













Method Chaining

Students continue practicing their Design Recipe skills, making lots of simple functions dealing with the Animals Dataset. Then they learn how to chain Methods together, and define more sophisticated subsets.

| Prerequisites | Defining Table Functions | | | | | | | | | | | | | | | | | | |
|--|---|---|-----------|--------|--------|--|--------------|--------|--------------------------------|---------------|---------|-----------------------------|-------------|-------|---|---|-------|---|--|
| Relevant Standards <div>OK K12CS CSTA NGSS</div> | Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere). | | | | | | | | | | | | | | | | | | |
| Lesson Goals | Students will be able to... <ul style="list-style-type: none">• Use method chaining to write more sophisticated analyses using less code• Identify bugs introduced by chaining methods in the wrong order | | | | | | | | | | | | | | | | | | |
| Student-facing Lesson Goals | <ul style="list-style-type: none">• Let’s practice writing functions and combining methods together. | | | | | | | | | | | | | | | | | | |
| Materials | <ul style="list-style-type: none">• Lesson slides (Google Slides)• Computer for each student (or pair), with access to the internet• Student workbook, and something to write with | | | | | | | | | | | | | | | | | | |
| Preparation | <ul style="list-style-type: none">• Make sure all materials have been gathered• Decide how students will be grouped in pairs• All students should log into CPO and open the "Animals Starter File" they saved from the prior lesson. If they don’t have the file, they can open a new one | | | | | | | | | | | | | | | | | | |
| Supplemental Resources | | | | | | | | | | | | | | | | | | | |
| Language Table | <table><tr><th>Types</th><th>Functions</th><th>Values</th></tr><tr><td>Number</td><td>num-sqrt, num-sqr, mean, median, modes</td><td>4, -1.2, 2/3</td></tr><tr><td>String</td><td>string-repeat, string-contains</td><td>"hello", "91"</td></tr><tr><td>Boolean</td><td>==, <, <=, >=, string-equal</td><td>true, false</td></tr><tr><td>Image</td><td>triangle, circle, star, rectangle, ellipse, square, text, overlay, bar-chart, pie-chart, bar-chart-summarized, pie-chart-summarized</td><td>   </td></tr><tr><td>Table</td><td>count, .row-n, order-by, .filter, .build-column</td><td></td></tr></table> | Types | Functions | Values | Number | num-sqrt, num-sqr, mean, median, modes | 4, -1.2, 2/3 | String | string-repeat, string-contains | "hello", "91" | Boolean | ==, <, <=, >=, string-equal | true, false | Image | triangle, circle, star, rectangle, ellipse, square, text, overlay, bar-chart, pie-chart, bar-chart-summarized, pie-chart-summarized |     | Table | count, .row-n, order-by, .filter, .build-column | |
| Types | Functions | Values | | | | | | | | | | | | | | | | | |
| Number | num-sqrt, num-sqr, mean, median, modes | 4, -1.2, 2/3 | | | | | | | | | | | | | | | | | |
| String | string-repeat, string-contains | "hello", "91" | | | | | | | | | | | | | | | | | |
| Boolean | ==, <, <=, >=, string-equal | true, false | | | | | | | | | | | | | | | | | |
| Image | triangle, circle, star, rectangle, ellipse, square, text, overlay, bar-chart, pie-chart, bar-chart-summarized, pie-chart-summarized |     | | | | | | | | | | | | | | | | | |
| Table | count, .row-n, order-by, .filter, .build-column | | | | | | | | | | | | | | | | | | |

Design Recipe Practice

25 minutes

Overview

Students practice more of what they learned in the previous lesson, applying the Design Recipe to simple table functions that operate on rows of the Animals Dataset. The functions they create - in addition to the ones they've already made - set up the method-chaining activity.

Launch

Launch

The Design Recipe is a powerful tool for solving problems by writing functions. It's important for this to be like second nature, so let's get some more practice using it!

Investigate

Define the Compute functions on [The Design Recipe \(Page 32\)](#) and [The Design Recipe \(Page 33\)](#).

Synthesize

Did students find themselves getting faster at using the Design Recipe? Can students share any patterns they noticed, or shortcuts they used?

Chaining Methods

25 minutes

Overview

Students learn how to perform multiple table operations (sorting, filtering, building) in the same line of code.

Launch

Now that we are doing more sophisticated analyses, we might find ourselves writing the following code:

```
# get a table with the nametags of all the fixed animals, ordered by species
fixed = animals-table.filter(is-fixed)
fixed-with-nametags = fixed.build-column("tag", nametag)
result = fixed-with-nametags.order-by("species", true)
```

That's a lot of code, and it also requires us to come up with names for each intermediate step! Pyret allows table methods to be *chained together*, so that we can build, filter *and* order a Table in one shot. For example:

```
# get a table with the nametags of all the fixed animals, ordered by species
result = animals-table.build-column("label", nametag).filter(is-fixed).order-by("species", true)
```

This code takes the `animals-table`, and builds a new column. According to our Contracts Page, `.build-column` produces a new Table, and that's the Table whose `.filter` method we use. That method produces *yet another Table*, and we call that Table's `order-by` method. The Table that comes back from that is our final result.

Teaching Tip

Use different color markers to draw *nested boxes* around each part of the expression, showing where each Table came from.

It can be difficult to read code that has lots of method calls chained together, so we can add a line-break before each “`.`” to make it more readable. Here's the exact same code, written with each method on its own line:

```
# get a table with the nametags of all the fixed animals, order by species
animals-table
  .build-column("label", nametag)
  .filter(is-fixed)
  .order-by("species", true)
```

Order matters: Build, Filter, Order.

Suppose we want to build a column and then use it to filter our table. If we use the methods in the wrong order (trying to filter by a column that doesn't exist yet), we might wind up crashing the program. Even worse, the program might work, but produce results that are incorrect!

produce results that are incorrect:

Investigate

When chaining methods, it's important to build first, then filter, and then order.

How well do you know your table methods? Complete [Chaining Methods \(Page 34\)](#) and [Chaining Methods 2: Order Matters! \(Page 35\)](#) in your Student Workbook to find out.

Synthesize

As our analysis gets more complex, method chaining is a great way to keep the code simple. But complex analysis also has more room for mistakes, so it's critical to think carefully when we use it!