2.7: Order of Operations

When more than one operator appears in an expression, the order of evaluation depends on the rules of precedence. For mathematical operators, Python follows mathematical convention. The acronym *PEMDAS* is a useful way to remember the rules:

- *Parentheses* have the highest precedence and can be used to force an expression to evaluate in the order you want. Since expressions in parentheses are evaluated first, \(2 \times (3-1)\) is 4, and \((1+1) \times (5-2)\) is 8. You can also use parentheses to make an expression easier to read, as in \((\text{minute} \times 100) / 60\), even if it doesn't change the result.

- *Exponentiation* has the next highest precedence, so \(2**1+1\) is 3, not 4, and \(3*1**3\) is 3, not 27.

- *Multiplication and Division* have the same precedence, which is higher than Addition and Subtraction, which also have the same precedence. So \(2*3-1\) is 5, not 4, and \(6+4/2\) is 8.0, not 5.

- *Operators with the same precedence* are evaluated from left to right. So the expression \(5-3-1\) is 1, not 3, because the \(5-3\) happens first and then \(1\) is subtracted from \(2\).

When in doubt, always put parentheses in your expressions to make sure the computations are performed in the order you intend.