

Name: \_\_\_\_\_



# **BOOTSTRAP:REACTIVE**

---

[www.bootstrapworld.org](http://www.bootstrapworld.org)

Class: \_\_\_\_\_



## Workbook v0.9

Brought to you by the Bootstrap team:

- Emma Youndtsmith
- Emmanuel Schanzer
- Kathi Fisler
- Joe Politz
- Shriram Krishnamurthi

Visual Design: Colleen Murphy

---

Bootstrap is licensed under a Creative Commons 3.0 Unported License. Based on a work from [www.BootstrapWorld.org](http://www.BootstrapWorld.org). Permissions beyond the scope of this license may be available at [schanzer@BootstrapWorld](mailto:schanzer@BootstrapWorld)

# Unit 1

	Racket Code	Pyret Code
<i>Numbers</i>	<pre>(define AGE 14) (define A-NUMBER 0.6) (define SPEED -90)</pre>	<p>AGE = 14  A-NUMBER = 0.6  SPEED = -90</p> <p>Two of your own:</p> <hr/> <p><b><u>WIDTH = 640</u></b></p> <hr/> <p><b><u>YEAR = 2017</u></b></p>
<i>Strings</i>	<pre>(define CLASS "Bootstrap") (define PHRASE "Coding is fun!") (define A-STRING "2500")</pre>	<p>CLASS = "Bootstrap"  PHRASE = "Coding is fun!"  A-STRING = "2500"</p> <p>Two of your own:</p> <hr/> <p><b><u>NAME = "Elizabeth"</u></b></p> <hr/> <p><b><u>CITY = "Philadelphia"</u></b></p>

<i>Images</i>	<pre>(define SHAPE   (triangle 40 "outline" "red"))  (define OUTLINE   (star 80 "solid" "green"))  (define SQUARE   (rectangle 50 50 "solid" "blue"))</pre>	<p>SHAPE =  <code>triangle(40, "outline", "red")</code></p> <p>OUTLINE =  <code>star(80, "solid", "green")</code></p> <p>SQUARE =  <code>rectangle(50, 50, "solid", "blue")</code></p> <p>One of your own:</p> <hr/> <p><b>MY-SHAPE =</b>  <code><u>rhombus(90, 60, "solid", "red")</u></code></p> <hr/>
<i>Booleans</i>	<pre>(define BOOL true)  (define BOOL2 false)</pre>	<p>BOOL = <code>true</code></p> <p>One of your own:</p> <hr/> <p><b>BOOL2 = false</b></p> <hr/>
<i>Functions</i>	<pre>; double : Number -&gt; Number ; Given a number, multiply by ; 2 to double it  (EXAMPLE (double 5) (* 2 5)) (EXAMPLE (double 7) (* 2 7))  (define (double n) (* 2 n))</pre>	<p># double : Number -&gt; Number  # Given a number, multiply by  # 2 to double it</p> <p>examples:  <code>double(5) is 2 * 5</code>  <code>double(7) is 2 * 7</code></p> <p>end</p> <p>fun double(n):  <code>2 * n</code></p> <p>end</p>

# Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

# double: Number → Number  
name domain range

## examples:

double (5) is 2 \* 5

double (7) is  $2^7$

end

```
fun double(n) :
```

$$2 * n$$

---

end

# triple: Number -> Number  
name domain range

## examples:

triple (  ) is  \* 3

triple  $(\text{ } \text{ } 40 \text{ } \text{ })$  is  $40^* 3$

end

```
fun triple(n):
```

---

n \* 3

end

# Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

# plus1 : Number -> Number  
name domain range

examples:

plus1 (n) is  $n + 1$   
plus1 (n) is  $n + 1$

end

fun plus1 (n):

n + 1

end

# mystery : Number -> Number  
name domain range

examples:

mystery (n) is  $n - 4$   
mystery (n) is  $n - 4$

end

fun mystery (n):

n - 4

end

# Fast Functions!

Fill out the contract for each function, then try to write two examples and the definition by yourself.

# red-spot : Number -> Image  
name domain range

examples:

red-spot(20) is circle(20, "solid", "red")  
red-spot(99) is circle(99, "solid", "red")

end

fun red-spot(radius):

circle(radius, "solid", "red")

end

#                    :                    ->                     
name domain range

examples:

                  (      ) is                     
                  (      ) is                   

end

fun                   (      ):

end

# Syntax and Style Bug Hunting: Pyret Edition

#1	<pre>SECONDS = (7)  STRING = my string</pre>	<p><b>SECONDS = 7</b></p> <p><b>STRING = “my string”</b></p>
#2	<pre>SHAPE1 = circle(50 “solid” “blue”)  SHAPE2 = triangle(75, outline, yellow)</pre>	<p><b>SHAPE1 = circle(50, “solid”, “blue”)</b></p> <p><b>SHAPE2 = triangle(75, “outline”, “yellow”)</b></p>
#3	<pre># triple : Number -&gt; Number # Multiply a given number by # 3 to triple it  examples:     triple(5) = 3 * 5     triple(7) = 3 * 7 end</pre>	<p><b># triple : Number -&gt; Number</b></p> <p><b># Multiply a given number by 3 to triple it</b></p> <p><b>examples:</b></p> <p><b>triple(5) is 3 * 5</b></p> <p><b>triple(7) is 3 * 7</b></p> <p><b>end</b></p>
#4	<pre>fun triple(n):     3 * n</pre>	<p><b>fun triple(n) :</b></p> <p><b>    3 * n</b></p> <p><b>end</b></p>
#5	<pre># ys : Number -&gt; Number # Given a number, create a solid # yellow star of the given size  examples:     ys(99) is star(99, “solid”, “yellow”)     ys(33) is star(99, “solid”, “yellow”)  ys(size):     star(size “solid” “yellow”) end</pre>	<p><b># ys : Number -&gt; Number</b></p> <p><b># Given a number, create a solid yellow star of the given size</b></p> <p><b>examples:</b></p> <p><b>ys(99) is star(99, “solid”, “yellow”)</b></p> <p><b>ys(99) is star(99, “solid”, “yellow”)</b></p> <p><b>end</b></p> <p><b>ys(size) :</b></p> <p><b>    star(size, “solid”, “yellow”)</b></p> <p><b>end</b></p>

