Arithmetic Expressions

Now that you've written a few programs, let's take a step back and discuss how to do arithmetic. The behavior of Python operators (+, -, *, /) depends on what type of data you have.

Content Learning Objectives

After completing this activity, students should be able to:

- Execute mathematical expressions similar to a calculator.
- Describe the function of the three Python division operators.
- Explain differences between integer and floating-point data.

Process Skill Goals

During the activity, students should make progress toward:

- Recognizing mathematical operations based on tables. (Information Processing)

Facilitation Notes

Help students get started quickly on the table in Model 1, so that they will have enough time to answer the questions. When reporting out, ask teams what surprises they had and what they learned from interacting with the code using a Shell.

If you are using Thonny (see Activity 1), now is a good opportunity to explain the "Assistant" that opens when errors occur. It may also be helpful to visualize assignment statements using the debugger. Thonny will even step into expressions, showing the order of operations. Have students check their work as you demonstrate the order and result of each expression.

On Model 2, report out after #15. Introduce the terms ***modulo operation*** (and/or ***modulus***) and ***remainder operator***. Explain that "x % y" can be pronounced "x mod y". #16 refers to the instructor hypothetically giving mints to teams to divide among themselves. Consider bringing real mints to build rapport with the class. Or you can use other small objects, like paperclips.

During Model 3, be aware of the difference between `**`, `pow`, and `math.pow`. The ** operator and built-in `pow` function preserve the data type of the operands: `5 ** 2` is 25. In contrast, `math.pow` always uses floating-point arithmetic: `math.pow(5, 2)` is 25.0. If students use `math.pow` on #24, they'll get the incorrect answer if it overflows. You can either warn students about `math.pow` in advance, or have students discover this issue when reporting out.

Model 1 Python Calculator

In a Python Shell window, ">>>" is a ***prompt*** indicating that the interpreter is waiting for input. All text entered after the prompt will be executed immediately as Python code.

If you type a Python ***expression*** (code that results in a value) after the prompt, Python will show the value of that expression, similar to a calculator. You can use Python's `math` module to perform more complex mathematical operations like logarithms and trigonometric operations.

Do not type anything yet! Read the questions first!

Python code	Predicted output	Actual output
`2 + 3`		5
`3 * 4 + 2`		14
`3 * 4 + 2.0`		14.0
`3(4 + 2)`		TypeError
`3 * (4 + 2)`		18
`5 / 10`		0.5
`5 / 10.0`		0.5
`5 / 9`		0.5555556
`2 ** 4`		16
`abs(-2) ** 4`		16
`math.pow(2, 4)`		NameError
`import math`		
`math.pow(2, 4)`		1.0
`sqrt(4)`		NameError
`math.sqrt(4)`		2.0
`math.cos(0)`		1.0
`math.pi`		3.141592653589793
`math.sin(math.pi / 2)`		1.0

Questions (15 min) Start time: _____

1. In the middle "Predicted output" column, write what value you expect will be displayed, based on your team's experience using a calculator. If there are any lines you are not confident about, place an asterisk next to your predicted output.

2. Open a Python Shell on your computer. Type each Python expression at the prompt, one line at a time, and write the corresponding Python output in the third column above. If an error occurs, write what type of error it was (i.e., the first word of the last line of the error message).

3. What does the ** operator do?

It raises a number to a power.

4. Based on the Python code in Model 1, identify four examples of:

a) mathematical operator +, *, /, **

b) mathematical function abs, math.pow, math.sqrt, math.cos, math.sin

5. For addition and multiplication to produce an output with a decimal value, what type of number must be part of the input? Provide justification for your team's answer.

At least one of the numbers must have a decimal value. For example, `3 * 4 + 2` is the integer value 14, but `3 * 4 + 2.0` is the decimal value 14.0.

6. Does division follow the same rule as in #5? Provide justification for your team's answer.

No; when dividing integers, the result is always a decimal number. The same is true even when there is no remainder, i.e., 8 / 4 is 2.0.

7. The output of Model 1 displayed three different errors. Explain the reason for each:

a) TypeError need to use * to multiply [1]

b) 1st NameError need to import math

c) 2nd NameError need "math." before function

8. Identify two differences between using a Python built-in function (e.g., `abs`) and a function from the `math` module.

Need to "`import math`" first, and all function names start with "`math.`" before the function.

[1] When students enter `3(4 + 2)` the interpreter says, "TypeError: 'int' object is not callable". In other words, the integer 3 is not a function and can't be called.

Model 2 Dividing Numbers

Table A

`9 / 4`	*evaluates to*	2.25
`10 / 4`	*evaluates to*	2.5
`11 / 4`	*evaluates to*	2.75
`12 / 4`	*evaluates to*	3.0
`13 / 4`	*evaluates to*	3.25
`14 / 4`	*evaluates to*	3.5
`15 / 4`	*evaluates to*	3.75
`16 / 4`	*evaluates to*	4.0

Table B

`9 // 4`	*evaluates to*	2
`10 // 4`	*evaluates to*	2
`11 // 4`	*evaluates to*	2
`12 // 4`	*evaluates to*	3
`13 // 4`	*evaluates to*	3
`14 // 4`	*evaluates to*	3
`15 // 4`	*evaluates to*	3
`16 // 4`	*evaluates to*	4

Table C

`9 % 4`	*evaluates to*	1
`10 % 4`	*evaluates to*	2
`11 % 4`	*evaluates to*	3
`12 % 4`	*evaluates to*	0
`13 % 4`	*evaluates to*	1
`14 % 4`	*evaluates to*	2
`15 % 4`	*evaluates to*	3
`16 % 4`	*evaluates to*	0

Questions (15 min)

Start time: _____

9. For each operator in Model 2, identify the symbol and describe the type of numerical result.

/ decimal // integer % integer

Note that Python refers to decimal numbers as "floating-point" numbers.

