Solving Word Problems with the Design Recipe

(Also available in WeScheme)

Students are introduced to the Design Recipe as a scaffold for breaking down word problems into smaller steps. They apply the Design Recipe to fixing a file that launches a rocket!

Lesson Goals	Students will be able to: • Understand how to use the Design Recipe to break down word problems. • Create a strong purpose statement that details in their own words what the function should do.
Student-Facing Lesson Goals	• Let's write our own functions in Pyret! • Let's use the Design Recipe to write functions from word problems.
Prerequisites	• Simple Data Types • Contracts • Functions: Contracts, Examples & Definitions
Materials	• PDF of all Handouts and Page • Rocket Height Starter File • Lesson Slides • Printable Lesson Plan (a PDF of this web page)
Supplemental Materials	• Optional Project: Design Recipe Telephone [rubric] • Additional Printable Pages for Scaffolding and Practice • Collaboration Starter File - For use with Design Recipe Telephone Set 1

Key Points for the Facilitator	

- Code should be easy to read! There may be other people looking at your code who could use a hint about what it does, and even the person who wrote the code could benefit from a note here and there. ***Comments*** are parts of a program that the computer ignores - they are for human eyes only!
- The ***Purpose Statement*** is a structured way of restating the problem. In a computer program, it's written as a comment in the code - something the computer doesn't read.
- Remind students that the ***Domain*** and ***Range*** of a function must be a *set* of possible inputs and outputs. In math, some of these sets have shorthands like Integers, Rationals, etc. In programming, we have shorthands for ***data types*** like Number, String, Image, Boolean, etc.
- If students struggle with creating the examples, use "Where'd You Get That" to help students build up their understanding around the concept.

Glossary

comments :: messages in the code, generally ignored by the computer, to help people interacting with the code understand what it is doing

data type :: a way of classifying values, such as: Number, String, Image, Boolean, or any user-defined data structure

domain :: the type or set of inputs a function expects, i.e., the independent variable(s) that govern the output of the function

example :: shows the use of a function on specific inputs and the computation the function should perform on those inputs

function definition :: code that names a function, lists its variables, and states the expression to compute when the function is used

purpose statement :: a concise, detailed description of what a function does with its inputs

range :: the type or set of outputs that a function produces, i.e., the dependent variable(s)

variable :: a name or symbol that stands for some value or expression, often a value or expression that changes

The Design Recipe *10 minutes*

Overview

In this lesson students build on what they already know about multiple representations of functions (contracts, examples and definitions) to write purpose statements and gain fluency with the Design Recipe.

Launch

A word problem is a description of a situation, but seeing the math behind the words can be challenging!

In this lesson, students are going to learn a strategy for breaking down word problems, called the **Design Recipe**. They have actually seen most of the steps of the Design Recipe, but they *haven't* seen how to put them together. There's also one part of the Design Recipe that they haven't seen yet: *writing a purpose statement*.

On its own, a Contract is not enough information to generate examples and write a ***function definition***. For example, the Contract for `circle` says it needs a Number and two Strings, but that's not the whole story! We can't use negative numbers for the radius, we can only use `"solid"` or `"outline"` for the first String, and the last String has to be a color!

Contracts are great, but we need more detail! Programmers and Mathematicians alike find it helpful to restate a problem in their own words. After all, if you can't explain a problem to someone else, you probably don't understand it yourself!

- On Matching Word Problems and Purpose Statements, we see four word problems and four ***purpose statements***.
- Take 2 minutes to read them and see if you can find any that describe the same thing and should be matched to each other.
- What pairs did you come up with?
- What do you Notice about those purpose statements? Do they have anything in common?

Purpose statements should have enough details to allow us to write examples without looking at the word problem!

- Turn to Writing Examples from Purpose Statements, read the purpose statements, and write examples that satisfy each of the contracts and purpose statements.
- *Optional:* for more practice, complete Writing Examples from Purpose Statements (2).

A good ***purpose statement*** must have three things:

1. A description of what the function *consumes*
2. A description of what the function *produces*
3. All the *relevant information* about how to produce that output

Investigate

- Turn to Fixing Purpose Statements.
- ChatGPT (an AI chatbot) has produced a purpose statement for each word problem… but ran into some difficulties.
- Fix ChatGPT's purpose statements, and then identify: What important information was missing from each purpose statement that you would need to solve the problem? What extra information was included, that wasn't needed to solve the problem?