# Unit 2

## Word Problem: double-radius

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

## Contract+Purpose Statement

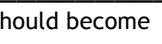
Every contract has three parts:

#	<u>double-radius</u> : <u>Number, String</u>	-> <u>Image</u>
	name	Domain
#	Consumes a number and a string, produces an outlined circle of the given color, whose radius is twice the given number	Range

## Give Examples

Write examples of your function in action

**examples:**

double-radius (   ) is  
the user types...  
   
circle(50 \* 2, "outline", "pink")  
...which should become  
   
double-radius (   ) is  
the user types...  
   
circle(918 \* 2, "outline", "orange")  
which should become

end

## Function

Circle the changes in the examples, and name the variables

Write the code, copying everything that isn't circled, and using names where you find variables!

```
fun double-radius ( radius, color ):
```

circle(radius \* 2, “outline”, color)

end

## Word Problem: double-width

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

## Contract+Purpose Statement

Every contract has three parts:

#	<u>double-width</u>	:	Number	->	Image
#	name		Domain		Range
Consumes a length and produces a solid green rectangle whose width is twice					
#	the given length				

## Give Examples

Write examples of your function in action

## examples:

double-width (  ) is  
the user types...

```
rectangle(45, 45)* 2, "solid", "green")
```

...which should become

double-width ( 8 ) is

the user types...

**length**

```
rectangle(8, 8 * 2, "solid", "green")
```

...which should become

end

## Function

Circle the changes in the examples, and name the variables.

Write the code, copying everything that isn't circled, and using names where you find variables!

fun double-width(length) :

```
rectangle(length, length * 2, "solid", "green")
```

end

# Word Problem: next-position

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

## Contract+Purpose Statement

Every contract has three parts:

# next-position : Number, Number → DeliveryState

name

Domain

Range

#Consumes x and y coordinates and produces a DeliveryState by adding 5 to x  
and subtracting 5 from y      What does the function do?

## Give Examples

Write examples of your function in action

examples:

next-position ( 30, 250 )      is

the user types...

delivery(30 + 5, 250 - 5)

...which should become

next-position ( 65, 800 )      is

the user types...

delivery(65 - 5, 800)

...which should become

end

## Function

Circle the changes in the examples, and name the variables.

Write the code, copying everything that isn't circled, and using names where you find variables!

fun next-position ( x , y ) :

delivery(x + 5, y - 5)

end

# Data Structure

```
# a CakeType is a flavor, layers, & is-iceCream
data CakeType:
    | cake(____ flavor :: String,
           _____ layers :: Number,
           _____ is-iceCream :: Boolean)
end
```

To make examples of this structure, I would write:

cake1 = \_\_\_\_\_

cake2 = cake("Red Velvet", 2, true)

To access the fields of **cake2**, I would write:

\_\_\_\_\_ cake2.flavor

\_\_\_\_\_ cake2.layers

\_\_\_\_\_ cake2.is-iceCream

## Word Problem: taller-than

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

Contract+Purpose Statement

# taller-than : CakeType, CakeType → Boolean

# Consumes 2 CakeTypes, produces true if the first CakeType has more layers than the second

Give Examples

Write examples of your function in action

examples :  
taller-than ( a-cake1 birthday-cake, a-cake2 chocolate-cake ) is

the user types...  
a-cake1 birthday-cake.layers > a-cake2 chocolate-cake.layers

...which should become  
a-cake1 taller-than ( strawberry-cake, a-cake2 pb-cake ) is

the user types...  
a-cake1 strawberry-cake.layers > a-cake2 pb-cake.layers

end

Function

Circle the changes in the examples, and name the variables.

Write the code, copying everything that isn't circled, and using names where you find variables!

fun taller-than ( a-cake1, a-cake2 ) :

a-cake1.layers > a-cake2.layers

end

## Word Problem: will-melt

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

### Contract+Purpose Statement

```
# will-melt : CakeType, Number -> Boolean
# Consumes a CakeType and a temperature, produces true if the temp is greater than 32
# degrees, AND the CakeType is an ice cream cake
```

### Give Examples

examples:

will-melt (cake3, 75) is  
cake3.is-iceCream and (75 > 32)

will-melt (cake4, 10) is  
cake4.is-iceCream and (10 > 32)

end

## Function

```
fun will-melt (a-cake, temp) :
```

a-cake.is-iceCream and (temp > 32)

end

## Vocabulary Practice

Below is a new structure definition:

**data** MediaType:

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

**end**

# an example book:

```
book1 = book("1984", "Orwell", 1949)
```

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

- |               |              |
|---------------|--------------|
| - contract    | - example    |
| - header      | - field      |
| - datatype    | - instance   |
| - constructor | - data block |
| - name        | - purpose    |

