Arithmetic Expressions

Now that you've written some code, let's take a step back and look at some common arithmetic operators. The behavior of Python operators (+, -, *, /) depends on what type of data you have.

Model 1 Python Calculator

An ***expression*** is a unit of code that represents a value to be computed. For example, if `x` is 5, the expression `3 * x` evaluates to 15. The following table shows other examples of expressions.

Python code	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)`	16.0
`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. Individually, review each row of the table. Put an asterisk (*) next to each row that looks different from what you would expect, based on your experience using a calculator.

2. As a team, discuss each line for which anyone on the team put an asterisk. If you are unsure about the meaning of the code or the output, please ask for help.

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

a) mathematical operator

b) mathematical function

4. 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.

5. Does division follow the same rule as in #4? Provide justification for your team's answer.

6. What is the difference between the `**` operator and the `math.pow()` function?

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

a) TypeError

b) 1st NameError

c) 2nd NameError

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

Model 2 Order of Operations

Python follows the usual order for arithmetic operations. The following table lists operators from highest to lowest ***precedence***.

Operator	Description
`( )`	Parentheses
`**`	Exponentiation
`+ -`	Positive, Negative (***unary*** operators)
`* /`	Multiplication, Division
`+ -`	Addition, Subtraction (***binary*** operators)
`=`	Assignment

Questions (15 min) Start time:

9. Determine the order of operations in the statement: `y = 9 / 2`

a) First operator to be evaluated:
b) Second operator:
c) Value of `y`:

10. Determine the order of operations in the statement: `x = 5 * -3`

a) First operator to be evaluated:
b) Second operator:
c) Third operator:
d) Value of `x`:

11. Determine the order of operations in the statement: `z = 2 * 4 ** (3 + 1)`

a) First operator to be evaluated:
b) Second operator:
c) Third operator:
d) Fourth operator:
e) Value of `z`:

12. The + and - operators show up twice in the table of operator precedence. For the Python statement `x = 5 * -3`, explain how you know whether the - operator is being used as a unary or binary operator.

13. What do the words "unary" and "binary" mean in this context?

14. What operator has the lowest precedence? Why do you think Python is designed that way?

15. What operators have the highest precedence? Why do you think Python is designed that way?

16. Enter the expressions below into a Python Shell. Why are the results different? Explain your answer in terms of operator precedence.

- `-3 ** 2` Result:
- `(-3) ** 2` Result:

Model 3 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:

17. For each operator in Model 3, identify the symbol and describe the type of numerical result.

18. 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.

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

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

21. 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?

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

23. 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`?

$$\begin{array}{r} 1 \\ 5\overline{)\,79} \\ -\,5 \\ \hline 2 \end{array}$$

24. 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.

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.

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