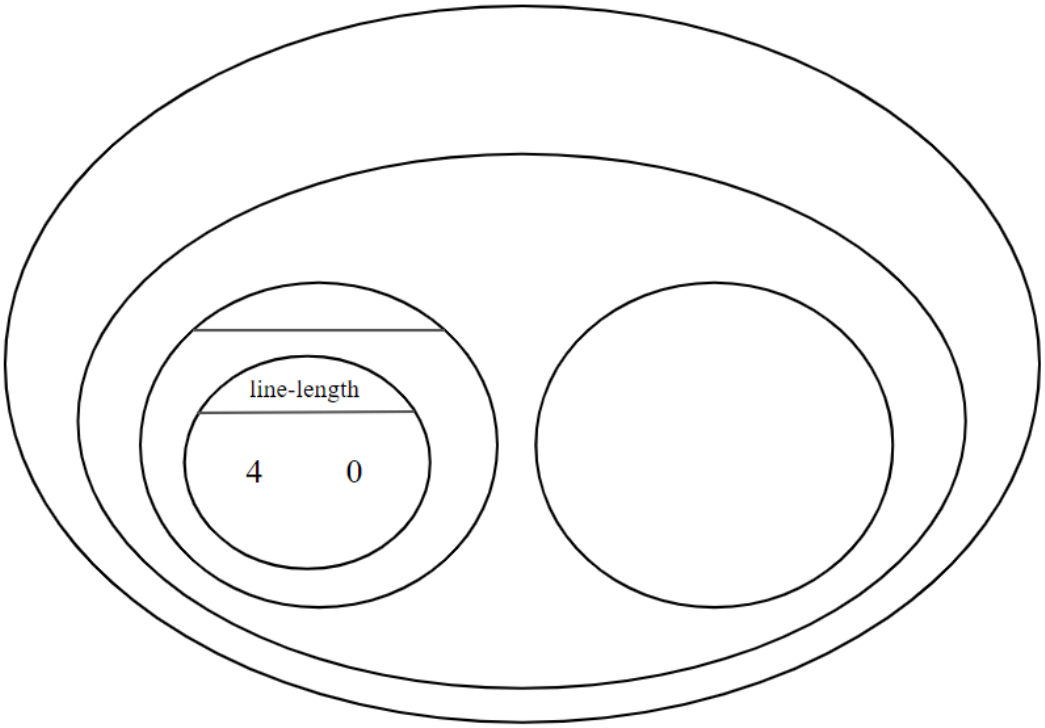
- We can use `line-length` to computer the horizontal and vertical distances and then use those to find the diagonal distance.
  - The horizontal distance between $x_1$ and $x_2$ is computed by `(line-length x2 x1)`.
  - The vertical distance between $y_2$ and $y_1$ is computed by `(line-length y2 y1)`.
- 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* $\sqrt{\ x}$