**author** is a \_\_\_\_\_ field

**book** is a \_\_\_\_\_ constructor

**MediaType** is a \_\_\_\_\_ datatype

**book1** is a \_\_\_\_\_ instance

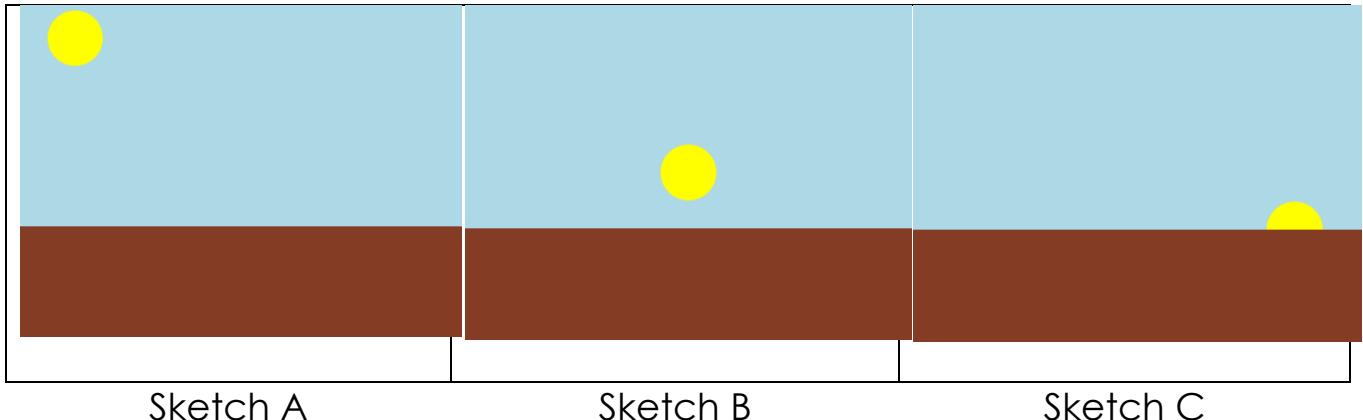
**title** is a \_\_\_\_\_ field

**data ... end** is a \_\_\_\_\_ data block

## Unit 3

# Identifying Animation Data Worksheet: Sunset

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes
Sun's x-coordinate	Location increases – moves to the right
Sun's y-coordinate	Location decreases – moves downward

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

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

(worksheet continues on the next page)

Define the Data Structure

```
# a SunsetState is the x and y location of a sunset  
  
data SunsetState:  
  
|   sunset(__ xpos :: Number,  
|           __ ypos :: Number  
| )  
  
end
```

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

**sunsetA** = sunset(50, 270)

**sunsetB** = sunset(200, 150)

**sunsetC** = sunset(340, 100)

## Word Problem: draw-state

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

Contract+Purpose Statement

# `draw-state` : `SunsetState` → `Image`

# Consumes a `SunsetState`, and produces an image of a sunset

Write an expression for each piece of your final image

<code>SUN =</code>	<code>circle(25, "solid", "yellow")</code>
<code>GROUND =</code>	<code>rectangle(400, 100, "solid" "brown")</code>
<code>SKY =</code>	<code>rectangle(400, 300, "solid", "light-blue")</code>

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

`fun` `draw-state` (`a-sunset`) :

`put-image(GROUND, 200, 50,`

`put-image(SUN, a-sunset.xpos, a-sunset.ypos,`

`SKY)))`

`end`

## Word Problem: next-state-tick

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

Contract+Purpose Statement

# next-state-tick : SunsetState → SunsetState

# Consumes a *SunsetState* and produces the next *SunsetState* by moving the sun 8 pixels to the right and 4 pixels down

Give Examples

Write examples of your function in action

examples :

next-state-tick ( sunset(50, 330) ) is

the user types...

sunset(50 + 8, 330 - 4)

...which should become

next-state-tick ( sunset(240, 120) ) is

the user types...

sunset(240 + 8, 120 - 4)

end

...which should become

Function

Circle the changes in the examples, and name the variables.

Write the code, copying everything that isn't circled, and using names where you find variables!

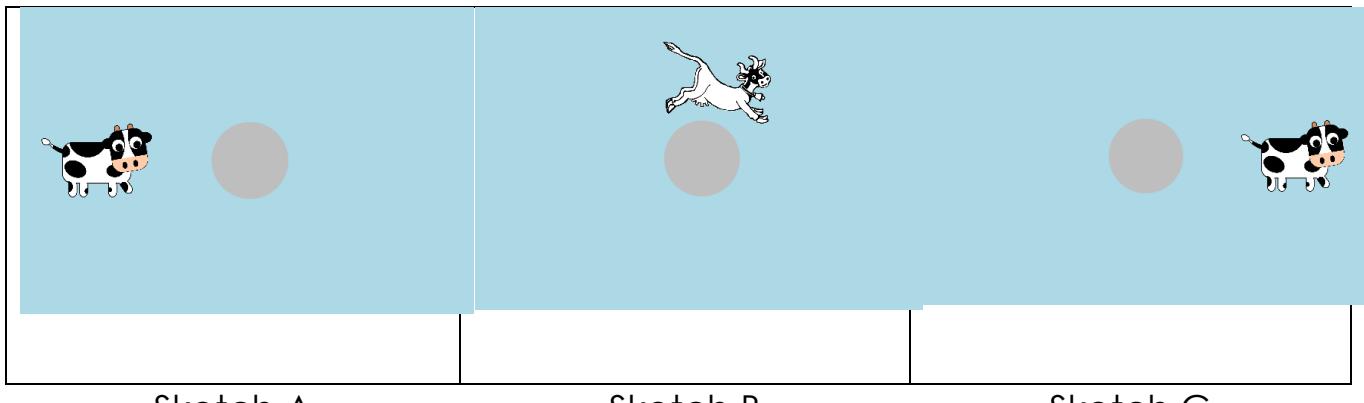
fun next-state-tick ( a-sunset ) :

sunset(a-sunset.x + 8, a-sunset.y - 4)

end

# Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes
Cow's x-coordinate	Location increases – moves to the right
Cow's y-coordinate	Stays consistent until cow reaches the moon, then increases, decreases when cow has finished jumping
Cow image	Switches from normal to jumping cow image when cow is jumping over the moon

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

Field name (dangerX, score, playerIMG...)	Datatype (Number, String, Image, Boolean...)
xpos	Number
ypos	Number
cow-img	Image