Synthesize

What are the important elements of purpose statements? Why are purpose statements useful?

The Design Recipe in your Classroom

The three steps of the Design Recipe are designed to mirror best practices that you may *already be using in your classroom*. It's merely a collection of those practices, assembled in a structured way with great care taken to connecting each practice with the others.

Writing the Contract and Purpose Statement is where students *understand* the word problem. If you have your students restate the problem in their own words, draw pictures, or underline the inputs and outputs in the word problem, *you're already using this practice!*

Writing examples and circling-and-labeling what changes is where students *apply* their understanding to concrete inputs. If you have your students work through some concrete examples before jumping straight to variables and formulas, and ask them "what's the rule?" or "what's the pattern?", *you're already using this practice!*.

Writing the definition is where students *formalize and abstract* this understanding to work with *any input*. This is where they identify the structure of the rule or pattern, independently of any specific inputs.

The order of the recipe is a recommendation based on 20+ years of research about what works for most students, but that doesn't mean this order works best for *every* student! Some may find it easier to work through a concrete example or two before thinking about Domain and Range, and there's nothing wrong with that. We encourage you to use the Recipe in your classroom as often as possible, teaching students to be flexible with the tools and representations it includes.

Rocket Height *25 minutes*

Overview

Students are given a non-working program, which uses a linear function to determine the height of a rocket after a given length of time. The "broken" code is provided to lower cognitive load, allowing students to focus on comprehension (reading the code) and making use of structure (identifying where it's broken).

Launch

There's a lot of buzz out there around Artificial Intelligence (A.I.) tools, which claim to be able to solve math problems, write essays, and even write code for us! A few students found some A.I. tools that claim to be able to write Pyret programs for them. They asked the A.I. to generate code that makes a rocket blast off, starting on the ground (height=0) when the rocket first blasts off (time=0). The A.I. wrote the program for them - but are they ready to hand it in to their teacher? How do they know if it really solved the problem correctly?

- Open the Rocket Height Starter File, and click "Run".
- What happens when you press the space bar?
  - *The seconds change, but the rocket doesn't move!*
- What were you expecting to happen?
  - *The rocket would move!*
- What happens when you press `b`?
  - *The seconds go back down, but the rocket height stays at 0*
- Is `rocket-height` working?
  - *No.*

Direct students to close the window with the rocket, so that they can see the code.

If there are examples for how the program *should* work, we can automatically detect when AI writes bad code for us. But we didn't get any warning here! Why not?

- Type `rocket-height(0)` into the Interactions Area.
- As the program is currently written, what happens when we give the `rocket-height` function an input of 0?
  - *It returns 0.*

- Is that what we want it to do?
  - *Yes!*
- As the program is currently written, what happens when we give the `rocket-height` function an input of 10?
  - *It returns 0.*
- Is that what we want it to do?
  - *No!*
- Why did the examples pass?
  - *The programmer only gave one example! We should always provide at least two examples. More complex functions will require us to think about what range of examples will be necessary to test that our function does what we want it to!*

We should always test a function definition against at least 2 examples!

Investigate

Let's use the Design Recipe to fix `rocket-height`, and get comfortable with writing ***purpose statements***.

Complete Word Problem: rocket-height.

As students work, circle the room and make sure that their ***purpose statements*** are strong enough that they could write ***examples*** without looking at the original word problem. Encourage students to circle what's changing in their examples and label with descriptive ***variables***.

- Once you've completed the Design Recipe page for `rocket-height`, type the code into Definitions Area, replacing any missing and incorrect code with your own.
- When it's working correctly, explore the other functions in the file.

For students needing more specific instructions about exploring the file, try the following:

- Remove the comment from before the `(blastoff rocket-height)` and test the program.
- Put the comment back in front of `(blastoff rocket-height)`, remove the comment from `(graph rocket-height)`, and test the program.
- Try out `(space rocket-height)`
- Try out `(everything rocket-height)`

Teacher Tool: "Where'd You Get That?"