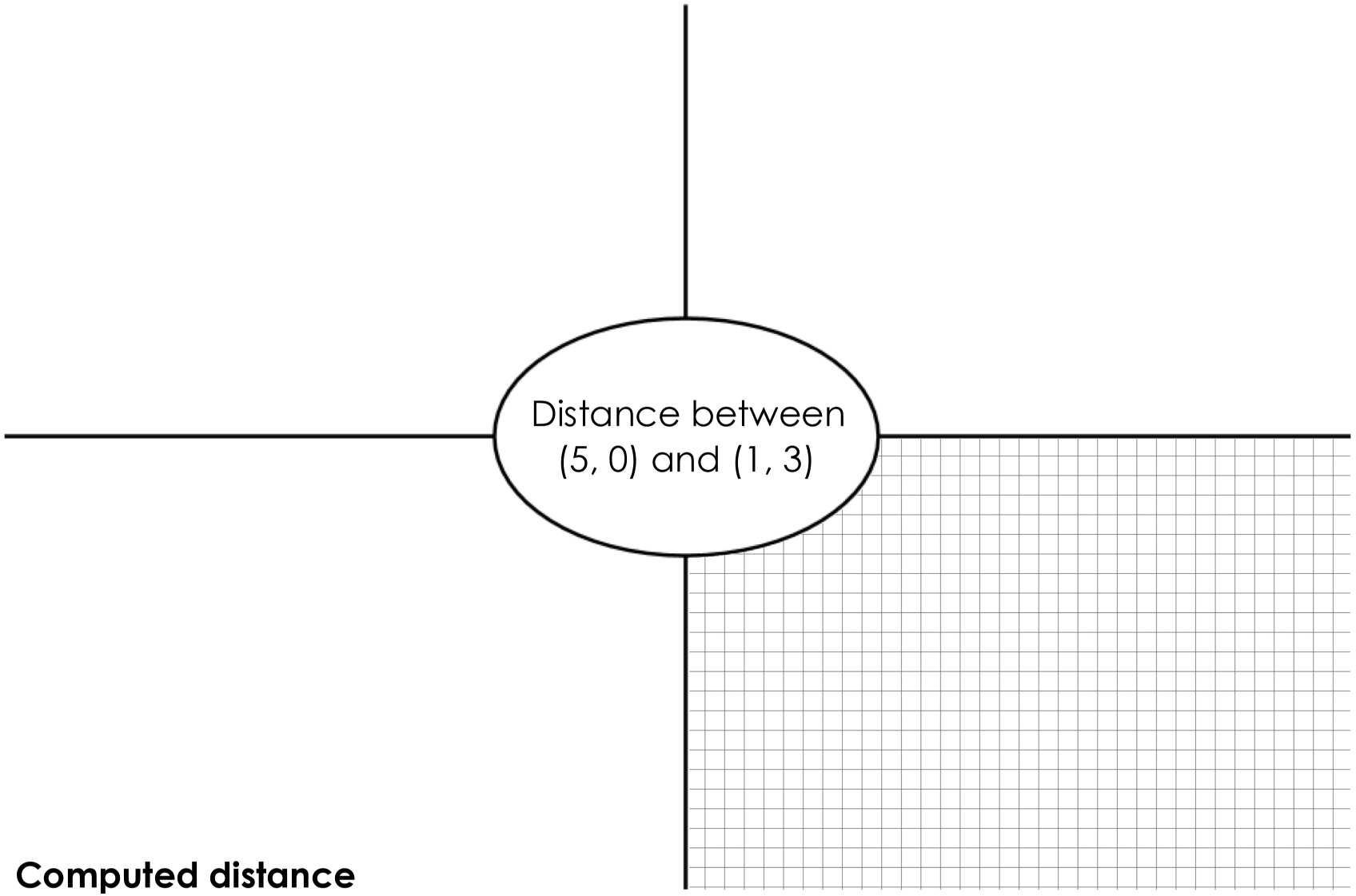

4) Convert the Circle of Evaluation to Code below.

