

# **Python Operator Precedence and Associativity**

Riya Jacob K

Assistant Professor on Contract

Dept of Computer Applications

# How does the operator precedence work in Python?

- When we group a set of values, variables, operators or function calls that turn out as an expression.
- And once you execute that expression, Python interpreter evaluates it as a valid expression.
- See a simple example given below.
- ```
>>> 3 + 4  
7
```
- Here, the '3 +4' is a [Python expression](#).
- It contains one operator and two operands. However, a more complex statement can include multiple operators.

## Give examples of operator precedence in Python

- See the below example which combines multiple operators to form a compound expression.
- # Multiplication get evaluated before
- # the addition operation
- # Result: 17

**5 + 4 \* 3**

- However, it is possible to alter the evaluation order with the help of parentheses ().
- It can override the precedence of the arithmetic operators.
- # Parentheses () overriding the precedence of the arithmetic operators
- # Output: 27

**(5 + 4) \* 3**

# Operator precedence table in Python

| Operators      | Usage                        |
|----------------|------------------------------|
| { }            | Parentheses (grouping)       |
| f(args...)     | Function call                |
| x[index:index] | Slicing                      |
| x[index]       | Subscription                 |
| x.attribute    | Attribute reference          |
| **             | Exponent                     |
| ~x             | Bitwise not                  |
| +x, -x         | Positive, negative           |
| *, /, %        | Product, division, remainder |
| +, -           | Addition, subtraction        |
| <<, >>         | Shifts left/right            |
| &              | Bitwise AND                  |
| ^              | Bitwise XOR                  |
|                | Bitwise OR                   |

in, not in, is, is not, <, <=, >, >=,

is, !=, ==

Comparisons, membership, identity

not x

Boolean NOT

and

Boolean AND

or

Boolean OR

lambda

Lambda expression

# Python operator associativity

- In the above table, you can confirm that some of the groups have many operators. It means that all operators in a group are at the same precedence level.
- And whenever two or more operators have the same precedence, then associativity defines the order of operations.
- **What does the associativity mean in Python?**
- The associativity is the order in which Python evaluates an expression containing multiple operators of the same precedence.
- Almost all operators except the exponent (\*\*) support the left-to-right associativity.

# Give examples of associativity in Python

- For example, the product (\*) and the modulus (%) have the same precedence. So, if both appear in an expression, then the left one will get evaluated first.
- # Testing Left-right associativity  
# Result: 1  
**print(4 \* 7 % 3)**
- # Testing left-right associativity  
# Result: 0  
**print(2 \* (10 % 5))**
- As said earlier, the only operator which has right-to-left associativity in Python is the exponent (\*\*) operator.

- See the examples below.
- # Checking right-left associativity of \*\* exponent operator

# Output: 256

**print(4 \*\* 2 \*\* 2)**

- # Checking the right-left associativity  
# of \*\*

# Output: 256

**print((4 \*\* 2) \*\* 2)**

- You might have observed that the 'print(4 \*\* 2 \*\* 2)' is similar to '(4 \*\* 2 \*\* 2)'.



# Nonassociative operators

- **What are nonassociative operators in Python?**
- Python does have some operators such as assignment operators and comparison operators which don't support associativity.
- Instead, there are special rules for the ordering of this type of operator which can't be managed via associativity.
-

# Give examples of nonassociative operators

- For example, the expression **5 < 7 < 9** does not mean **(5 < 7) < 9** or **5 < (7 < 9)**.
- Also, the statement **5 < 7 < 9** is same as **5 < 7** and **7 < 9**, and gets evaluated from left-to-right.
- Moreover, chaining of assignments operators like **a = b = c** is perfectly alright whereas the '**a = b += c**' will result in an error.
- # Set the values of a, b, c  
x = 11, y = 12, z = 13  
# Expression is incorrect  
# Non-associative operators  
# Error -> SyntaxError: invalid syntax  
**x = y += 12**