(worksheet continues on the next page)

Define the Data Structure

```
# a CowState is the x and y-position of the cow,
and the cow's image

data CowState:
| cow( xpos :: Number,
|           ypos :: Number,
|           cow-img :: Image )
end
```

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

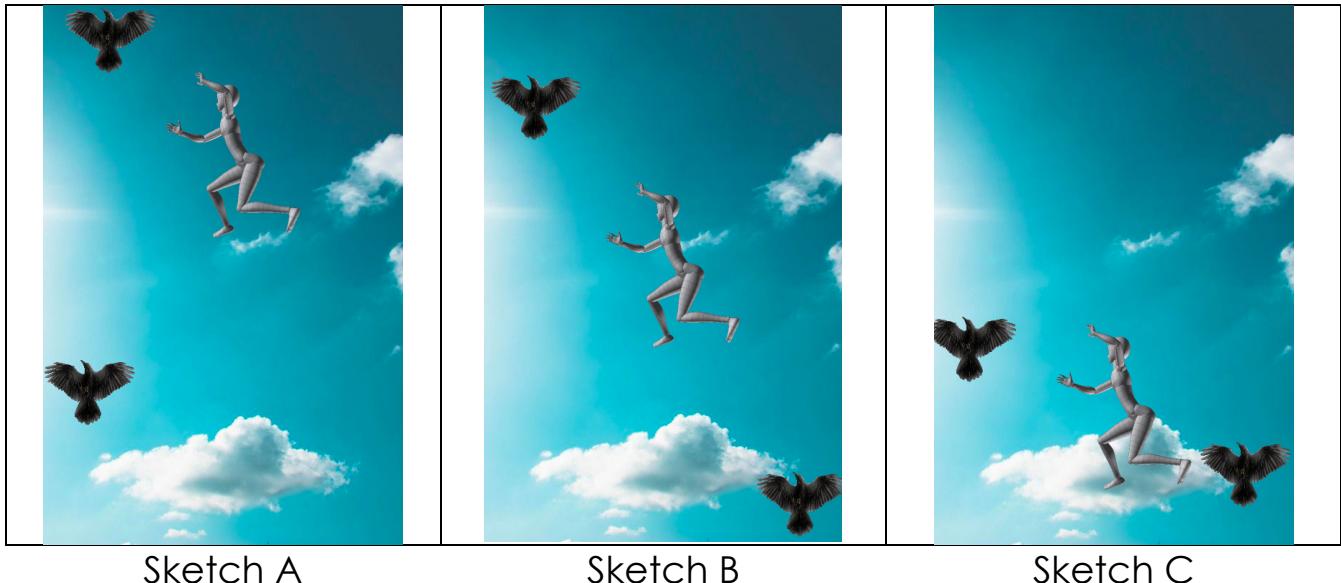
**CowA** = cow(100, 200, NORMAL-COW)

**CowB** = cow(300, 300, FLYING-COW)

**CowC** = cow(400, 200, NORMAL-COW)

# Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes
Person's y-coordinate	Decreases consistently
Bird1's x coordinate	Increases consistently when onscreen, resets at random when it resets
Bird1's y-coordinate	Increases consistently
Bird2's x-coordinate	Increases consistently when onscreen, resets at random when it resets
Bird2's y-coordinate	Increases consistently

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

Field name (dangerX, score, playerIMG...)	Datatype (Number, String, Image, Boolean...)
personY	Number
bird1x	Number
bird1y	Number
bird2x	Number
bird2y	Number

(worksheet continues on the next page)

Define the Data Structure

# a **FallingState** is a person's x-coordinate, and  
the x and y coordinates of 2 flying birds

data **FallingState**:

```
| fall(____ persony :: Number,  
|         ____ bird1x :: Number,  
|             ____ bird1y :: Number,  
|                 ____ bird2x :: Number,  
|                     ____ bird2y :: Number )
```

end

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

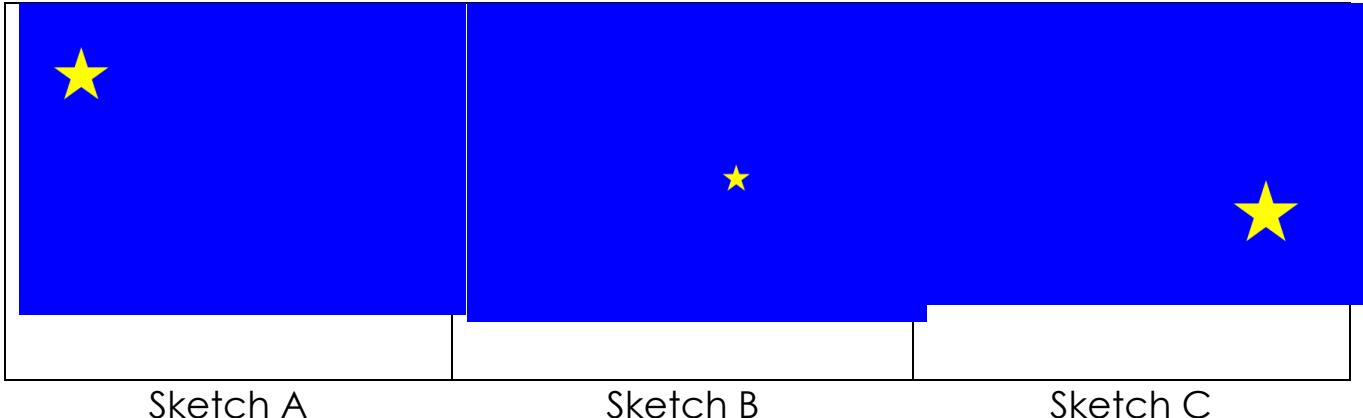
**FallingA** = fall(600, 100, 700, 75, 250)

