Order of Operations

(Also available for Pyret)

Students learn to model arithmetic expressions with a visual tool for order of operations, known as "Circles of Evaluation".

Prerequisites	Coordinates and Game Design
Relevant Standards CC-Math	*Select one or more standards from the menu on the left (⌘-click on Mac, Ctrl-click elsewhere).*
Lesson Goals	Students will be able to: • Model an arithmetic expression using ***Circles of Evaluation***. • Translate Circles of Evaluation into code.
Student-facing Goals	• I can write Circles of Evaluation for a given arithmetic ***expression***. • I can translate a Circle of Evaluation model into code. • I can use numbers and operations in a programming environment.
Materials	• Lesson slides (Google Slides) • Circles of Evaluation template (Google Doc) • Circles of Evaluation Mixed Review (original (Page 13), solution) • Circles of Evaluation with Square Roots (original (Page 16), solution) • Multiple Representations - Order of Operations (PDF)
Preparation	• Make sure all materials have been gathered
Supplemental Resources	• Coordinates, Circles of Evaluation, and Code (Quizizz) • Order of Operations (Quizizz) • Data Types & Circles of Evaluation (Desmos Activity) • Circles of Evaluation Review - Blank Template (Desmos Activity) • Circles of Evaluation Review - Scaffolded (Desmos Activity) • Data Types, Circles of Evaluation, and Contracts (Desmos Activity)

Key Points For The Facilitator

- Error messages are the computer trying to give us a clue that something is wrong. Model reacting to ***error message***s with interest to demonstrate to students that the messages are a helpful tool.
- After the first few exercises in creating Circles of Evaluation, ask students whether they create them from the 'inside-out' (drawing the inner circles first) or from the 'outside-in.' After they’ve given their responses, have them try using the OTHER way!
- Up until now, we didn’t have a visual spatial model for explaining the order of operations. Ask students to compare Circles of Evaluation to previous methods they’ve learned (PEMDAS, GEMAS, etc)
- For a memory hook, model the "bug that crawls through the circle" explanation.
- Students may benefit from using multiple colors to distinguish between the different smaller expressions and parentheses.

For a textbook-like version of materials similar to these, you may wish to see the prior unit-based version.

Glossary

circle of evaluation :: a diagram of the structure of an expression (arithmetic or code)
definitions area :: the left-most text box in the Editor where definitions for values and functions are written
editor :: software in which you can write and evaluate code
error message :: information from the computer about errors in code
expression :: a computation written in the rules of some language (such as arithmetic, code, or a Circle of Evaluation)
function :: a mathematical object that consumes inputs and produces an output
interactions area :: the right-most text box in the Editor, where expressions are entered to evaluate
value :: a specific piece of data, like 5 or "hello"

Warmup

Students should open WeScheme in their browser, and click "Log In". This will ask them to log in with a valid Google account (Gmail, Google Classroom, YouTube, etc.), and then show them the "My Programs" page. This page is empty - they don’t have any programs yet! Have them click "Start a New Program".

Numbers

10 minutes

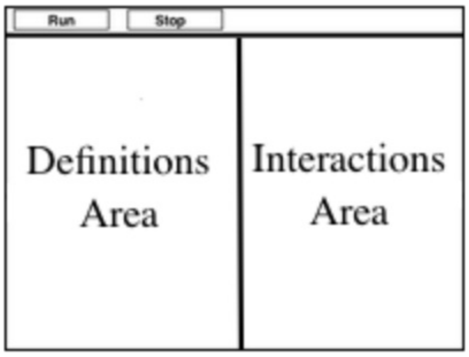
Overview

Students experiment with the Editor, exploring the different kinds of numbers and how they behave in this programming language.

Launch

This screen is called the ***Editor***, and it looks something like the diagram you see here. There are a few buttons at the top, but most of the screen is taken up by two large boxes: the ***Definitions Area*** on the left and the ***Interactions Area*** on the right.

The ***Definitions Area*** is where programmers define values and functions that they want to keep, while the ***Interactions Area*** allows them to experiment with those values and functions. This is like writing function definitions on a blackboard, and having students use those functions to compute answers on scrap paper.



For now, we will only be writing programs in the **Interactions Area** on the right.