10. If the result of the / operator were rounded to the nearest integer, would this be the same as the result of the // operator? Explain how the results in Table A compare to Table B.

No, the pattern is off by two rows. The 0.5 and 0.75 values would round up, but in the second table they round down.

11. If the table included more rows, list all numbers // 4 would evaluate to 2 and all the numbers // 4 would evaluate to 4.

8, 9, 10, and 11 evaluate to 2.
16, 17, 18, and 19 evaluate to 4.

12. Based on the results of Table C, propose another number % 4 evaluates to 0, and explain what all these numbers have in common.

Other numbers include 0, 4, 8, 20, 24. All of these numbers are multiples of four.

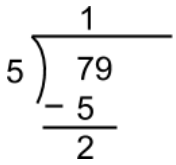
13. Consider the expressions in Table C that evaluate to 1. How do the left ***operands*** in these expressions (i.e., 9, 13) differ from those that evaluate to 0?

They each differ by one; they are one higher than a multiple of four.

14. Describe the reason for the repeated sequence of numbers (0, 1, 2, 3) for the result of % 4.

The difference (remainder) increases by one until the number is exactly divisible by 4.

15. Recall how you learned to do long division in elementary school. Finish solving for $79 \div 5$ below. Which part of the answer is `79 // 5`, and which part is `79 % 5`?



We first bring the 9 down, then 5 goes into 29 five times, and so we subtract 25. The final answer is 15 remainder 4. So `79 // 5` is 15, and `79 % 5` is 4.

16. Imagine that you are given candy mints to divide evenly among your team members.

a) If your team receives 11 mints, how many mints would each student get, and how many are left over? Write a Python expression to compute each result.

`11 / 3` is 3 and `11 % 3` is 2 or `11/ 4` is 2 and `11 % 4` is 3

b) If your team receives 2 mints, how many mints would each student get, and how many are left over? Write a Python expression to computes this result.

`2 / 3` is 0 and `2 % 3` is 2 or `2 / 4` is 0 and `2 % 4` is 2

17. Python has three division operators: "floor division", "remainder", and "true division". Which operator (symbol) corresponds to each name?

`//` is floor division, because it throws away the decimal place (i.e., it "floors" the result). `%` is the remainder operator, which is sometimes called the modulo operator. `/` is true division, because it gives you the mathematically correct answer.

Model 3 Integers and Floats

Every value in Python has a ***data type*** which determines what can be done with the data. Enter the following code, one line at a time, into a Python Shell. Record the output for each line (if any) in the second column.

Python code	Shell output
`integer = 3`	
`type(integer)`	<class 'int'>
`type("integer")`	<class 'str'>
`pi = 3.1415`	
`type(pi)`	<class 'float'>
`word = str(pi)`	
`word`	'3.1415'
`number = float(word)`	
`print(word * 2)`	3.14153.1415
`print(number * 2)`	6.283
`print(word + 2)`	TypeError
`print(number + 2)`	5.14159
`euler = 2.7182`	
`int(euler)`	2
`round(euler)`	3

Questions (15 min) Start time: _____

18. What is the data type (`int`, `float`, or `str`) of the following values? (Note: if you're unsure, use the `type` function in a Python Shell.)

a) pi float

b) integer int

c) word str

d) number float

19. List the function calls that convert a value to a new data type.

The calls are: `str(pi)`, `float(word)`, and `int(euler)`. Note there is a function named after each data type.

20. How does the behavior of the operators (+ and *) depend on the data type?

The + operator appends text, and the * operator copies text.

21. What is the difference between the `int` function and the `round` function?

The `int` function truncates the value, throwing away the decimal places. The `round` function rounds the value up or down to the nearest integer.

22. What is the value of `3 + 3 + 3`? What is the value of `.3 + .3 + .3`? If you enter these expressions into a Python Shell, what do you notice about the results?

Adding three 3's is 9, but adding three .3's is 0.8999999999999999. The answer is slightly off when using floating-point numbers.

23. In order to store a number with 100% accuracy, what data type is required? How might you precisely represent a bank account balance of $123.45?

Integers should be used to avoid floating-point errors. Simply multiply the balance by 100 when storing, and divide by 100 when displaying. $123.45 would be represented as 12345 cents.

24. Try calculating a very large integer in a Python Shell, for example, 123^{456}. Is there a limit to the integers that Python can handle?

There is no limit, other than the computer's memory size. But the larger the integer, the longer it takes to compute it.

25. Try calculating a very large floating-point number in a Python Shell, for example, 123.0^{465}. Is there a limit to the floating-point numbers that Python can handle?

Yes; at some point the numbers get too big. For example, `123.0 ** 456` results with an OverflowError: 'Numerical result out of range'.

26. Summarize the difference between the numeric data types (`int` and `float`). What are their pros and cons?

Integers have unlimited range and precision, but floating-point numbers are an approximation. (Note: `float` in Python is usually implemented using `double` in C.)