**FallingB** = fall(450, 100, 650, 330, 100)

**FallingC** = fall(300, 440, 97, 60, 270)

# Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes
Star's x-coordinate	Increases consistently
Star's y-coordinate	Decreases consistently
Star's size	Increases until a set size, then decreases
Star's growth rate	Either negative or positive, based on size of the star

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

Field name (dangerX, score, playerIMG...)	Datatype (Number, String, Image, Boolean...)
xpos	Number
ypos	Number
star-size	Number
growth-rate	Number

(worksheet continues on the next page)

Define the Data Structure

```
# a StarState is the x and y-coordinate of the  
star, its size, and growth rate  
  
data StarState:  
    | pstar(_____  
    |     xpos :: Number,  
    |     ypos :: Number,  
    |     star-size :: Number,  
    |     growth-rate :: Number  
end
```

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

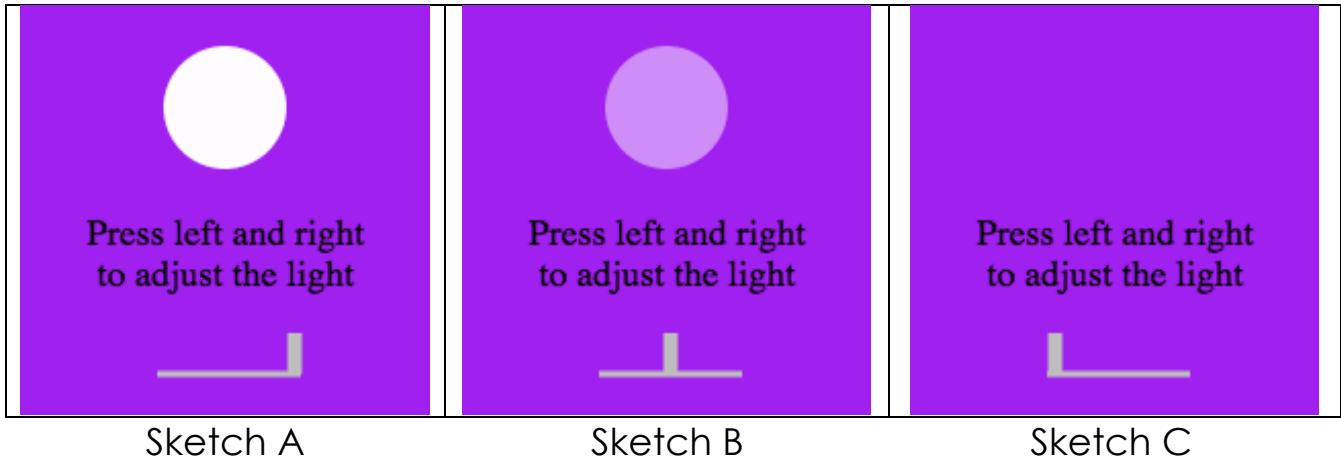
PulsingStarA = pstar(100, 400, 60, 2)

PulsingStarB = pstar(450, 200, 30, -2)

PulsingStarC = pstar(600, 130, 60, 2)

# Identifying Animation Data Worksheet

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes
Dimmer switch x-coordinate	Increases and decreases when arrow keys are pressed
Light brightness/circle opacity	Increases and decreases based on dimmer switch's x-coordinate

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

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

(worksheet continues on the next page)

Define the Data Structure

```
# a DimmerState is the x-coordinate of a light  
dimmer switch
```

```
data DimmerState:
```

```
| dimmer( _____ xpos :: Number  
| _____  
| _____ )
```

```
end
```

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

**LightDimmerA** = **dimmer( 60 )**

**LightDimmerB** = **dimmer( 30 )**

**LightDimmerC** = **dimmer( 0 )**

## Unit 4

## Word Problem: location

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

Contract+Purpose Statement

# location : DeliveryState → String  
# Consumes a *DeliveryState* and produces the location of the box:  
either “road”, “delivery zone” “house”, or “air”

Give Examples

examples :

location (delivery(100, 400)) is “air”  
location (delivery(320, 75)) is “delivery zone”  
location (delivery(50, 75)) is “road”  
location (delivery(600, 240)) is “house”

end

(worksheet continues next page)

## Function

```
fun location(a-delivery) :  
    if (a-delivery.x <= 300) and (a-delivery.y <= 100) :  
        “road”  
    else if (a-delivery.x > 300) and (a-delivery.y <= 100) :  
        “delivery zone”  
    else if (a-delivery.x >= 450) and (a-delivery.y <= 300) :  
        “house”  
    else: “air”  
  
end  
end
```