Investigate

Math is a language, just like English, Spanish, or any other language. We use nouns, like "bread", "tomato", "mustard" and "cheese" to describe physical objects. Math has ***values***, like the numbers 1, 2 or 3, to describe quantities.

Try typing the number `42` on the right, and then hitting "Enter" or "Return". What did this number evaluate to? (Hint: Numbers should evaluate to themselves - if you didn't get back the same number you put in, something is very wrong!) If working in pairs, make sure you each take a turn at the keyboard. Suggestions:

- How *large* of a number can you enter?
- How *small* of a number can you enter?
- What happens if you type two numbers on the same line?
- Do fractions work? Decimals?
- Do negative numbers work?

Remember, we're only trying *numbers* for now, not operations like $3 - 6$, $\sqrt{16}$ or 4^2

Notice & Wonder

In pairs, students will each try entering a variety of numbers in the Interactions Area, hitting "Enter" each time to see what the computer does. Then they will write down what they Notice and Wonder on Notice and Wonder (Page 8).

- What did you Notice? What do you Wonder?
- Did you get any error messages? If so, read it carefully - what do you think it means?

Student Misconceptions

- Students who try division by writing `3/2` and get an answer may falsely assume that they've performed division. In fact, what they've done is entered a *rational number*. ("Two-thirds" is *equivalent* to the expression "two divided by three", but only insofar as they result in the same value. "2" is equivelent to expression "10 minus 8", for the same reason!)
- Rational numbers can be converted back and forth between fraction and decimal forms by clicking on them.

Synthesize

Our programming language knows about many types of numbers, and they behave pretty much the way they do in math. Our Editor is also pretty smart, and can automatically switch between showing a rational number as a fraction or a decimal, just by clicking on it!

Order of Operations

30 minutes

Overview

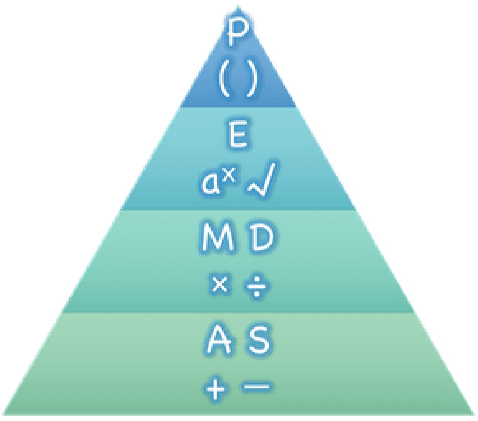
Students are given a challenging expression that exposes common misconceptions about order of operations. The goal is to demonstrate that a brittle, fixed notion of order of operations is *not good enough*, and lead students to a deeper understanding of Order of Operations as a grammatical device. The Circles of Evaluation are introduced as "sentence diagramming for arithmetic".

Launch

Humans also use verbs like "throw", "run", "build" and "jump" to describe operations on these nouns. Mathematics has ***functions*** - or "operations" - like addition and subtraction, which are operations performed on values. Just as you can " **spread** *mustard* on *bread* ", a person can also " **add** *four* and *five* ".

A mathematical expression is like a sentence: it's an instruction for doing something. The expression 4+5 tells us to add 4 and 5. To evaluate an expression, we follow the instructions in the expression. The expression $4 + 5$ evaluates to 9.

Sometimes, we need multiple expressions to accomplish a task, and it will matter in which order they come. For exmple, if you were to write instructions for making a sandwich, it would matter very much which instruction came first: melting the cheese, slicing the bread, spreading the mustard, etc. The order of functions matters in mathematics, too.



Mathematicians didn't always agree on the order of operations, but now we have a common set of rules for how to evaluate expressions. The pyramid on the right summarizes the order. When evaluating an expression, we begin by applying the operations written at the top of the pyramid (multiplication and division). Only after we have completed all of those operations can we move down to the lower level. If both operations are present (as in $4 + 2 - 1$), we read the expression from left to right, applying the operations in the order in which they appear.

But this set of rules is brittle, and doesn't always make it clear what we need to do. Check out the expression below. What do you think the answer is? This math problem went viral on social media recently, with math teachers arguing about what the answer was! Why might they disagree on the solution?

$$6 \div 2(1 + 2)$$

Order of Operations mneumonic devices like PEMDAS, GEMDAS, etc focus on how to get the answer. What we need is a *better way to read math* .

