Name: ____

Student Workbook

BOOTSTRAP
Equity • Scale • Rigor

Workbook v3.0

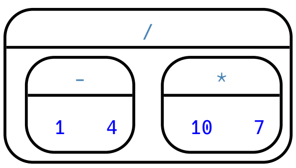
Brought to you by the Bootstrap team:

- Emmanuel Schanzer
- Kathi Fisler
- Shriram Krishnamurthi
- Dorai Sitaram
- Joe Politz
- Jennifer Poole
- Ed Campos
- Ben Lerner
- Flannery Denny

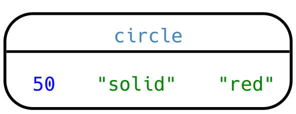
Visual Designer: Colleen Murphy

Starting to Program: Order of Operations & Contracts

- 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 read and load everything in the Definitions Area and erase anything that was typed into the Interactions Area.
- Our programming language has many types of **values**:
  - **Numbers** can be integers like `42`, decimals like `0.5`, or even fractions like `1/3`. Clicking on a fraction or a decimal will cause it to switch from one to the other.
  - **Strings** are anything in quotes, such as `"Programming is fun!"`. A Number written in quotes is *still a String!*
- Our language also has **functions** you've seen before, such as addition (`+`), subtraction (`-`), multiplication (`*`) and division (`/`).
  - **Order of Operations** is incredibly important when programming. To help us organize our math into something we can trust, we can *diagram* a math expression using the **Circles of Evaluation**. For example, the expression $(1-4) \div (10 \times 7)$ can be diagrammed as shown below.



  - To convert a **Circle of Evaluation** into code, we walk through the circle from outside-in, moving left-to-right. We type an open parenthesis when we *start* a circle, and a close parenthesis when we *end* one. Once we're in a circle, we write whatever is on the left of the circle, then the **function** at the top, and then whatever is on the right. The circle above, for example, would be programmed as `(1 - 4) / (10 * 7)`.
- **Images** are pictures that are produced by functions. The `circle` function, for example, takes a Number as the radius, a String to determine if the circle should be `"solid"` or `"outline"`, and a String to specify the color. You can see the Circle of Evaluation and the Code below:




```
circle(50, "solid", "red")
```

- There are a *lot* of functions in this language! We can make many different shapes, manipulate Strings and Numbers, and a whole lot more. Keeping track of what every function takes in and what it gives back is impossible! To help us remember how to use each function, programmers write down something called a **Contract**. Contracts include the **Name** of the function, what it takes in (called the **Domain**) and what it gives back (called the **Range**). You have space at the very back of your workbook to write all the Contracts for functions that you discover!

Notice and Wonder

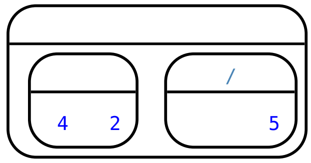
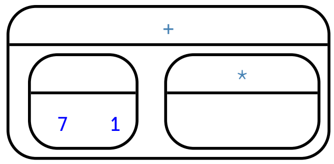
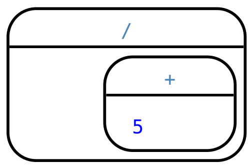
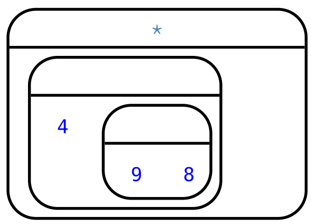
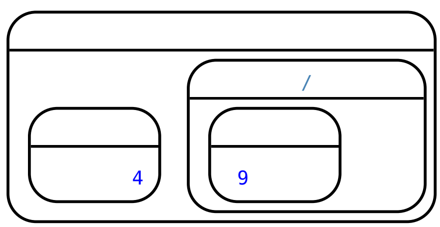
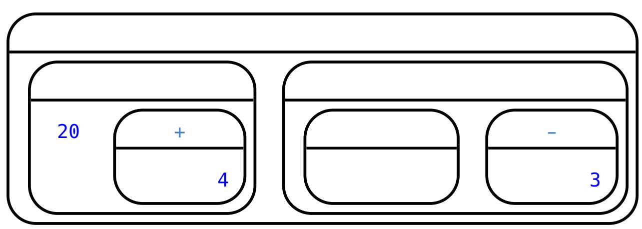
Try typing numbers into the Interactions Area, hitting "Enter", and see what you get back! Some ideas:

1. What is the largest number you can enter? The smallest?
2. Can you write decimals? Fractions?
3. After you get back a decimal, try clicking on it. What happens?
4. Can you write negative numbers? Negative fractions?
5. What else can you try?

What do you Notice?	What do you Wonder?

Completing Circles of Evaluation from Arithmetic Expressions (2)

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	$6 \times 4 + \frac{9 - -6}{5}$	
Challenge	$\frac{20}{6 + 4} - \frac{5 \times 9}{-12 - 3}$	

Creating Circles of Evaluation from Arithmetic Expressions (3)

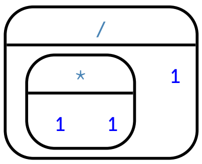
For each math expression on the left, draw its Circle of Evaluation on the right.

	Math Expression	Circle of Evaluation
1	$4-(6-17)$	
2	$25+14-12$	
3	$1+15\times 5$	
4	$\frac{15}{10+4\times -2}$	

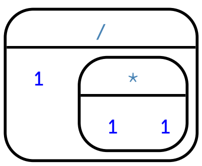
Matching Circles of Evaluation and Arithmetic Expressions

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

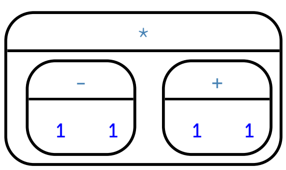
Circle of Evaluation | **Arithmetic Expression**



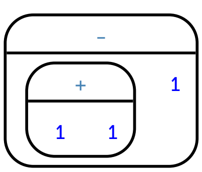
1 **A** $\frac{1}{1 \times 1}$



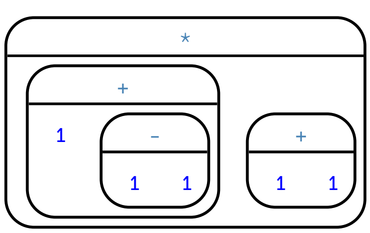
2 **B** $1 + 1 - 1$



3 **C** $\frac{1 \times 1}{1}$



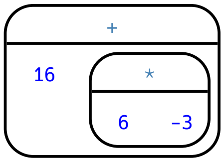
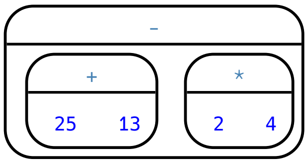
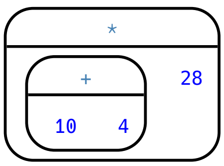
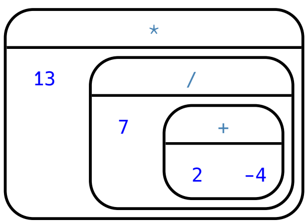
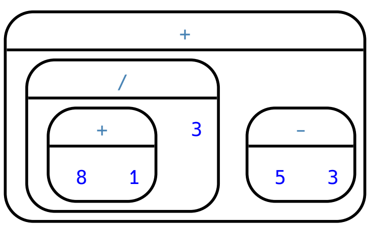
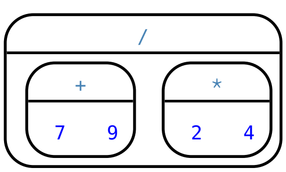
4 **D** $(1 + (1 - 1)) \times (1 + 1)$



5 **E** $(1 - 1) \times (1 + 1)$

Completing Partial Code from Circles of Evaluation

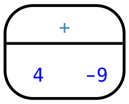
For each Circle of Evaluation on the left, finish the Code on the right by filling in the blanks.

