3.1 Expressions and Statements

Constants

Constants' values do not change. They are fixed values. Constants can be numbers, letters, or strings. We can have numeric constants and string constants. String constants are usually enclosed within single-quotes or double-quotes. When we pass a constant to the print function, it prints the value of that constant to the console. For example, when we say ‘print(123)’ Python prints ‘123’ onto the console.

Variables

A variable is a named place in the memory where a programmer can store data. This stored data can be later retrieved using the variable name. Choosing a variable name is up to the programmer. You can even change the value of a variable at a later stage. This is where a variable is different from a constant. Look at the below example where we have the variables x and y and the values assigned are 25 and 50.

```
>>> x = 25
>>> y = 50
```

Later you can change the value of x to 200 for example. In this example, variables x and y carry numeric values.

Statements

We can identify different types of statements. The first one you see below is an assignment statement. There we assign the value ‘5’ to the variable called ‘x’. Secondly also we have an assignment statement with an expression. You see the numeric operator there, which is the addition of value 5 to the variable x. Thirdly, you see a print statement where we ask Python to print the value of variable ‘x’ to the console.

```
>>> x = 5        [Assignment Statement]
>>> x = x + 5    [Assignment with expression]
print(x)         [Print Statement]
```

Assignment Statement

In an assignment statement, on the left-hand side, we have the variable to store the result. On the right-hand side, we have the expression to be evaluated.

```
>>> x = x + 5
```

Numeric Expressions

Numeric expressions include numeric operators. You will notice familiar operations like addition, subtraction, and multiplication, also some not-so-familiar operator symbols. For example, the operator symbol for multiplication in Python is the asterisk. Slash for division, double asterisk for power notation and percentage symbol for the remainder, and so on.

\+ Addition
\- Subtraction
* Multiplication
/ Division
** Power
% Remainder

Operator Precedence

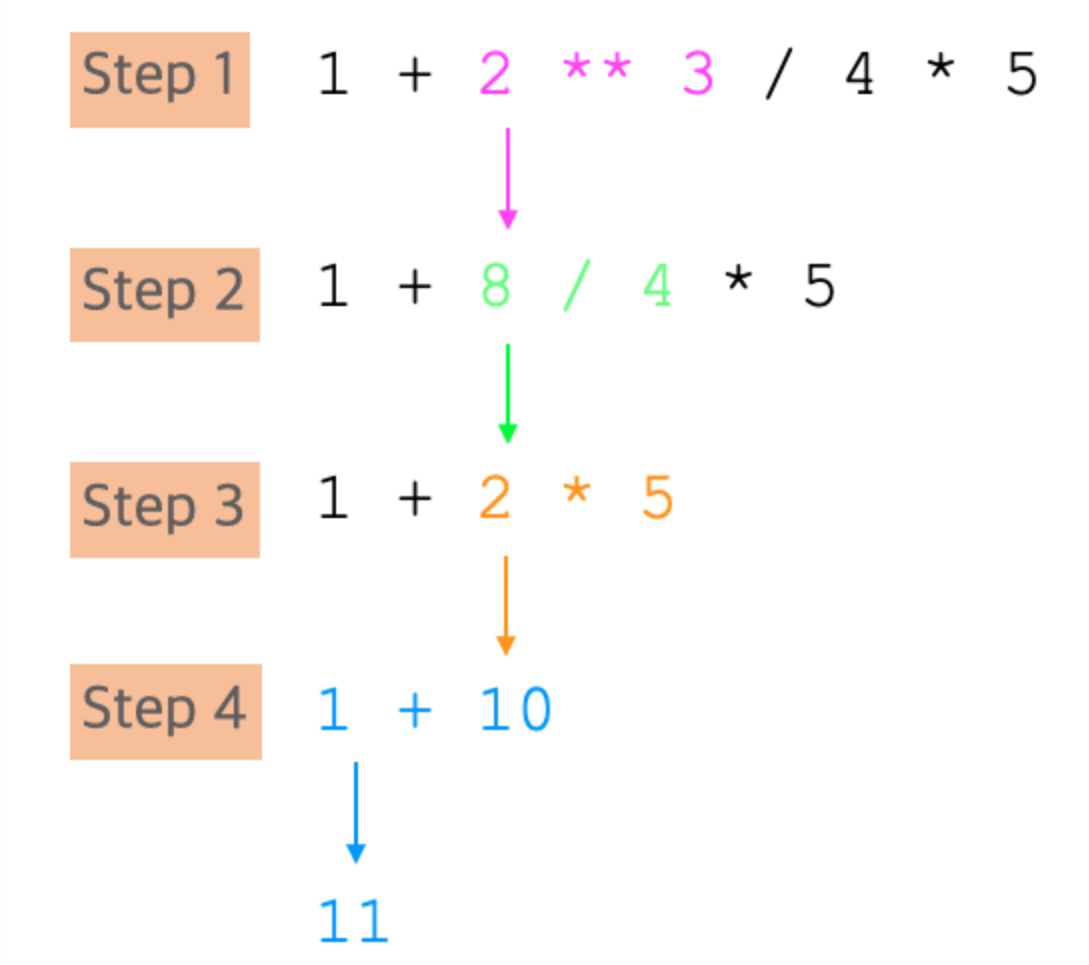
Now, what to do when there is more than one operator at the same time? This is a common issue even when it comes to regular mathematical operations. This is why we need to have an order of evaluation. This concept is called operator precedence. It helps us to decide which operation to do first and which one to do next and so on.

The rules can be identified as follows. Parenthesis should be evaluated first and then look at the exponentiation. Multiplication, division, and remainder operations come next. Then comes addition and subtraction. Left to right is the last in the list.

Parenthesis
Power
Multiplication
Addition
Left to Right

Let's look at an example.

We have no parenthesis here. Of course, if we had parenthesis this would have been less complicated. We do have exponentiation, that is two to the power three, denoted with a double asterisk. It should be evaluated first as it is the highest priority operation we find in this example. The result of step one is eight. Then in step two, we have both multiplication and division which are similar in priority level. So we apply the left-to-right rule and we find division before multiplication. So, in step two we divide eight by four resulting in two. Step 3 is straightforward, as we will do multiplication before addition, which results in ten. And at the last step, we only have addition to perform and the final result is eleven. Note that Python will do all these operations for you, in the correct order.



It is important to remember the rules from top to bottom. It is a good practice to use parenthesis when writing programs so as to minimize possible confusion. Keeping mathematical expressions simple is important so that they can be easily understood. One thing to recommend would be to break long mathematical operations into smaller, less complicated steps. That way, the program you write will be much more readable and easier for someone else to understand.

Further readinng: https://docs.python.org/3/reference/simple_stmts.html