# Syntax and Style Bug-Hunting: Piecewise Edition

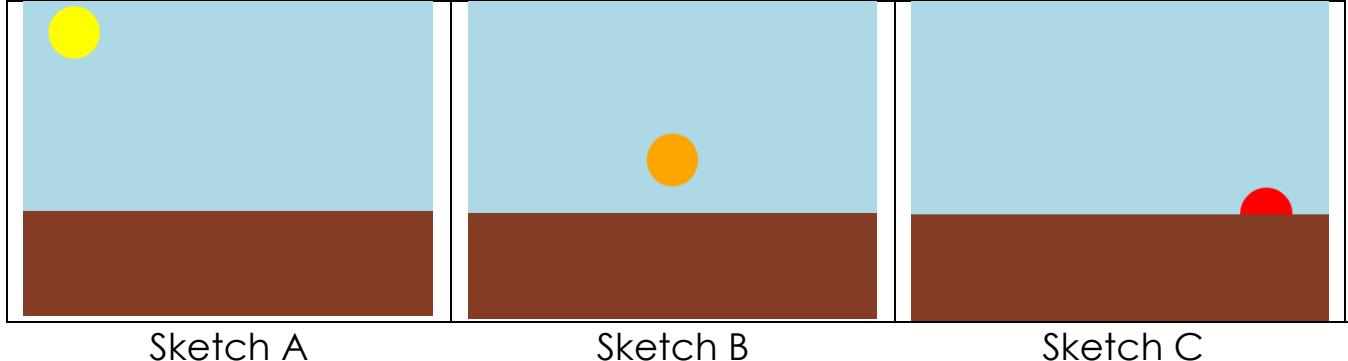
	Buggy Code	Correct Code / Explanation
Round 1	<pre>fun piecewisefun(n):     if (n &gt; 0): n     else: 0</pre>	<p>No 'end' statements</p> <pre>fun piecewisefun(n):     if (n &gt; 0): n     else: 0     end end</pre>
Round 2	<pre>fun cost(topping):     if string-equal(topping, "pepperoni"): 10.50     else string-equal(topping, "cheese"): 9.00     else string-equal(topping, "chicken"): 11.25     else string-equal(topping, "broccoli"): 10.25     else: "That's not on the menu!"     end end</pre>	<p>Need to use 'else if' for all but the final condition</p> <pre>fun cost(topping):     if string-equal(topping, "pepperoni"): 10.50     else if string-equal(topping, "cheese"): 9.00     else if string-equal(topping, "chicken"): 11.25     else if string-equal(topping, "broccoli"): 10.25     else: "That's not on the menu!"     end end</pre>
Round 3	<pre>fun absolute-value(a b):     if a &gt; b: a - b     b - a     end end</pre>	<p>No 'else' statement before final condition</p> <pre>fun absolute-value(a b):     if a &gt; b: a - b     else: b - a     end end</pre>
Round 4	<pre>fun best-function(f):     if string-equal(f, "blue"):         "you win!"     else if string-equal(f, "blue"):         "you lose!"     else if string-equal(f, "red"):         "Try again!"     else: "Invalid entry!"     end end</pre>	<p>First and second condition are the same    (This program will run, but the second condition will never be evaluated)</p> <pre>if string-equal(f, "blue"): "you win!" else if string-equal(f, "green"): "you lose!"</pre>

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

Make the sun change colors- yellow at the top of the screen, orange in the middle, and red at the bottom (the horizon)

Draw a sketch for three distinct moments of the animation, focusing on the new behavior



Sketch A

Sketch B

Sketch C

What NEW things are changing? Are they independent of existing fields?

Thing	Describe how it changes
Sun's color	Changes from yellow to orange to red, based on sun's y-coordinate

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

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

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

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

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

**top-left** = sunset(50, 270)

**middle** = sunset(200, 150)

**bottom-right** = sunset(340, 100)

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

---

---

---

---

---

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

---

---

---

---

---

## Word Problem: draw-sun

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

Contract+Purpose Statement

# draw-sun : SunsetState → Image  
# Consumes a `SunsetState` and produces a circle whose color ("yellow", "orange", or "red") is based on the sunset's y coordinate

Give Examples

examples :

draw-sun (sunset(50, 270)) is circle(25, "solid", "yellow")  
draw-sun (sunset(200, 150)) is circle(25, "solid", "orange")  
draw-sun (sunset(340, 100)) is circle(25, "solid", "red")  
draw-sun (sunset(400, 0)) is circle(25, "solid", "red")

end

(worksheet continues next page)

## Function

```
fun draw-sun (a-sunset) :  
if a-sunset.y >= 225 :  
    circle(25, "solid", "yellow")  
else if a-sunset.y >= 150 :  
    circle(25, "solid", "orange")  
else: circle(25, "solid", "red")  
  
end  
end
```

## Unit 5

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

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

Draw a sketch for three distinct moments of the animation, focusing on the new behavior



Sketch A

Sketch B

Sketch C

What NEW things are changing? Are they independent of existing fields?

Thing	Describe how it changes
Hunger level	Decreases by 2 each tick
Sleep level	Decreases by 1 each tick

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

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

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

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

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

**FULLPET** = pet(100, 100)

**MIDPET** = pet(50, 75)

**LOSEPET** = pet(0, 0)

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

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

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

next-state-tick(LOSEPET) is LOSEPET

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

---

---

---

---

---

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

On a keypress, if the user pressed "f" (for "feed"), hunger should increase by 10. If the user pressed "s" (for "sleep"), sleep should increase by 5. If the user presses any other keys, nothing should change.

Draw a sketch for three distinct moments of the animation, focusing on the new behavior



Sketch A



Sketch B



Sketch C

What NEW things are changing? Are they independent of existing fields?

Thing	Describe how it changes
Hunger	Increases by 10 if 'f' key is pressed
Sleep	Increases by 5 if 's' key is pressed

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

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

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

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

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

**FULLPET** = pet(100, 100)

**MIDPET** = pet(50, 75)

**LOSEPET** = pet(0, 0)

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

next-state-key(FULLPET, “f”) is FULLPET

next-state-key(FULLPET, “s”) is FULLPET

next-state-key(MIDPET, “f”) is pet(50 + 10, 75)

next-state-key(MIDPET, “s”) is pet(50, 75 + 5)

next-state-key(LOSEPET, “s”) is LOSEPET

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

---

---

---

---

---

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

**When either of the pet's hunger or sleep levels reaches 0, the game is lost and the pet is sad- the happy pet image is replaced with a sad pet image**

Draw a sketch for three distinct moments of the animation, focusing on the new behavior



Sketch A

Sketch B

Sketch C

What NEW things are changing? Are they independent of existing fields?

Thing	Describe how it changes
Cat's image	Changes from happy to sad cat image when either of the cat's levels reaches 0

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

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

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

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

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

**FULLPET** = pet(100, 100)

**MIDPET** = pet(50, 75)

**LOSEPET** = pet(0, 0)

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

draw-state(LOSEPET) is put-image(...