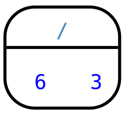
Instead of a rule for computing answers, let's start by diagramming the math itself! We can *draw the structure* of this grammer in mathematics using something called the **Circles of Evaluation** . The rules are simple:

1) Every Circle must have one - and only one! - function, written at the top

That means that Numbers (e.g. - `3` , `-29` , `77.01` ...) are still written by themselves. It's only when we want to *do something* like add, subtract, etc. that we need to draw a Circle.

2) The inputs to the function are written left-to-right, in the middle of the Circle.

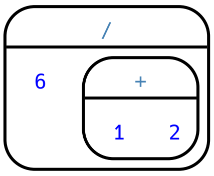
If we want to draw the Circle of Evaluation for $6 \div 3$, the division function (`/`) is written at the top, with the `6` on the left and the `3` on the right.



What if we want to use multiple functions? How would we draw the Circle of Evaluation for $6 \div (1+2)$? Drawing the Circle of Evaluation for the $1+2$ is easy. But how do divide 6 by that circle?

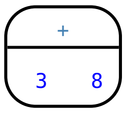
Circles can contain other Circles

We basically replace the `3` from our earlier Circle of Evaluation with *another* Circle, which adds 1 and 2!

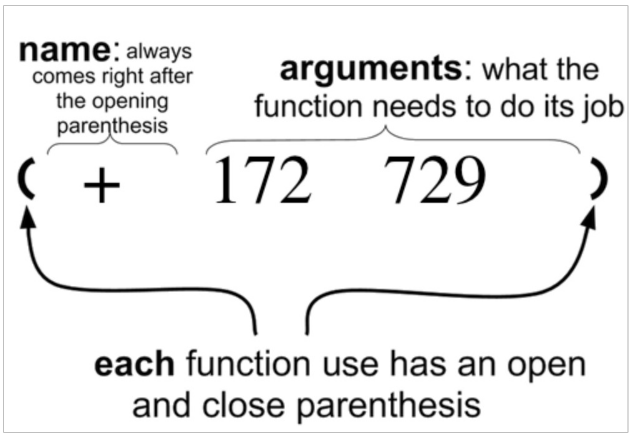


Circles of Evaluation *help us write code*

When converting a Circle of Evaluation to code, it's useful to imagine a spider crawling through the circle from the left and exiting on the right. The first thing the spider does is cross over a curved line (an open parenthesis!), then visit the operation - also called the *function* - at the top. After that, she crawls from left to right, visiting each of the inputs to the function. Finally, she has to leave the circle by crossing another curved line (a close parenthesis).

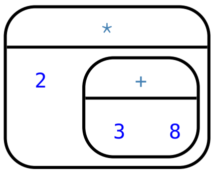
Expression	→	$3+8$
Circle of Evaluation	→	
Code	→	`(+ 3 8)`

All of the expressions that follow the function name are called arguments to the function. The following diagram summarizes the shape of an expression that uses a function.


name: always comes right after the opening parenthesis
arguments: what the function needs to do its job
(+ 172 729)
each function use has an open and close parenthesis

Practice creating Circles of Evaluation using the common operators (`+` , `-` , `*` , `/`).

- Do spaces matter when typing in functions?
- Does the order of the numbers matter in the functions? Which functions?
- What do the error messages tell us?
- What connections do you see between the expression, circle, and code?

Expression	→	$2 \times (3+8)$
Circle of Evaluation	→	
Code	→	`(* 2 (+ 3 8))`

- Why are there two closing parentheses in a row, at the end of the code?

- If an expression has three sets of parentheses, how many Circles of Evaluation do you expect to need?

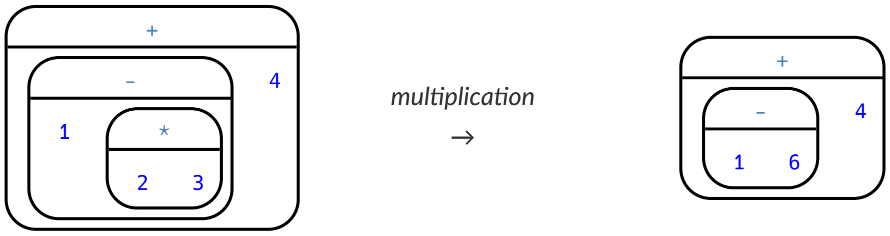
Circles of Evaluation ***help us get the correct answer***

Aside from helping us catch mistakes before they happen, Circles of Evaluation are also a useful way to think about *transformation* in mathematics. For example, you may have heard that "any subtraction can be transformed to a negative addition." For example, $1 - 2$ can be transformed to $1 + (-2)$.