---

---

**Circle of Evaluation**

**Code**

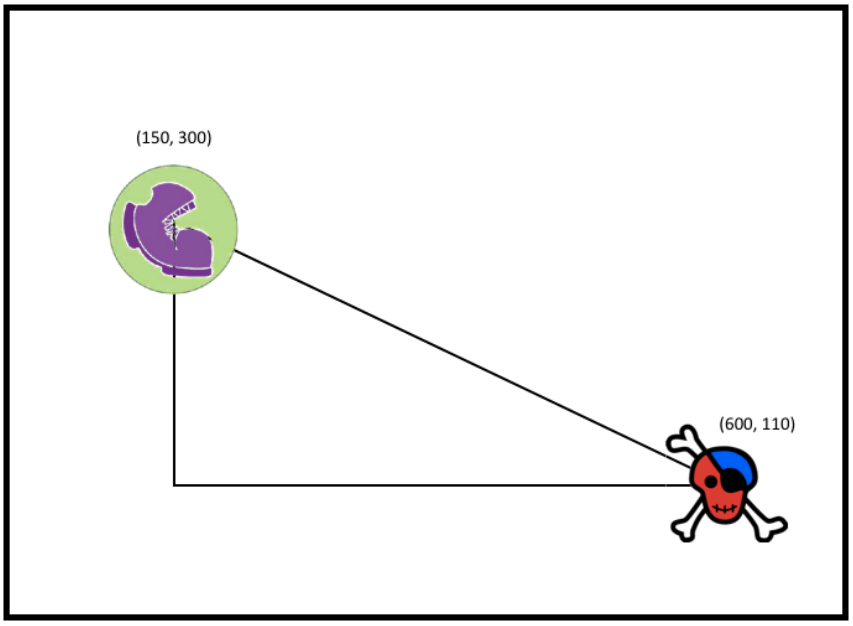



**Computed distance between (5, 0) and (1, 3)**

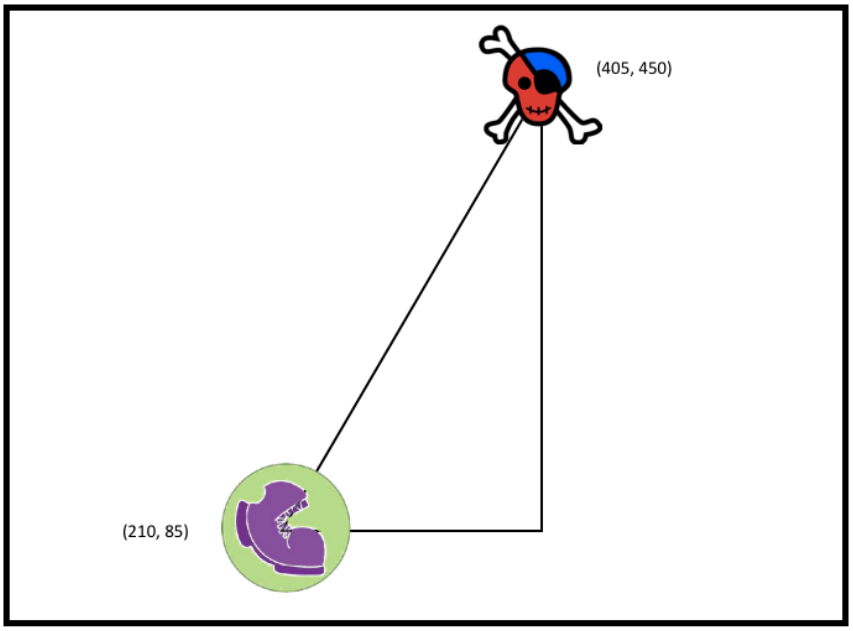
**Graph**

# Distance From Game Coordinates

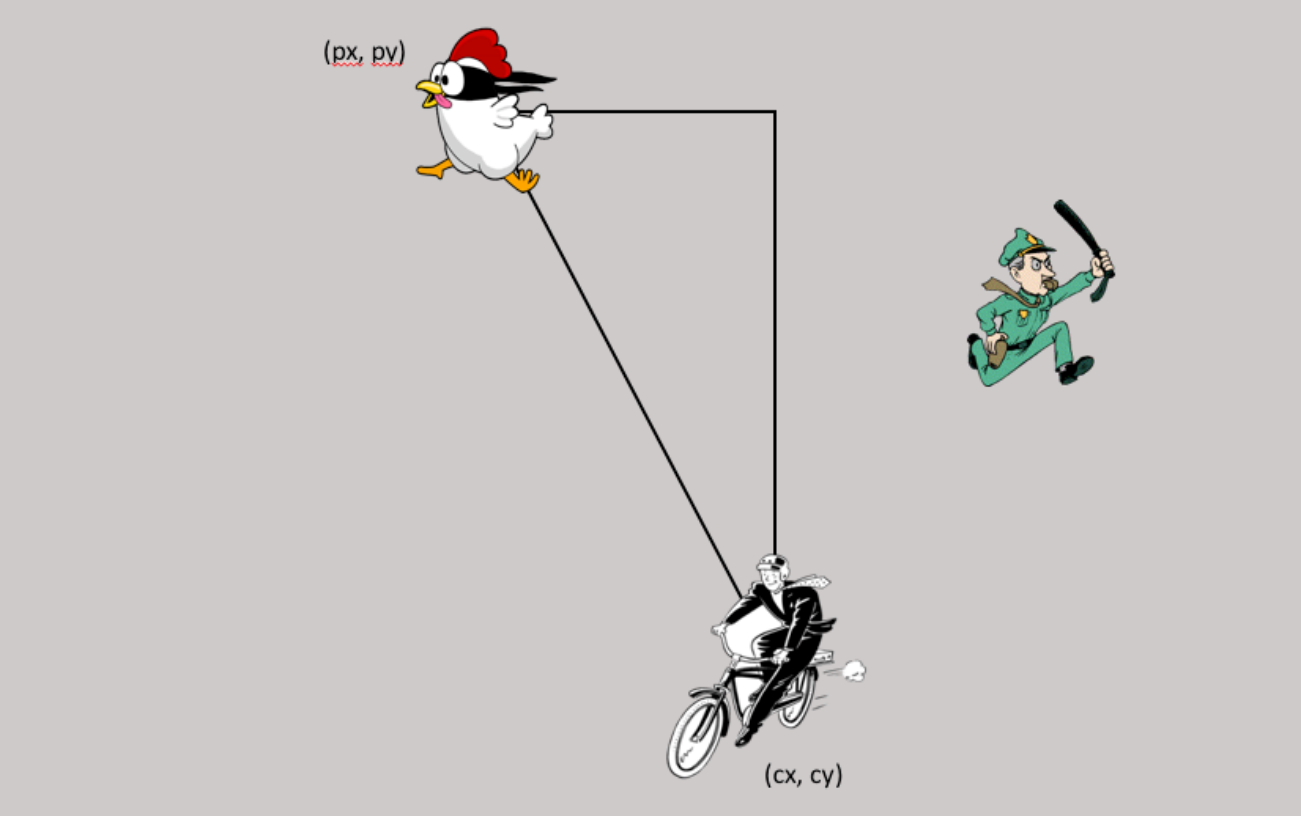
For each of the game screenshots, write the code to calculate the distance between the indicated characters. *The first one has been done for you.*




```
(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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

## Definition

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

(define ( _____ )
function name | variable(s)

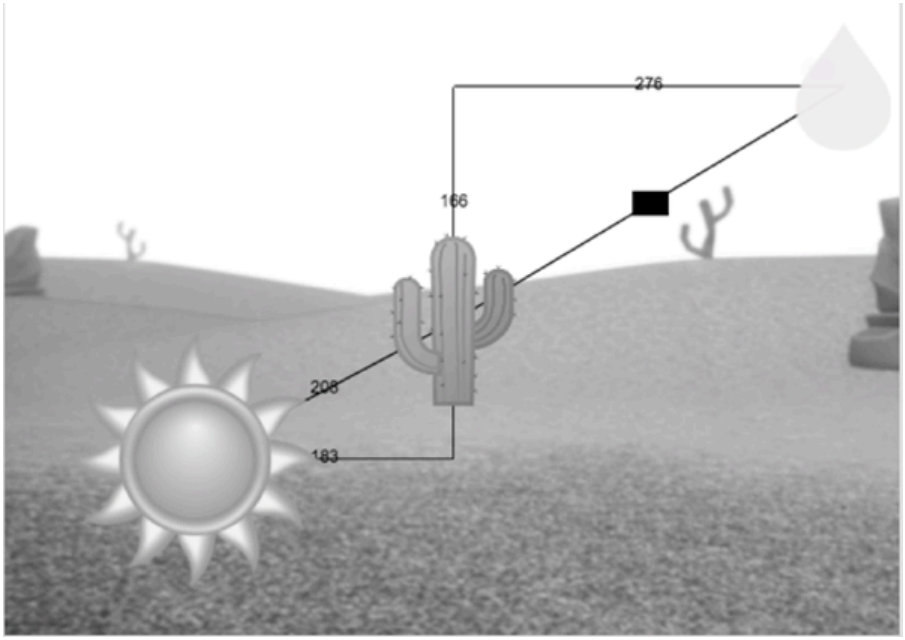
_____ )
what the function does with those variable(s)

# 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.

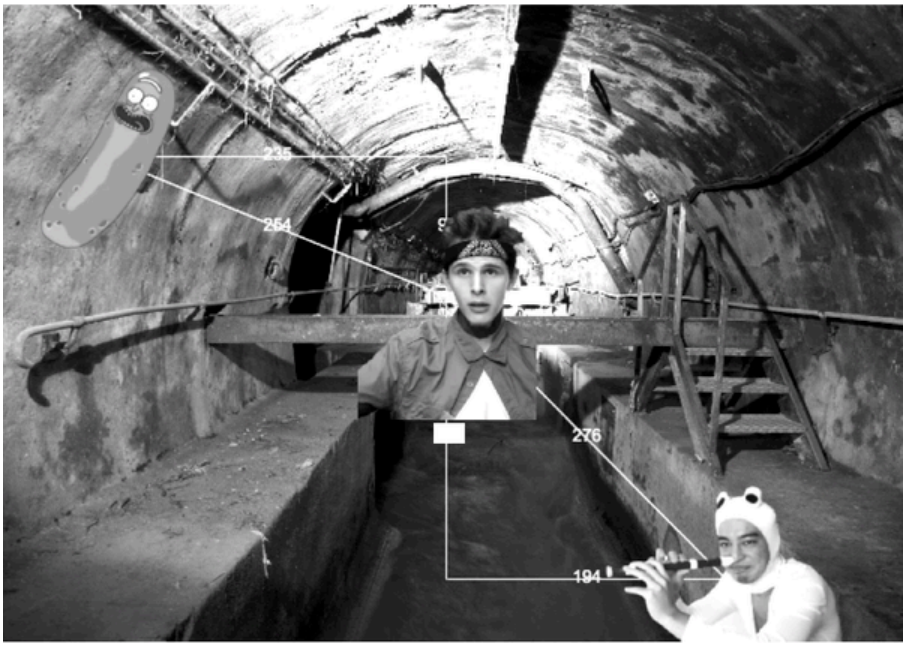
---

---




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

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

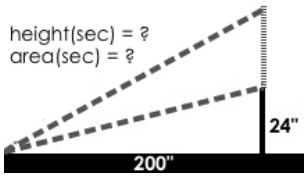



$\sqrt{276^2 - 194^2}$

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

# Top Down / Bottom Up

A retractable flag pole starts out 24 inches tall, and grows taller at a rate of 0.6in/sec. An elastic is anchored 200 inches from the base and attached to the top of the pole, forming a right triangle. Using a top-down or bottom-up strategy, define functions that compute the *height* of the pole and the *area* of the triangle after a given number of seconds.




**Directions:** Define your first function (`height` or `area`) here.

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

---

**Directions:** Define your second function (`height` or `area`) here.

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