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

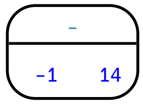
Translating Circles of Evaluation to Code

Translate the Circles of Evaluation into Code.

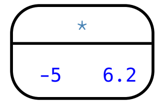
1)



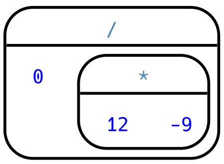
2)



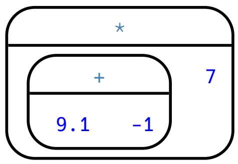
3)



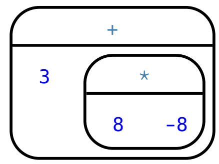
4)



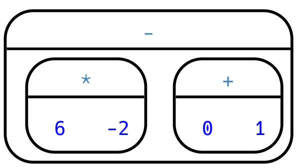
5)



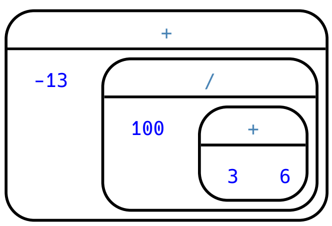
6)



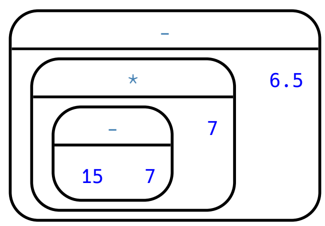
7)



8)

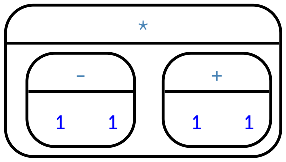
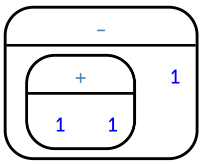
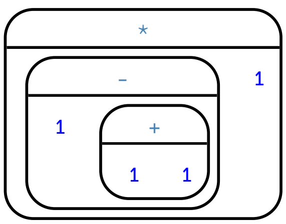
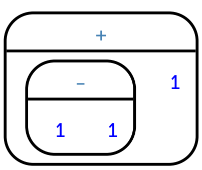
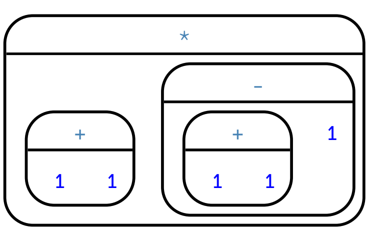


9)



Matching Circles of Evaluation & Code

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

Circle of Evaluation			Code
	1	A	`(1 - (1 + 1)) * 1`
	2	B	`(1 - 1) * (1 + 1)`
	3	C	`(1 + 1) * ((1 + 1) - 1)`
	4	D	`(1 + 1) - 1`
	5	E	`(1 - 1) + 1`

Arithmetic Expressions to Circles of Evaluation & Code

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

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

Translating Circles of Evaluation to Code - w/Square Roots

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

HINT: The function name is `num-sqrt` .

	Arithmetic	Circle of Evaluation	Code
1	$\sqrt{9}$		
2	$\sqrt{5+1}$		
3	$\sqrt{4}+1$		
4	$3\sqrt{3}+\sqrt{7}$		

Exploring Image Functions

By now you know how to make stars in this programming language. Can you figure out how to make triangles, based on what you know about making stars? Rectangles? What other shapes might we be able to make? When you've discovered code to make a new shape, draw the Circle of Evaluation in the table below, along with a sketch of the shape. Then add the function to your contracts page.

1) Use the space below to draw the Circles of Evaluation for the new functions, and draw a picture of what the function produces.

Circle of Evaluation		Image
`star` `50 "solid" "black"`	produces →	
	produces →	
	produces →	
	produces →	

Mystery Functions!

2) There is a function called `regular-polygon` with 4 inputs. What do they mean?

---

3) There is a function called `radial-star` with 5 inputs. What do they mean?

---

4) There is a function called `text` . Try to figure out how to use it! What do the inputs mean?

Reading for Domain and Range

As you think about the functions below, remember that you can always type them into your interactions window in the Editor!

1) What is the **name** of the function being used in: `string-length("broccoli") + 8`	
2) What is the **domain** of the outermost function being used in: `scale(2, circle(40, "solid", "blue"))`	
3) What is the **domain** of the innermost function being used in: `scale(2, circle(40, "solid", "blue"))`	
4) How many **arguments** does the `+` operator take in: `string-length("broccoli") + 8`	
5) What is the **range** of the function `string-length` ?	
6) Is `text` a *String* , a *function* , or an *Image* ?	
7) Is the **range** of `text` a *String* or an *Image* ?	
8) What is the first **argument** to the `circle` function in: `scale(2, circle(40, "solid", "blue"))`	

Composing Image Functions

You'll be investigating these functions with your partner:

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

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

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

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

4) Try out `rotate` , `flip-horizontal` , and `flip-vertical` . Use the space below to write your Code, then test out your Code in Pyret when you're ready.

Function Composition — Practice

1) Draw a Circle of Evaluation and write the Code for a **solid, green star, size 50** .

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.

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

Defining Values and Functions

- We can define values in our program, giving them names that we can refer to later instead of re-typing the same thing over and over. This works the same way it does in math: $x = 5 + 1$ defines the symbol x to be the number 6.
- In our language, we can define value by writing `var x = 5 + 1`. Here are a few value definitions:

```
x = 5 + 1
y = x * 7
food = "Pizza!"
dot = circle(y, "solid", "red")
```

- We can also define new **functions** in our language, to make it do things it didn't do before! To do this, we use a step-by-step process called the **Design Recipe**.
  - The first step is to write the **Contract** for the function you want to build. Remember, a Contract must include the Name, Domain and Range for the function!
  - 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!
  - The second step is to 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.
  - Circle the parts that are changing, and label them with a short **variable name** that explains what they do.
  - Finally, the third step is to 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!

