# Function Composition

(Also available for WeScheme)

Students encounter new image transformation functions and strengthen their understanding of Circles of Evaluation by using functions within other functions.

| | |
|---|---|
| **Lesson Goals** | Students will be able to:<br>• Use functions as building-blocks, composing them to achieve a desired affect<br>• Diagram function composition using the Circles of Evaluation<br>• Compose functions when programming |
| **Student-facing Goals** | • I can map a path from one number to another by composing functions<br>• I can use Circles of Evaluation to show how functions can be composed |
| **Materials** | • Lesson Slides<br>• Function Cards - print and cut<br>• Diagramming Function Composition (Page 26)<br>• Function Composition — Green Star (Page 27)<br>• Function Composition — Your Name (Page 28)<br>• Function Composition — scale-xy (Page 29)<br>• More than one way to Compose an Image! (Page 30)<br>• *Optional: Function Composition Matching Activity (Desmos)* |
| **Supplemental Resources** | • Random Integer Generator<br>• Circles of Evaluation Review - Blank Template |
| **Preparation** | • Make sure all materials have been gathered<br>• Decide how students will be grouped in pairs |
| **Key Points For The Facilitator** | • Check frequently for understanding of ***data types*** and ***contracts*** during this lesson and throughout subsequent lessons. |

| | |
|---|---|
| | • When students encounter errors, encourage them to check their Contracts page and show their work using Circles of Evaluation.<br>• Students will use their Contracts page frequently, so it should be kept in an accessible, convenient location. |

**Language Table**

| Types | Functions | Values |
|---|---|---|
| Number | `+`, `-`, `*`, `/`, `num-expt`, `num-sqr`, `num-sqrt` | `4`, `-1.2`, `2/3`, `pi` |
| String | `string-length`, `string-repeat`, `string-contains` | `"hello"`, `"91"` |
| Boolean | `<`, `<>`, `<=`, `>=`, `<`, `>`, `==`, `<>`, `>=` | `true`, `false` |
| Image | `star`, `triangle`, `circle`, `square`, `rhombus`, `ellipse`, `regular-polygon`, `radial-star` | |

Click here to see the prior unit-based version.

***Glossary***

**contract** :: a statement of the name, domain, and range of a function

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

**definitions area** :: the left-most text box in the Editor where definitions for values and functions are written

**image** :: a type of data for pictures

**interactions area** :: the right-most text box in the Editor, where expressions are entered to evaluate

## Composing Functions 10 minutes

Students are given a scaffolded activity that forces them to use the output of one function as the input to another - to *compose* them.

### Launch

Divide students into groups of 3-4, and distribute a set of Function Cards to each group. Write down pairs of integers on the board, representing the "starting numbers" and "ending numbers". These integers should range from -50 to +50, but you can change the difficulty of the activity by making that span wider (more difficult) or more narrow (less difficulty). You can find a random integer generator here.

- Each group has a set of functions, each of which takes an input and produces an output. I can start with the number `4`, for example, and give it to the function `add6`. What will the output be? *10*
- I can also *compose* functions, meaning that the output of one is immediately passed into another. For example, I could compose `add6` and `double`, so the `10` gets passed into the next function, and doubled to produce `20`. What would happen if I composed `add6` with `double` *and* with `half`? *10*
- For each of the starting numbers on the board, your job is to figure out which functions to compose in order to get to the end.
- You will need to use some functions more than once, and that’s ok!

### Investigate

Give students time to experiment with this. You can make the activity more challenging by asking them to find the *shortest path* from start to end, using the smallest number of compositions.

### Synthesize

If two groups come up with different compositions that achieve the same end result, have them share their ideas!

# Diagramming Function Composition

## *Overview*

The Circles of Evaluation are extended to provide a visual-spatial metaphor for function composition, in addition to Order of Operations.

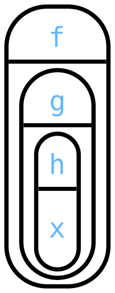
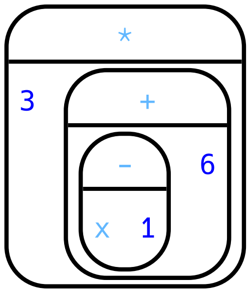
## *Launch*

Three of the function cards we just used were for the functions `f`, `g` and `h`:

- `f` multiplied its input by 3
- `g` added six to its input
- `h` subtracted one from its input

We can compose those function in any order. If we composed them as `f(g(h(x)))` and evaluated them for `x = 4` what would happen?