Suppose someone tells you that $1 - 2 * 3 + 4$ can be rewritten as $1 + (-2) * 3 + 4$. These two expressions will definitely give us the same answer, but this transformation is actually *incorrect*! It doesn't not use the negative addition rule at all! **Take a moment to think: what's the problem?**

We can use the Circles of Evaluation to figure it out!

The first Circle is just the original expression. The multiplication happens first, so let's see how multiplication changes this circle:



As you can see, replacing the subtraction with a negative addition happens to the *result* of the multiplication. We can't actually change the `2` into a `-2`, because it isn't actually being subtracted from `1`!

Sure, we got the same answer - but that doesn't mean the way we got it was correct. If all that mattered was getting the right answer, we could just as easily have replaced the whole expression with $5 - 6$. And that is *definitely* not a correct transformation!

Any time you make a transformation in math (replacing $10 - 2$ with 8 because of subtraction, or replacing $2 + 6$ with $6 + 2$ because of commutativity), you need to make sure the transformation is *correct*. The Circles of Evaluation help us see these transformation *visually*, rather than forcing us to keep them in our heads.

Circles of Evaluation

The Circles of Evaluation are a critical pedagogical tool in this course. They place the focus on the *structure* of mathematical expressions, as a means of combating the harmful student belief that the only thing that matters is the *answer*. They can be used to diagram arithmetic sentences to expose common misconceptions about Order of Operations, and make an excellent scaffold for tracing mistakes when a student applies the Order of Operations incorrectly. They are also a bridge representation, which naturally connects to function composition and converting arithmetic into code.

Investigate

- Students complete Arithmetic Expressions to Circles of Evaluation & Code (Page 15) page in their workbook. They should *draw all of the Circles first* and check their work, before converting to code.
- Students complete the Translating Circles of Evaluation to Code (Page 13).
- If time allows, partners should take turns entering the code into the editor.

The Circles of Evaluation are a great way to visualize *other* functions you already know, such as square and square root! **Note:** In WeScheme, we use `sqrt` as the name of the square root function, and `sqr` as the function that squares its input.

- Students complete Translating Circles of Evaluation to Code - w/Square Roots (Page 16) with their partners and test their code in the editor.
- OPTIONAL: Using this graphic organizer, (1) create the code that represents this Circle of Evaluation, (2) translate this into code, (3) evaluate the expression using the order of operations, and (4) then compare and contrast the three methods.

Strategies For English Language Learners

MLR 7 - Compare and Connect: Gather students' graphic organizers to highlight and analyze a few of them as a class, asking students to compare and connect different representations.

Closing

Have students share back what they learned from the Circles of Evaluation. You may want to assign traditional Order of Operations problems from your math book, but instead of asking them simply to compute the answer - or even list the steps - have them *draw the circle* .

Additional Exercises

- Completing Circles of Evaluation from Math Expressions (1) (original , answers)
- Completing Circles of Evaluation from Math Expressions (2) (original (Page 9) , answers (Page 9))
- Creating Circles of Evaluation from Math Expressions (1) (original , answers)
- Creating Circles of Evaluation from Math Expressions (2) (original , answers)
- Creating Circles of Evaluation from Math Expressions (3) (original (Page 10) , answers (Page 10))
- Converting Circles of Evaluation to Math Expressions (1) (original , answers)
- Converting Circles of Evaluation to Math Expressions (2) (original , answers)
- Matching Circles of Evaluation and Math Expressions (original (Page 11) , answers (Page 11))
- Evaluating Circles of Evaluation (1) (original , answers)
- Evaluating Circles of Evaluation (2) (original , answers)
- Completing Code from Circles of Evaluation (original (Page 12) , answers (Page 12))
- Converting Circles of Evaluation to Code (1) (original , answers)
- Converting Circles of Evaluation to Code (2) (original , answers)
- Matching Circles of Evaluation and Code (original (Page 14) , answers (Page 14))