Defining Values - Explore

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

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

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

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

Ideas: `eye-color` (a String), `age` (a Number), `fav-shape` (an Image)

Defining Values - Practice

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

____

Using the `PRIZE-STAR` definition from above, draw the Circle of Evaluation and write the Code for each of the exercises. One Circle of Evaluation has been done for you.

2) The outline of a pink star that is 3 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 of size 65 that has been rotated 45 degrees

Circle of Evaluation:

Code: ______________________________

5) The outline of a pink star that is 3 times the size of the original **and** has been rotated 45 degrees

Circle of Evaluation:

Code: ______________________________

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

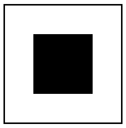
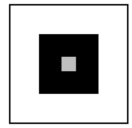
____

____

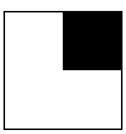
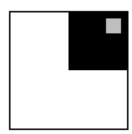
____

Combining Images

For each of the images below, write the code that would reproduce that image using `overlay`. The first one has been done for you. (The outermost square is of size 80)

	
`overlay(` `  square(40, "solid", "black"),` `  square(80, "outline", "black"` `))`	

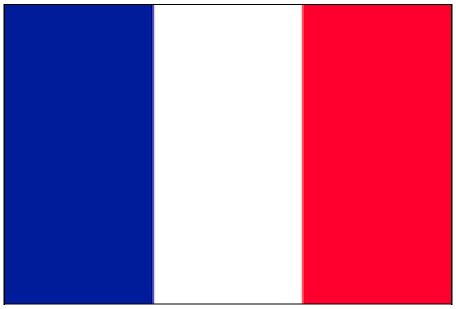
For each of the images below, write the code that would reproduce that image using `put-image`. The first one has been done for you. (The outermost square is of size 80)

	
`put-image(` `  square(40, "solid", "black"),` `  60, 60,` `  square(80, "outline", "black"` `))`	

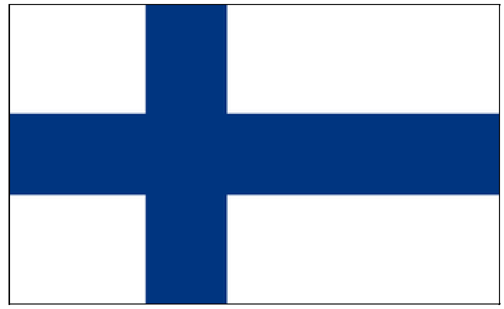
Decomposing Flags

Each of the flags below is shown with their width and height. Identify the shapes that make up each flag - including color and dimensions - that make up each flag. Use the flag's dimensions to estimate the dimensions of the different shapes.

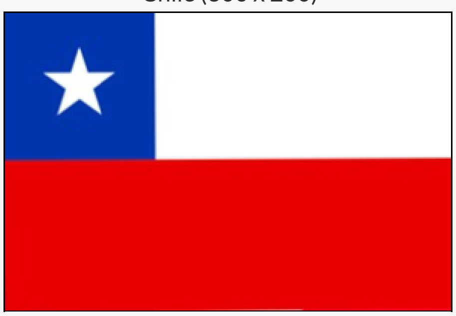
France (300 x 200)



Finland (300 x 184)



Chile (300 x 200)



Niger (350 x 300)

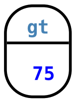
Cameroon (300 x 200)

Cuba (300 x 184)

Mapping Examples with Circles of Evaluation

Contract: ______________________________

Purpose Statement: ______________________________

If I type...	→	It should map to...
EXAMPLE #1: Circle of Evaluation  Code: `gt(75)`	→	Circle of Evaluation:  Code: `triangle(75, "solid", "green")`
EXAMPLE #2: Circle of Evaluation Code:	→	Circle of Evaluation: Code:

Fast Functions

There is space below to define four different functions, writing their Contracts, two examples, and the definition itself. The function `gt` - which makes solid green triangles of a given size - is provided as an example. Can you define `bc` as a function which makes solid blue circles of a given *radius*?

```
# gt :: Number -> Image
examples:
  gt ( 10 ) is triangle(10, "solid", "green")
  gt ( 20 ) is triangle(20, "solid", "green")
end
fun gt( size ):
  triangle(size, "solid", "green")
end
```

```
# ________ :: ________________________ -> ________
examples:
  ________ ( ________ ) is ________________________
  ________ ( ________ ) is ________________________
end
fun ________ ( ________ ):
  ________________________________________
end
```

```
# ________ :: ________________________ -> ________
examples:
  ________ ( ________ )
      is ________________________________________
  ________ ( ________ )
      is ________________________________________
end
fun ________ ( ________ ):
  ________________________________________
end
```

```
# ________ :: ________________________ -> ________
examples:
  ________ ( ________ )
      is ________________________________________
  ________ ( ________ )
      is ________________________________________
end
```



```
fun ________ ( ________ ):
```

Word Problem: rocket-height

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

Contract and Purpose Statement

Every contract has three parts…

__________ :: ____________________________ -> __________

function name *domain* *range*

____

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

__________ **(** __________ **) is** ____________________

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

__________ **(** __________ **) is** ____________________

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

end

Definition

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

fun __________ **(** __________ **):**

function name *variable(s)*

____

what the function does with those variable(s)

end

Writing Quality Purpose Statements

3 Reads

1st Read: What is this problem about?	2nd Read: What are the Quantities?
3rd Read: What is a good Purpose Statement?	

Stronger & Clearer

Purpose Statement 1st Revision:
Purpose Statement 2nd Revision:

Mapping Examples with Circles of Evaluation

Contract: ______________________________

Purpose Statement: ______________________________

If I type…	→	It should map to…
EXAMPLE #1: Circle of Evaluation Code:	→	Circle of Evaluation: Code:
EXAMPLE #2: Circle of Evaluation Code:	→	Circle of Evaluation: Code:

The Design Recipe

Directions : Write a function `marquee` that takes in a message and returns that message in large gold letters.

Contract and Purpose Statement

Every contract has three parts...

_______________ :: ____ -> _______________

function name | domain | range

____

what does the function do?

Examples

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

examples:

_______________ (_______________) **is** _______________________________

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

_______________ (_______________)

function name | input(s)

is ____

what the function produces

end

Definition

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

fun _______________ (_______________):

function name | variable(s)

____

what the function does with those variable(s)

end

Directions : Write a function `circle-area` that takes in a radius and returns the area of the circle.

Contract and Purpose Statement

Every contract has three parts...

_______________ :: ____ -> _______________

function name | domain | range

____

what does the function do?

Examples

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

examples:

_______________ (_______________) **is** _______________________________

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

_______________ (_______________) **is** _______________________________

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

end

Definition

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

fun _______________ (_______________):

function name | variable(s)

____

what the function does with those variable(s)

end

The Design Recipe

Directions : Write a function `minimum-wage` , that takes in a number of hours worked and returns the amount a worker will get paid at $10.25/hr.