We can diagram the function composition using Circles of Evaluation (see first column, below). In the second column, we've replaced the function names in each Circle of Evaluation with *what each function does*:

| Function Composition | Order of Operations |
|---|---|
|  |  |

The circles show us that in order to evaluate $f(g(h(4)))$

- First we would have to evaluate $h(4)$, subtracting `1` from `4` to get `3`
- Then we would evaluate $g(3)$, adding `6` to `3` to get `9`
- Then we would evaluate $f(27)$, tripling `9` to get `27`

## *Investigate*

Have students turn to Diagramming Function Composition (Page 26) to practice writing, translating and evaluating Circles of Evaluation of composed functions.

## *Synthesize*

Do $f(g(h(x)))$ and $g(h(f(x)))$ evaluate to the same thing? *No!*

Why not? *order matters!*

## Composing Functions in Code 20 minutes

### *Overview*

The Circles of Evaluation are extended to functions that do more than compute values.

### *Launch*

The contracts page in your workbook is just like the Function Cards from this activity. Your job as a programmer is to figure out how to compose those functions to get where you want to go, in the most clever or elegant way possible.

### *Investigate*

Have students log into code.pyret.org (CPO) open a new program and save it as Function Composition. Have students open to Function Composition — Green Star (Page 27), in which they will be drawing circles of evaluation to help them write expressions to compose a series of images.

- Make sure students are using the ***Definitions area*** (left side) for code they want to keep and are using the ***Interactions area*** (right side) to test code or try out new ideas.
- When students are finished, check their work, and ask them to change the color of all of the stars to "gold" or another color of your choosing.

Then have students open to Function Composition — Your Name (Page 28) in which they will create a text ***image*** of their name and experiment with other functions.

#### Strategies for Facilitation

While students are exploring, be available for support but encourage student discussion to solve problems. Many student questions can be addressed with these responses: *Did you try drawing the Circle of Evaluation first? Did you check the contract? Have you pressed the Run button to save your Definitions changes?*

Encourage students to practice writing comments in the code to describe what is being produced, using # at the beginning of the line.

If you have time, you can also have students work with Function Composition — scale-xy (Page 29) and/or Function Composition Matching Activity (Desmos)

## Synthesize

- What do all of these functions have in common? *They all produce images, they all change some element of the original image*
- Does using one of these functions change the original image? *No, it creates a whole new image*
- What does the number in `scale` represent? *The scale factor by which the image should grow or shrink*
- What does the number in `rotate` represent? *The rotation angle, measured counterclockwise*
- The Domain and Range for `flip-horizontal` is Image -> Image. Why can I use the output of the `text` function as an *input* for `flip-horizontal`? *Because the `text` function produces an Image, which is then used as the input for `flip-horizontal`.*

### Strategies for English Language Learners

MLR 1 - Stronger and Clearer Each Time: As an alternative, display the discussion questions during the last 5 minutes of the Explore and ask students to discuss the questions with their partner, asking each other for explanation and details and coming up with the clearest, most precise answer they can. Student pairs can then share with another pair and compare their responses before moving into a full class discussion.

### Fun with Images!

Now that students have learned how to use all of these image-composing functions, you may want to give them a chance to create a design of their own, tasking them with using at least 4 functions to create an image of their choosing.
Our Flags lesson also dives deeper into image composition.

## Composing Multiple Ways Optional

### Overview

Students identify multiple expressions that will create the same image, and think about the merits of one expression over another.

### Launch

As is often true with solving math problems, there is more than one way to get the same composed image.
Suppose I wrote the code: `scale(3, star(50, "solid", "red"))`.
What's another line of code I could write that would produce the exact same image? `star(150, "solid", "red")`

### Investigate

Students complete More than one way to Compose an Image! (Page 30).

### Synthesize

There is a special function in code.pyret.org (CPO) that let's us test whether or not two images are equal.
`is-image= :: Image, Image -> Boolean`
Use it to test whether all of the expressions you wrote successfully build the same images.

- Could we have written more expressions to create the same images?
- Are all of the ways to write the code equally efficient?

---