**Definition**

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

(define ( _____ )
function name variable(s)

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

# Word Problem: collision?

**Directions:** Use the Design Recipe to write a function `collision?`, 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 makes use of the `distance` function to check if they are close enough to collide.

**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…

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name | input(s) | what the function produces

**Definition**

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

(define ( _____ )
function name | variable(s)

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

# Design Recipe

**Directions:**

## Contract and Purpose Statement

Every contract has three parts…

; _____ : _____ -> _____
function name Domain Range

; _____
what does the function do?

## Examples

Write some examples, then circle and label what changes…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

## Definition

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

(define ( _____ )
function name variable(s)

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

---

**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…

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

*(EXAMPLE (* _____ *)* _____ )
function name input(s) what the function produces

## Definition

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

(define ( _____ )
function name variable(s)

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

# Contracts for Algebra (Wescheme)

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 |
|---|---|---|
| `; *` | `::` `( Number a , Number b )` | `-> Number` |
| `(* 1 2)` | | |
| `; +` | `::` `( Number a , Number b )` | `-> Number` |
| `(+ 1 2)` | | |
| `; -` | `::` `( Number a , Number b )` | `-> Number` |
| `(- 1 2)` | | |
| `; /` | `::` `( Number a , Number b )` | `-> Number` |
| `(/ 1 2)` | | |
| `; <` | `::` `( Number a , Number b )` | `-> Boolean` |
| `(< 3 4) ; produces true` | | |
| `; <=` | `::` `( Number a , Number b )` | `-> Boolean` |
| `(<= 3 3) ; produces true, because 3 is equal to 3` | | |
| `; =` | `::` `( Number a , Number b )` | `-> Boolean` |
| `(= 3 4) ; produces false` | | |
| `; >` | `::` `( Number a , Number b )` | `-> Boolean` |
| `(> "a" "b") ; produces false` | | |
| `; >=` | `::` `( Number a , Number b )` | `-> Boolean` |
| `(>= 3 4) ; produces false, because 3 is neither greater-than nor equal-to 4` | | |
| `; above` | `::` `( Image above , Image below )` | `-> Image` |
| `(above (circle 10 "solid" "black") (square 50 "solid" "red"))` | | |
| `; and` | `::` `( Boolean condition1 , Boolean condition2 )` | `-> Boolean` |
| `(and (> 0 1)  (= 4 4)) ; produces false because both conditions must be true` | | |
| `; beside` | `::` `( Image left , Image right )` | `-> Image` |
| `(beside (circle 10 "solid" "black") (square 50 "solid" "red"))` | | |

| Name | Domain | Range |
|---|---|---|
| ; circle | :: ( Number , String, , String ) | -> Image |
| | (circle 50 "solid" "purple") | |
| ; ellipse | :: ( Number , Number , String , String ) | -> Image |
| | (ellipse 100 50 "outline" "orange") | |
| ; expt | :: ( Number , Number ) | -> 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")) | |
| ; image-url | :: ( String ) | -> Image |