Contract and Purpose Statement

Every contract has three parts...

________________ :: ____ -> ________________

function name | domain | range

____

what does the function do?

Examples

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

examples:

________________ (________________) **is** ________________________________

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

________________ (________________) **is** ________________________________

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

end

Definition

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

fun ________________ (________________):

function name | variable(s)

____

what the function does with those variable(s)

end

Directions : Write a function `tip-calculator` that takes in the cost of a meal and returns the 15% tip for that meal.

Contract and Purpose Statement

Every contract has three parts...

________________ :: ____ -> ________________

function name | domain | range

____

what does the function do?

Examples

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

examples:

________________ (________________) **is** ________________________________

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

________________ (________________) **is** ________________________________

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

end

Definition

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

fun ________________ (________________):

function name | variable(s)

____

what the function does with those variable(s)

end

The Design Recipe

Directions : Getting a gym membership costs $150, and then there's a $45/month fee after that. Write a function `globo-gym` that takes in a number of months and produces the cost of a membership for that many months.

Contract and Purpose Statement

Every contract has three parts...

__________ :: ____________________ -> __________

function name | domain | range

____

what does the function do?

Examples

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

examples:

__________ (__________) **is** ____________________

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

__________ (__________) **is** ____________________

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

end

Definition

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

fun __________ (__________):

function name | variable(s)

____

what the function does with those variable(s)

end

Directions : The cost of a ride is a starting price of $2.50, plus $1.50/mile. Write a function `rideshare` , that takes in a number of miles and produces the cost of that 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...

examples:

__________ (__________) **is** ____________________

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

__________ (__________) **is** ____________________

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

end

Definition

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

fun __________ (__________):

function name | variable(s)

____

what the function does with those variable(s)

end

The Design Recipe

Directions : Write a function `moving` that takes in the days and number of miles driven and returns the cost of renting a truck. The truck is $55 per day and each driven mile is 15¢.

Contract and Purpose Statement

Every contract has three parts...

__________ :: ____________________ -> __________

function name *domain* *range*

____

what does the function do?

Examples

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

```
examples:
  ________ ( __________ ) is ____________________
  function name   input(s)     what the function produces
  ________ ( __________ ) is ____________________
  function name   input(s)     what the function produces
end
```

Definition

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

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

  ______________________________________________
    what the function does with those variable(s)
end
```

Directions : Write a function `lawn-area` that takes in the length and width of a rectangular lawn and returns its area.

Contract and Purpose Statement

Every contract has three parts...

__________ :: ____________________ -> __________

function name *domain* *range*

____

what does the function do?

Examples

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

```
examples:
  ________ ( __________ ) is ____________________
  function name   input(s)     what the function produces
  ________ ( __________ ) is ____________________
  function name   input(s)     what the function produces
end
```

Definition

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

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

  ______________________________________________
    what the function does with those variable(s)
end
```

The Design Recipe

Directions : Write a function `rect-perimeter` that takes in the length and width of a rectangle and returns the perimeter of that rectangle.

Contract and Purpose Statement

Every contract has three parts...

________________ :: ________________________________ -> ________________

function name | domain | range

____

what does the function do?

Examples

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

examples:

________________ (________________) **is** ________________________________

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

________________ (________________) **is** ________________________________

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

end

Definition

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

fun ________________ (________________):

function name | variable(s)

____

what the function does with those variable(s)

end

Directions : Write a function `rectprism-vol` that takes in the length, width, and height of a rectangular prism and returns the Volume of a rectangular prism.

Contract and Purpose Statement

Every contract has three parts...

________________ :: ________________________________ -> ________________

function name | domain | range

____

what does the function do?

Examples

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

examples:

________________ (________________) **is** ________________________________

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

________________ (________________) **is** ________________________________

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

end

Definition

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

fun ________________ (________________):

function name | variable(s)

____

what the function does with those variable(s)

end

The Design Recipe

Directions : Write a function `split-tab` that takes in a cost and the number of people sharing the bill and splits the cost equally.

Contract and Purpose Statement

Every contract has three parts...

______________ :: ____ -> ______________

function name *domain* *range*

____

what does the function do?

Examples

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

examples:

______________ (______________) **is** ______________________________

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

______________ (______________) **is** ______________________________

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

end

Definition

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

fun ______________ (______________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Directions : Write a function `num-cube` that takes in a number and returns the cube of that number.

Contract and Purpose Statement

Every contract has three parts...

______________ :: ____ -> ______________

function name *domain* *range*

____

what does the function do?

Examples

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

examples:

______________ (______________) **is** ______________________________

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

______________ (______________) **is** ______________________________

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

end

Definition

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

fun ______________ (______________):

function name *variable(s)*

____

what the function does with those variable(s)

end

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.

Contract and Purpose Statement

Every contract has three parts...

______________ :: ____ -> ______________

function name | domain | range

____

what does the function do?

Examples

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

examples:

______________ (______________) **is** ______________________________

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

______________ (______________) **is** ______________________________

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

end

Definition

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

fun ______________ (______________):

function name | variable(s)

____

what the function does with those variable(s)

end

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

Contract and Purpose Statement

Every contract has three parts...

______________ :: ____ -> ______________

function name | domain | range

____

what does the function do?

Examples

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

examples:

______________ (______________) **is** ______________________________

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

______________ (______________) **is** ______________________________

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

end

Definition

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

fun ______________ (______________):

function name | variable(s)

____

what the function does with those variable(s)

end

Problem Decomposition

- Sometimes a problem is too complicated to solve all at once. Maybe there are too many variables, or there is just so much information that we can't get a handle on it!
- We can use **Problem Decomposition** to break those problems down into simpler pieces, and then work with the pieces to solve the whole. There are two strategies we can use for decomposition:
  - **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 either one!

Word Problem: revenue

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

Contract and Purpose Statement

Every contract has three parts...

\# ________________ :: ____ -> ________________

function name *domain* *range*

\# ____

what does the function do?

Examples

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

```
examples:
```

________________ (________________) **is** ________________________________

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

________________ (________________) **is** ________________________________

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

```
end
```

Definition

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

fun ________________ (________________):

function name *variable(s)*

____

what the function does with those variable(s)

```
end
```

Word Problem: cost

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...

examples:

__________ (______________) **is** ______________________________

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

__________ (______________) **is** ______________________________

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

end

Definition

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

fun __________ (__________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Word Problem: profit

Directions : Use the Design Recipe to write a function `profit` that calculates 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...

examples:

__________ (__________) **is** ____________________

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

__________ (__________) **is** ____________________

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

end

Definition

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

fun __________ (__________):

function name *variable(s)*

____

what the function does with those variable(s)

end

Introduction to Computational Data Science

Many important questions ("What's the best restaurant in town?", "Is this law good for citizens?", etc.) are answered with *data* .
Data Scientists try and answer these questions by writing *programs that ask questions about data* .
Data of all types can be organized into **Tables** .