[put hunger and sleep bars at current position on screen] ...

put-image(SADCAT, 300, 200, BACKGROUND))))

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

# has-lost : PetState -> Boolean

# helper function which returns true when hunger or sleep reaches 0

examples:

has-lost(FULLPET) is false

has-lost(LOSEPET) is true

end

# Build Your Own Animation

# Animation Design Worksheet

Draw a sketch for three distinct moments of the animation

--	--	--

Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

Define the Data Structure

# a \_\_\_\_\_ **State** is \_\_\_\_\_

data \_\_\_\_\_ **State**:

| \_\_\_\_\_ ( \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_ )

end

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

\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_

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

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

# Collision

# Distance:

The Player is at (4, 2) and the Target is at (0, 5).

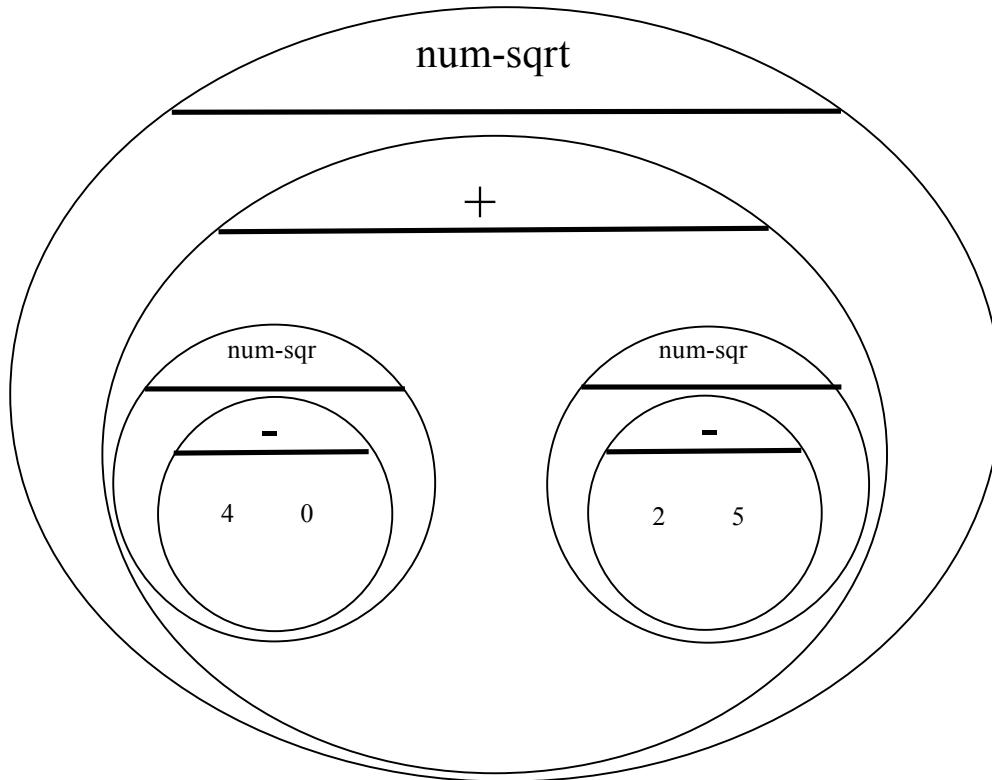
Distance takes in the player's x, player's y, character's x and character's y.

Use the formula below to fill in the EXAMPLE:

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

---

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



---

Convert it into Pyret code:

`num-sqrt(num-sqr(4 - 0) + num-sqr(2 - 5))`

# Word Problem: distance

Write a function `distance`, which takes FOUR inputs:

- `px`: The x-coordinate of the player
- `py`: The y-coordinate of the player
- `cx`: The x-coordinate of another game character
- `cy`: The y-coordinate of another game character

It should return the distance between the two, using the Distance formula:

$$\text{Distance}^2 = (px - cx)^2 + (py - cy)^2$$

## Contract+Purpose Statement

# distance : Number, Number, Number, Number -> Number  
Consumes the coordinates of 2 characters: px, py, cx, and cy, produces  
# the distance between them using the distance formula

## Give Examples

Write examples of your function in action

examples:      
distance ( 4, 2, 0, 5 )    is

$$\text{num-sqrt}(\text{num-sqr}(4 - 0) + \text{num-sqr}(2 - 5))$$

  
distance ( 80, 33, 6, 50 )    is

$$\text{num-sqrt}(\text{num-sqr}(80 - 6) + \text{num-sqr}(33 - 50))$$

end

## Function

fun distance ( px, py, cx, cy ) :

$$\text{num-sqrt}(\text{num-sqr}(px - cx) + \text{num-sqr}(py - cy))$$

end

# Word Problem: is-collision

Write a function `is-collision`, which takes FOUR inputs:

- ❑ px: The x-coordinate of the player
- ❑ py: The y-coordinate of the player
- ❑ cx: The x-coordinate of another game character
- ❑ cy: The y-coordinate of another game character

It should return true if the coordinates of the player are within **50 pixels** of the coordinates of the other character. Otherwise, false.

## Contract+Purpose Statement

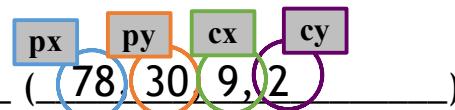
```
# is-collision : Number, Number, Number, Number -> Boolean
    Consumes the coordinates of 2 characters: px, py, cx, and cy, produces
# true if the distance between them is less than 50 pixels
```

## Give Examples

Write examples of your function in action

examples:

is-collision



is

distance(78, 30, 9, 2) < 50

is-collision



is

distance(20, 40, 60, 80) < 50

end

## Function

```
fun is-collision (px, py, cx, cy) :
```

```
distance(px, py, cx, cy) < 50
```

end

# DESIGN RECIPE

## Contract+Purpose Statement

Every contract has three parts:

# \_\_\_\_\_ : \_\_\_\_\_ -> \_\_\_\_\_  
name Domain Range

# \_\_\_\_\_  
What does the function do?

## Give Examples

Write examples of your function in action

examples:

\_\_\_\_\_ (\_\_\_\_\_ ) is  
the user types...