This is a powerful tool that forces students to explain their thinking, making deeper connections between steps, and helps teachers guide students to find their own mistakes. It requires two people: the Challenger, and the Defender. Most of the time, the teacher is in the role of Challenger.

The Challenger starts at the **bottom** of the page, physically pointing to one part of the *Definition* and asking "Where'd you get that?" The Defender has to *physically point* somewhere in the Examples, and explain what they're pointing to supports their Definition.

Next, the Challenger starts asking about the *Examples* and the Defender needs to show how their *Contract and Purpose* support them. This is repeated for every other step in the recipe, as students work their way back to the original word problem:

- **Challenger** (pointing at the `seconds` variable in the Definition): Where'd you get that?
- **Defender** (pointing at label in the Examples): Well, I circled the parts of the Examples that change, and labeled them as "seconds".
- **Challenger** (pointing at the label): OK, but where did you get that label?
- **Defender** (pointing at Purpose Statement): I used "seconds" in the Purpose Statement.
- **Challenger** (pointing at Purpose Statement): Where'd you get that term?
- **Defender** (pointing to Word Problem): I got it from reading the Word Problem.

***Optional:* For teachers who cover quadratic and exponential functions, or have students who need more of a challenge**, checkout the Rocket Height Challenges:

- **Changing slope:** Can you make the rocket fly faster? Slower?
- **Changing sign:** Can you make the rocket sink down instead of fly up?

- **Motivating Quadratic Functions:** Can you make the rocket *accelerate over time*, so that it moves faster the longer it flies?
- **Practicing the Quadratic Formula:** Can you make the rocket blast off *and then land again*?
- **More practice:** Can you make the rocket blast off, *reach a maximum height of exactly 1000 meters*, and then land?
- **More practice:** Can you make the rocket blast off, reach a maximum height of exactly 1000 meters, and then land after exactly 100 seconds?
- **Motivating Exponential Functions:** Can you make the rocket fly to the edge of the the universe?

Synthesize

Even great programmers make mistakes sometimes. And in a world where AI is being used more and more, it's critical that we be able to write examples so that we can detect when AI messes up!

- What problems did you fix in AI's code?
- What did the other functions do?
- Which step in the Design Recipe are you feeling the most confident about? The least? *At this stage, it is normal for students to feel most confident about the Contract and Examples, and the least confident about Purpose Statements and Definitions.*

Project Idea: Design Recipe Telephone

Most computer programs are written by huge teams! It is critical that each team member records their thinking with enough detail for other team members to be able to pick up where they left off. In *Optional Project:* Design Recipe Telephone *[rubric]*, students collaborate to complete a series of Design Recipe Problems, with each student being responsible for only one part of each problem in the set.

You can use any word problems you like, but we have provided two sets that lend themselves particularly well to the activity. One set can be used to collaboratively update the functions in Collaboration Starter File - For use with Design Recipe Telephone Set 1, which generates a cool mystery image if all three problems are solved correctly!

Additional Exercises

For more practice connecting Examples and Contracts, have students complete Writing Examples from Purpose Statements (2).

While most problems in a math book ask students to *solve* something, the actual challenge is figuring out *what the equation is that needs to be solved:* setting it up is where the thinking happens, and solving it is just arithmentic.

You can find blank Design Recipes at the back of the book, or print additional ones of your own.

Optional: Ask students to create their own appropriately challenging word problem (with a solution) and collect the responses for later use as "Do Now" tasks or formative assessment.

To help you apply the Design Recipe to more of your scope and sequence, we've provided a library of Design Recipe worksheets which connect to various curricular goals. We hope that you will be inspired by this library, and begin using the Design Recipe with more of the problems in your book!

- The Design Recipe (Restaurants)
- The Design Recipe (Direct Variation)
- The Design Recipe (Slope/Intercept)
- The Design Recipe (Negative Slope/Intercept)
- The Design Recipe (Geometry - Rectangles)
- The Design Recipe (Geometry - Rectangular Prisms)
- The Design Recipe (Geometry - Circles)
- The Design Recipe (Geometry - Cylinders)
- The Design Recipe (Breaking Even)
- The Design Recipe (Marquee & Cubing)