- Every Table has a **header row** and some number of **data rows**.
- **Quantitative data** is numeric and measures *an amount*, such as a person's height, a score on a test, distance, etc. A list of quantitative data can be ordered from smallest to largest.
- **Categorical data** is data that specifies *qualities*, such as sex, eye color, country of origin, etc. Categorical data is not subject to the laws of arithmetic — for example, we cannot take the "average" of a list of colors.

Answering questions with data can take many forms. Here are a few types of questions, each requiring a different kind of analysis:

- **Lookup Questions** can be answered just by finding the right row and column of a table. (e.g., "How old is Toggle?")
- **Compute Questions** can be answered by computing over a single row or column. (e.g., "What is the average weight of animals from the shelter?")
- **Relate Questions** require looking for trends across multiple columns. (e.g., "Do cats tend to be adopted sooner than dogs?")

The Animals Dataset

name	species	sex	age	fixed	legs	pounds	weeks
Sasha	cat	female	1	false	4	6.5	3
Snuffles	rabbit	female	3	true	4	3.5	8
Mittens	cat	female	2	true	4	7.4	1
Sunflower	cat	female	5	true	4	8.1	6
Felix	cat	male	16	true	4	9.2	5
Sheba	cat	female	7	true	4	8.4	6
Billie	snail	hermaphrodite	0.5	false	0	0.1	3
Snowcone	cat	female	2	true	4	6.5	5
Wade	cat	male	1	false	4	3.2	1
Hercules	cat	male	3	false	4	13.4	2
Toggle	dog	female	3	true	4	48	1
Boo-boo	dog	male	11	true	4	123	24
Fritz	dog	male	4	true	4	92	3
Midnight	dog	female	5	false	4	112	4
Rex	dog	male	1	false	4	28.9	9
Gir	dog	male	8	false	4	88	5
Max	dog	male	3	false	4	52.8	8
Nori	dog	female	3	true	4	35.3	1
Mr. Peanutbutter	dog	male	10	false	4	161	6
Lucky	dog	male	3	true	3	45.4	9
Kujo	dog	male	8	false	4	172	30
Buddy	lizard	male	2	false	4	0.3	3
Gila	lizard	female	3	true	4	1.2	4
Bo	dog	male	8	true	4	76.1	10
Nibblet	rabbit	male	6	false	4	4.3	2
Snuggles	tarantula	female	2	false	8	0.1	1
Daisy	dog	female	5	true	4	68	8
Ada	dog	female	2	true	4	32	3
Miaulis	cat	male	7	false	4	8.8	4
Heathcliff	cat	male	1	true	4	2.1	2
Tinkles	cat	female	1	true	4	1.7	3
Maple	dog	female	3	true	4	51.6	4

Categorical or Quantitative?

For each piece of data below, circle whether it is **Categorical** or **Quantitative** data.

1	Hair color	categorical	quantitative
2	Age	categorical	quantitative
3	ZIP Code	categorical	quantitative
4	Year	categorical	quantitative
5	Height	categorical	quantitative
6	Sex	categorical	quantitative
7	Street Name	categorical	quantitative

For each question, circle whether it will be answered by **Categorical** or **Quantitative** data.

8	We'd like to find out the average price of cars in a lot.	categorical	quantitative
9	We'd like to find out the most popular color for cars.	categorical	quantitative
10	We'd like to find out which puppy is the youngest.	categorical	quantitative
11	We'd like to find out which cats have been fixed.	categorical	quantitative
12	We want to know which people have a ZIP code of 02907.	categorical	quantitative
13	We'd like to sort a list of phone numbers by area code.	categorical	quantitative

Questions and Column Descriptions

What questions can you ask about the animals dataset? Come up with at least one **Lookup**, **Compute**, **Relate** or **Can't Answer** question, and write them as wonders below. (Note: These question types are defined on Page 1.)

What do you NOTICE about this dataset?	What do you WONDER about this dataset?	*Question Type*
		Lookup *Compute* *Relate* *Can't answer*
		Lookup *Compute* *Relate* *Can't answer*
		Lookup *Compute* *Relate* *Can't answer*
		Lookup *Compute* *Relate* *Can't answer*
		Lookup *Compute* *Relate* *Can't answer*
		Lookup *Compute* *Relate* *Can't answer*

1. This dataset is ____Animals that came from an animal shelter____, which contains __31__ data rows.

2. Some of the columns are:

   a. ____species____, which contains ____categorical____ data. Some example values are:

   ____"cat", "dog", and "rabbit"____.

   b. ________________, which contains ________________ data. Some example values are:

   ________________________________.

What's on your mind?

Introduction to Programming in Pyret

Programming languages involve different *datatypes*, such as Numbers, Strings, and Booleans.

- Numbers are values like `1`, `0.4`, `1/3`, and `-8261.003`.
  - Numbers are *usually* used for quantitative data and other values are *usually* used as categorical data.
  - In Pyret, any decimal *must* start with a 0. `0.22` is valid, but `.22` is not.
- Strings are values like `"Emma"`, `"Rosanna"`, `"Jen and Ed"`, or even `"08/28/1980"`.
  - In Pyret, all strings *must* be surrounded in quotation marks.
- Booleans are either `true` or `false`.

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

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

Applying Functions also works the way it does in math. The function name is first, followed by a list of **arguments** in parentheses.

- In math this could look like `f(5)` or `f(g(10, 4))`.
- In Pyret this could look like `star(50, "solid", "red")`.
- There are many other Pyret functions, for example `num-sqr`, `num-sqrt` `triangle`, `star`, `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 *types of values* the function consumes, and in what order.
- The *Range* of the function - what *type of value* the function produces.

Value Definitions (like `x = 4`, or `y = 9 + 6`) also work the way they do in math. Every value definition starts with a *name*, followed by an equals sign, and then an expression. Once a value is defined, it can be refered to by name.

Numbers and Strings

Make sure you've loaded the code.pyret.org editor, and clicked "Run".

1. Try typing `42` into the Interactions Area and hitting "Enter". What happens?
2. Try typing in other Numbers. What happens if you try a decimal like `0.5` ? A fraction like `1/3` ? Try really big Numbers, and really small ones.
3. String values are always in quotes. Try typing your name (in quotes!). What happens when you hit Enter?
4. Try typing your name with the opening quote, but *without* the closing quote. What happens? Now try typing it without any quotes.
5. Is *42* the same as `"42"` ? Why or why not? Write your answer below:

Operators

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

7. Type in the following expressions, one at a time: `4 + 2 + 6` , `4 + 2 * 6` , `4 + (2 * 6)` . What do you notice? Write your answer below:

8. Try typing in `4 + "cat"` , and then `"dog" + "cat"` . What can you conclude from this? Write your answer below:

Booleans

Boolean expressions are yes-or-no questions and will always evaluate to either `true` ("yes") or `false` ("no"). What will each of the expressions below evaluate to? Write down the result in the blanks provided, and type them into Pyret if you're not sure.