| | (image-url "https://bootstrapworld.org/images/icon.png") | |
| ; isosceles-triangle | :: ( Number , Number , String , String ) | -> Image |
| | (isosceles-triangle 50 20 "solid" "grey") | |
| ; or | :: ( Boolean , Boolean ) | -> Boolean |
| | (or (> 1 0) (= 4 4)) ; produces true if one or more conditions are be true | |
| ; overlay | :: ( Image , Image ) | -> Image |
| | (overlay (circle 10 "solid" "black") (square 50 "solid" "red")) | |
| ; radial-star | :: ( Num , Num , Num , Str , Str ) | -> Image |
| | (radial-star 6 20 50 "solid" "red") | |
| ; rectangle | :: ( Number , Number , String , String ) | -> Image |
| | (rectangle 100 50 "outline" "green") | |
| ; regular-polygon | :: ( Number , Number , String , String ) | -> Image |
| | (regular-polygon 25 5 "solid" "purple") | |
| ; rhombus | :: ( Number , Number , String , String ) | -> Image |
| | (rhombus 100 45 "outline" "pink") | |
| ; right-triangle | :: ( Number , Number , String , String ) | -> Image |
| | (right-triangle 50 60 "outline" "blue") | |
| ; rotate | :: ( Number , Image ) | -> Image |
| | (rotate 45 (star 50 "solid" "darkblue")) | |
| ; scale | :: ( Number , Image ) | -> Image |
| | (scale 1/2 (star 50 "solid" "lightblue")) | |

| Name | Domain | Range |
| --- | --- | --- |
| ; 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 |
| *undefined* | | |
| ; 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")* | | |
| ; 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" "darkgreen")* | | |

| | | |
| --- | --- | --- |
| | :: | -> |

| | | |
| --- | --- | --- |
| | :: | -> |

| | | |
| --- | --- | --- |
| | :: | -> |


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.