...which should become

\_\_\_\_\_ (\_\_\_\_\_ ) is  
the user types...

...which should become

end

## Function

Circle the changes in the examples, and name the variables.

fun \_\_\_\_\_ (\_\_\_\_\_ ) :

end

# DESIGN RECIPE

## Contract+Purpose Statement

Every contract has three parts:

# \_\_\_\_\_ : \_\_\_\_\_ -> \_\_\_\_\_  
name Domain Range

# \_\_\_\_\_  
What does the function do?

## Give Examples

Write examples of your function in action

examples:

\_\_\_\_\_ (\_\_\_\_\_ ) is  
the user types...

...which should become

\_\_\_\_\_ (\_\_\_\_\_ ) is  
the user types...

...which should become

end

## Function

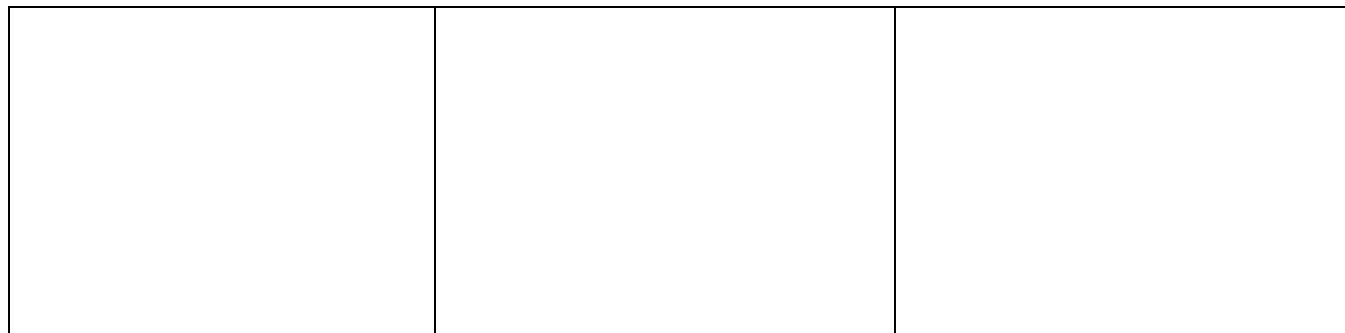
Circle the changes in the examples, and name the variables.

fun \_\_\_\_\_ (\_\_\_\_\_ ) :

end

# Animation Design Worksheet

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

Define the Data Structure

# a \_\_\_\_\_ **State** is \_\_\_\_\_

data \_\_\_\_\_ **State**:

| \_\_\_\_\_ ( \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_ )

end

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

\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_

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

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

# Animation Design Worksheet

Draw a sketch for three distinct moments of the animation

--	--	--

Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

Define the Data Structure

# a \_\_\_\_\_ **State** is \_\_\_\_\_

data \_\_\_\_\_ **State**:

| \_\_\_\_\_ ( \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_ )

end

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

\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_

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

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

# Animation Design Worksheet

Draw a sketch for three distinct moments of the animation

--	--	--

Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

Define the Data Structure

# a \_\_\_\_\_ **State** is \_\_\_\_\_

data \_\_\_\_\_ **State**:

| \_\_\_\_\_ ( \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_  
| \_\_\_\_\_ )

end

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

\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_  
\_\_\_\_\_ = \_\_\_\_\_

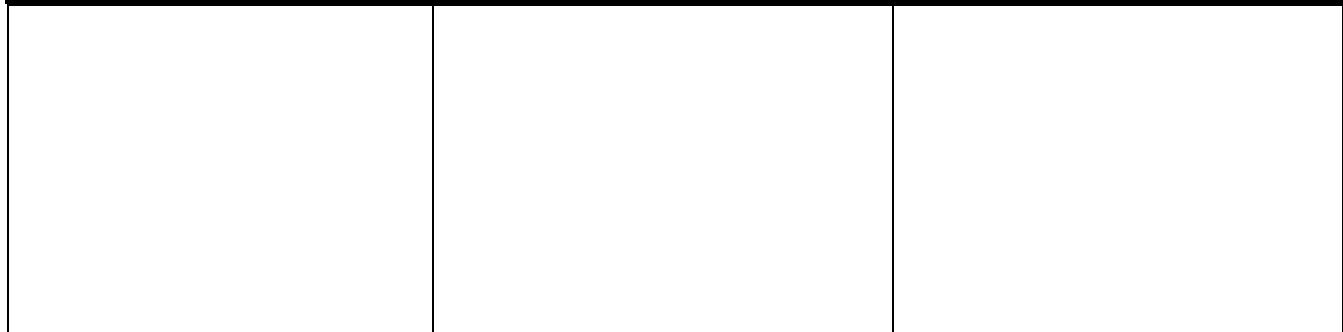
Write an example for one of the functions on the previous page:

\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_  
\_\_\_\_\_

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

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

\_\_\_\_\_ = \_\_\_\_\_

\_\_\_\_\_ = \_\_\_\_\_

\_\_\_\_\_ = \_\_\_\_\_

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

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

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

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

# Animation Extension Worksheet

Describe the goal of your change: what new feature or behavior will it add to your animation?

Draw a sketch for three distinct moments of the animation



Sketch A

Sketch B

Sketch C

What things are changing?

Thing	Describe how it changes

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

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

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

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

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

\_\_\_\_\_ = \_\_\_\_\_

\_\_\_\_\_ = \_\_\_\_\_

\_\_\_\_\_ = \_\_\_\_\_

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

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

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

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

\_\_\_\_\_

## Contracts

## Contracts