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

13) In your own words, desribe what **>** does. ______________________

14) In your own words, describe what **<=** does. ______________________

15) In your own words, desribe what **<>** does. ______________________

16) How many **Numbers** are there in the entire universe? ______________________

17) How many **Strings** are there in the entire universe? ______________________

18) How many **Images** are there in the entire universe? ______________________

19) How many **Booleans** are there in the entire universe? ______________________

Defining Functions

We can **define our own functions**, using a technique called the **Design Recipe**.

- We use the Design Recipe to help us define functions **and think through problems clearly**.
- The first step is to write a **Contract** and **Purpose Statement** for the function, which specify the Name, Domain and Range of the function and give a summary of what it does.
- The second step is to **write at least two examples**, which show how the function should work for specific inputs. These examples help us see patterns, and we express those patterns by **circling and labeling** what changes.
- The final step is to **define the function**, which generalizes our examples.

The Design Recipe

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:: (size :: Number) -> Image

function name | domain | range

\# Consumes a size, and produces a solid green triangle of that size.

what does the function do?

Examples

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

```
examples:
    ______________ ( ______________ ) is ______________
    function name     input(s)          what the function produces
    ______________ ( ______________ ) is ______________
    function name     input(s)          what the function produces
end
```

Definition

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

```
fun gt( size ):
    function name   variable(s)
  triangle(size, "solid", "green")
    what the function does with those variable(s)
end
```

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

\# ______________

what does the function do?

Examples

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

```
examples:
    ______________ ( ______________ ) is ______________
    function name     input(s)          what the function produces
    ______________ ( ______________ ) is ______________
    function name     input(s)          what the function produces
end
```

Definition

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

```
fun ______________ ( ______________ ):
    function name     variable(s)
  ______________
    what the function does with those variable(s)
end
```

The Design Recipe

Directions : Define a function called `sticker` , which draws 50px stars in whatever color is input.

Contract and Purpose Statement

Every contract has three parts...

\# ______________ :: ______________________________________ -> ______________

function name — domain — range

\# ____

what does the function do?

Examples

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

examples:

______________ (______________) **is** ______________________________

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

______________ (______________) **is** ______________________________

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

end

Definition

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

fun ______________ (______________):

function name — variable(s)

____

what the function does with those variable(s)

end

Directions : Define a function called `nametag` , which consumes a `Row` of the animals table and draws their name in purple, 10px letters. (Assume you have rows `animalA` and `animalB` defined.)

Contract and Purpose Statement

Every contract has three parts...

\# nametag:: (r :: Row) -> Image

function name — domain — range

\# Consumes an animal, and produces that animal's name in purple, 10px letters.

what does the function do?

Examples

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

examples:

nametag ("animalA")

function name — input(s)

is ______________________________

what the function produces

______________ (______________)

function name — input(s)

is ______________________________

what the function produces

end

Definition

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

fun nametag(r):

function name — variable(s)

text(r["name"], 10, "purple")

what the function does with those variable(s)

end

What's on your mind?

Applying Functions

Type this line of code into the interactions area and hit "Enter": `triangle(50, "solid", "red")`

1)	What is the name of this function?	________________
2)	What did the expression evaluate to?	________________
3)	How many arguments does `triangle` expect?	________________
4)	What does the `triangle` function produce? (Numbers? Strings? Booleans?)	________________

Catching Bugs

The following lines of code are all BUGGY! Can you spot the mistake? If you have time, type in the buggy code and see if Pyret agrees with you!

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

Can you spot the mistake? ________________

What error message does Pyret return? ________________

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

Can you spot the mistake? ________________

What error message does Pyret return? ________________

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

Can you spot the mistake? ________________

What error message does Pyret return? ________________

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

Can you spot the mistake? ________________

What error message does Pyret return? ________________

9) `triangle 20, "solid", "red")`

Can you spot the mistake? ________________

What error message does Pyret return? ________________

Contracts

Consider the following contract:

```
rotate :: (degree :: Number, img :: Image) -> Image
```

What is the **Name** of this function? ______________________________

How many things are in this function's **Domain** ? ______________________________

What is the **type** of this function's **first argument** ? ______________________________

What is the **name** of this function's **second argument** ? ______________________________

What is the **Range** of this function? ______________________________

Circle the expression below that is the correct application of this function, based on its contract.

1. `rotate(45, 90)`
2. `rotate(circle(99, "solid", "green"))`
3. `rotate(25, rectangle(7, 10, "outline", "black"))`
4. `rotate(rectangle(7, 10, "outline", "black"), 25)`

Matching Expressions and Contracts

Match the contract (left) with the expression described by the function being used (right).

Contract			Expression
make-id :: (name :: String, age :: Number) -> Image	1	A	`make-id("Hannah", "Smith")`
phone-bill :: (minutes :: Number, texts :: Number) -> Number	2	B	`make-id("George", 17)`
phone-bill :: (minutes :: Number) -> Number	3	C	`phone-bill(31, 287)`
make-id :: (first :: String, last :: String) -> Image	4	D	`make-id("Jessica", "Jones", 32)`
make-id :: (first :: String, last :: String, age :: Number) -> Image	5	E	`phone-bill(55)`

What's on your mind?

Plotting and Displaying Data

Data Scientists use **displays** to visualize data. You've probably seen some of these charts, graphs and plots yourselves! When it comes to displaying **Categorical Data** , there are two displays that are especially useful.

1. **Bar charts** show the *count or percentage* of rows in each category.
    - Bar charts provide a visual representation of the frequency of values in a categorical column.
    - Bar charts have a bar for every category in a column.
    - The more rows in a category, the taller the bar.
    - Bars in a bar chart can be show in *any order*, without changing the meaning of the chart. However, bars are usually shown in some sensible order (bars for the number of orders for different t-shirt sizes might be presented in order of smallest to largest shirt).
2. **Pie charts** show the *percentage* of rows in each category.
    - Pie charts provide a visual representation of the relative frequency of values in a categorical column.
    - Pie charts have a slice for every category in a column.
    - The more rows in a category, the larger the slice.
    - Slices in a pie chart can be shown in *any order*, without changing the meaning of the chart. However, slices are usually shown in some sensible order (e.g. slices might be shown in alphabetical order or from the smallest to largest slice).

Exploring Displays

Using your Contracts page and the Animals Starter File, make each type of display below in pyret. Then sketch the displays and answer the questions. Be sure to add examples of the code you use to your contracts page!

Pie Charts	Bar Charts
Sketch a pie chart here.	Sketch a bar chart here.
Pie charts are constructed from 1 column(s).	Bar charts are constructed from ______ column(s).
They show **categorical** data.	They show ______ data.
What does this display tell us? ______ ______	What does this display tell us? ______ ______

Box Plots	Histograms
Sketch a box plot here.	Sketch a histogram here.
Box plots are constructed from ______ column(s).	Histograms are constructed from ______ column(s).
They show ______ data.	They show ______ data.
What does this display tell us? ______ ______	What does this display tell us? ______ ______

(More) Exploring Displays

For each type of display, fill in the information below.

Scatter Plots	Linear Regression Plots
Sketch a box plot here.	Sketch a histogram here.
Box plots are constructed from ________ column(s).	Histograms are constructed from ________ column(s).
They show ________________________ data.	They show ________________________ data.
What does this display tell us? ____ ____	What does this display tell us? ____ ____

What’s on your mind?

Data Displays and Lookups

Data scientists use data visualizations to gain better insights into their data, and to communicate their findings with others. Making a display requires answering three questions:

1. **What data** is being displayed? This could be "a random sample of 2000 people", "every animal from the shelter", or "students' aged 14-17".
2. **What variables** are being explored? Are we looking at the `species` column? The number of `kilograms` that an animal weighs? Searching for a relationship between a person's `income` and their `height` ?
3. **What display** is being used, given the variables being explored? If it's a quantitative variable, we might use a histogram or box plot. If it's categorical, we could use a pie or bar chart. If it's two quantitative variables, we probably want a scatter plot.

When **looking up a data Row** from a Table, programmers use the `row-n` method. This method takes a single number as its input, which tells the computer which Row we want. *Note: Rows are numbered starting at zero!*
For example:

```
animals-table.row-n(0)  # access the 1st data row
animals-table.row-n(16) # access the 17th data row
```

When **looking up a column** from a Row, programmers use square brackets and the name of the column they want.
For example:

```
animals-table.row-n(11)["age"]      # look up the age of the animal in the 12st data row
animals-table.row-n(14)["species"]  # look up the species of the animal in the 15th data
row
```

Throughout the rest of the workbook, we will sometimes refer to animalA and animalB.

```
animalA = animals-table.row-n(4)
animalB = animals-table.row-n(13)
```

What Display Goes with Which Data?

Match the Display with the description of the data being plotted. Some descriptions may go with more than one display!

Pie Charts	**1**	**A**	1 column of Quantitative Data
Bar Charts	**2**		
Histograms	**3**	**B**	2 columns of Quantitative Data
Box Plots	**4**		
Scatter Plots	**5**	**C**	1 column of Categorical Data

Data Displays

Fill in the tables below, then write the Pyret code that will make that display. The first column has been filled in for you.

1) A `pie-chart` showing the `species` of animals from the shelter.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

2) A `bar-chart` showing the `sex` of animals from the shelter.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

3) A `histogram` of the number of `pounds` that animals weigh.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

4) A `box-plot` of the number of `pounds` that animals weigh.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

5) A `scatter-plot` , using the animals' `species` as the labels, `age` as the x-axis, and `pounds` as the y-axis.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

6) A `scatterplot` , using the animals' `name` as the labels, `pounds` as the x-axis, and `weeks` as the y-axis.

Which Rows?	Which Column(s)?	What Display?
All the animals		

code: ______________________________

Lookup Questions

The table below represents four pets:

`pets-table`

name	sex	age	pounds
"Toggle"	"female"	3	48
"Fritz"	"male"	4	92
"Nori"	"female"	6	35.3
"Maple"	"female"	3	51.6

1) *Match* each Lookup Question (left) to the code that will give the answer (right).

"How much does Maple weigh?"	**1**	**A**	`pets-table.row-n(3)`
"Which is the last row in the table?	**2**	**B**	`pets-table.row-n(2)["name"]`
"What is Fritz's sex?"	**3**	**C**	`pets-table.row-n(1)["sex"]`
"What's the third animal's name?"	**4**	**D**	`pets-table.row-n(3)["age"]`
"How much does Nori weigh?"	**5**	**E**	`pets-table.row-n(3)["pounds"]`
"How old is Maple?"	**6**	**F**	`pets-table.row-n(0)`
"What is Toggle's sex?"	**7**	**G**	`pets-table.row-n(2)["pounds"]`
"What is the first row in the table?"	**8**	**H**	`pets-table.row-n(0)["sex"]`

2) Fill in the blanks (left) with code that will produce the value (right).

a.	pets-table.row-n(3)["name"]	"Maple"
b.	______________________________	"male"
c.	______________________________	4
d.	______________________________	48
e.	______________________________	"Nori"

What's on your mind?

Defining Row Functions & Using Table Methods

Methods are special functions that are attached to pieces of data. We use them to manipulate Tables.

- In this course, the methods we'll be using are
  - `row-n` - consumes an index (starting with zero!) and produces a row from a table
  - `order-by` - consumes the name of a column and a Boolean value to determine if that table should be sorted by that column in ascending order
  - `filter` - consumes a *Boolean-producing function*, and produces a table containing only rows for which the function returns `true`
  - `build-column` - consumes the name of a new column, and a function that produces the values in that column for each Row
- Unlike functions, methods can't be used alone. They have a "secret" argument, which is the data they are attached to. They are written as part of that data, separated by a dot. For example:

```
shapes.row-n(2)
```

- Contracts for methods are different from other functions. They include the type of the data as part of their names. For example:

```
<table>.row-n :: (index :: Number) -> Row
```

Reading Function Definitions

Make sure you have the "Table Methods Starter File" open on your computer, and click "Run".

1	How many functions are defined here?	
2	What are their names?	
3	What is the domain of `is-dog` ?	
4	What is the range of `is-old` ?	
5	What is the range of `lookup-name` ?	
6	What does `is-fixed(animalA)` evaluate to?	
7	What does `lookup-name(animalB)` evaluate to?	
8	What does `is-old(animalA)` evaluate to?	
9	What does `is-dog(animalA)` evaluate to?	
10	What does `is-fixed` do?	
11	What does `lookup-name` do?	
12	What does `is-old` do?	

The Design Recipe

For the word problems below, assume `animalA` and `animalB` are defined as the data rows for Felix and Midnight, respectively.

Directions : Define a function called `lookup-fixed` , which looks up whether or not an animal is fixed.

Contract and Purpose Statement

Every contract has three parts...

\# lookup-fixed:: (r :: Row) -> Boolean

function name | domain | range

\# Consumes an animal, and looks up the value in the fixed column.

what does the function do?

Examples

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

examples:

_____ (_____) **is** _____

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

_____ (_____) **is** _____

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

end

Definition

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

fun lookup-fixed(r):

function name | variable(s)

r["fixed"]

what the function does with those variable(s)

end

Directions : Define a function called `lookup-sex` , which consumes a Row of the animals table and looks up the sex of that animal.

Contract and Purpose Statement

Every contract has three parts...

\# _____ :: _____ -> _____

function name | domain | range

\# _____

what does the function do?

Examples

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

examples:

_____ (_____) **is** _____

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

_____ (_____) **is** _____

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

end

Definition

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

fun _____ (_____):

function name | variable(s)

_____

what the function does with those variable(s)

end

The Design Recipe

For the word problems below, assume `animalA` and `animalB` are defined as the data rows for Felix and Midnight, respectively.

Directions : Define a function called `is-cat` , which consumes a `Row` of the animals table and *computes* whether the animal is a cat.

Contract and Purpose Statement

Every contract has three parts...

\# is-cat:: (r :: Row) -> Boolean

function name — domain — range

\# Consumes an animal, and computes whether the species equals "cat"

what does the function do?

Examples

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

examples:

is-cat (animalA) **is** ________

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

________ (________) **is** ________

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

end

Definition

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

fun is-cat(r):

function name — variable(s)

```
r["species"] == "cat"
```

what the function does with those variable(s)

end

Directions : Define a function called `is-young` , which consumes a Row of the animals table and *computes* whether it is less than four years old.

Contract and Purpose Statement

Every contract has three parts...

\# ________ :: ________ -> ________

function name — domain — range

\# ________

what does the function do?

Examples

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

examples:

________ (________) **is** ________

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

________ (________) **is** ________

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

end

Definition

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

fun ________ (________):

function name — variable(s)

____

what the function does with those variable(s)

end

What's on your mind?

Method Chaining

Method chaining allows us to apply multiple methods with less code.

For example, instead of using multiple definitions, like this:

```
with-labels = animals-table.build-column("labels", nametag)
cats = with-labels.filter(is-cat)
cats.order-by("age", true)
```

We can use method-chaining to write it all on one line, like this:

```
animals-table.build-column("labels", nametag).filter(is-cat).order-by("age", true)
```

Order Matters! The methods are applied in the order they appear. For example, trying to order a table by a column that hasn't been built will result in an error.

The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Directions : Define a function called `is-dog` , which consumes a `Row` of the animals table and *computes* whether the animal is a dog.

Contract and Purpose Statement

Every contract has three parts...

\# is-dog:: (r :: Row) -> Boolean

function name | domain | range

\# Consumes an animal, and computes whether the species == "dog"

what does the function do?

Examples

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

```
examples:
  is-dog ( "animalA" ) is animalA["species"] == "dog"
  is-dog ( "animalB" ) is
end
```

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

Definition

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

```
fun is-dog( r ):
  r["species"] == "dog"
end
```

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

Directions : Define a function called `is-female` , which consumes a Row of the animals table and returns true if the animal is female.

Contract and Purpose Statement

Every contract has three parts...

\# ______ :: ______ -> ______

function name | domain | range

\# ______

what does the function do?

Examples

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

```
examples:
  ______ ( ______ ) is ______
  ______ ( ______ ) is ______
end
```

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

Definition

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

```
fun ______ ( ______ ):
  ______
end
```

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

The Design Recipe

For the word problems below, assume you have `animalA` and `animalB` defined in your code.

Directions : Define a function called `is-old` , which consumes a Row of the animals table and *computes* whether it is more than 12 years old.

Contract and Purpose Statement

Every contract has three parts…

______________ :: ____ -> ______________

function name domain range

____

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

______________ (______________) **is** ______________________________

function name input(s) what the function produces

______________ (______________) **is** ______________________________

function name input(s) what the function produces

end

Definition

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

fun ______________ (______________):

function name variable(s)

____

what the function does with those variable(s)

end

Directions : Define a function called `name-has-s` , which returns true if an animal's name contains the letter "s"

Contract and Purpose Statement

Every contract has three parts…

`name-has-s`:: ____ -> ______________

function name domain range

____

what does the function do?

Examples

Write some examples, then circle and label what changes…

examples:

______________ (______________)

function name input(s)

is ____

what the function produces

______________ (______________)

function name input(s)

is ____

what the function produces

end

Definition

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

fun `name-has-s`(`r`):

function name variable(s)

`string-contains(r["name"], "s")`

what the function does with those variable(s)

end

Chaining Methods

You have the following functions defined below (read them *carefully!*):

```
fun is-fixed(r): r["fixed"]                    end
fun is-young(r): r["age"] < 4                  end
fun nametag(r):  text(r["name"], 20, "red") end
```

The table `t` below represents four animals from the shelter:

name	sex	age	fixed	pounds
"Toggle"	"female"	3	true	48
"Fritz"	"male"	4	true	92
"Nori"	"female"	6	true	35.3
"Maple"	"female"	3	true	51.6

Match each Pyret expression (left) to the description of what it does (right).

`t.order-by("age", true)`	1	A	Produces a table containing only Toggle and Maple
`t.filter(is-fixed)`	2	B	Produces a table of only young, fixed animals
`t.build-column("sticker", nametag)`	3	C	Produces a table, sorted youngest-to-oldest
`t.filter(is-young)`	4	D	Produces a table with an extra column, named "sticker"
`t.filter(is-young)` `.filter(is-fixed)`	5	E	Produces a table containing Maple and Toggle, in that order
`t.filter(is-young)` `.order-by("pounds", false)`	6	F	Produces a table containing the same four animals
`t.build-column("label", nametag)` `.order-by("age", true)`	7	G	Won't run: will produce an error
`t.order-by("agee", false)`	8	H	Produces a table with an extra "label" column, sorted youngest-to-oldest

Chaining Methods 2: Order Matters!

You have the following functions defined below (read them *carefully!*):

```
fun is-female(r): r["sex"] == "female"  end
fun kilograms(r): r["pounds"] / 2.2     end
fun is-heavy(r):  r["kilos"] > 25       end
```

The table `t` below represents four animals from the shelter:

name	sex	age	fixed	pounds
"Toggle"	"female"	3	true	48
"Fritz"	"male"	4	true	92
"Nori"	"female"	6	true	35.3
"Maple"	"female"	3	true	51.6

Match each Pyret expression (left) to the description of what it does (right). **Note: one description might match multiple expressions!**

`t.order-by("kilos", true)`	1	A	Produces a table containing Toggle, Nori and Maple, with an extra column showing their weight in kilograms
`t.filter(is-female)` `  .build-column("kilos", kilograms)`	2	B	Produces a table containing Maple, Nori and Toggle (in that order)
`t.build-column("kilos", kilograms)` `  .filter(is-heavy)`	3	C	Produces a table containing only Fritz, with a single extra column called `kilos`
`t.filter(is-heavy)` `  .build-column("kilos", kilograms)`	4	D	Won't run: will produce an error
`t.build-column("kilos", kilograms)` `  .filter(is-heavy)` `  .order-by("sex", true)`	5	E	Produces a table containing only Fritz, with two extra columns
`t.build-column("female", is-female)` `  .build-column("kilos", kilograms)` `  .filter(is-heavy)`	6	F	Produces a table containing Maple and Fritz

What's on your mind?