Python Operator Precedence

Python Operator Precedence

An expression may have multiple operators to be evaluated. The operator precedence defines the order in which operators are evaluated. In other words, the order of operator evaluation is determined by the operator precedence.

If a certain expression contains multiple operators, their order of evaluation is determined by the order of precedence. For example, consider the following expression

```
>>> a = 2+3*5
```

Here, what will be the value of **a**? - yes it will be 17 (multiply 3 by 5 first and then add 2) or 25 (adding 2 and 3 and then multiply with 5)? Python's operator precedence rule comes into picture here.

If we consider only the arithmetic operators in Python, the traditional **BODMAS** rule is also employed by Python interpreter, where the **brackets** are evaluated first, the **division** and **multiplication** operators next, followed by **addition** and **subtraction** operators. Hence, a will become 17 in the above expression.

In addition to the operator precedence, the associativity of operators is also important. If an expression consists of operators with same level of precedence, the associativity determines the order. Most of the operators have left to right associativity. It means, the operator on the left is evaluated before the one on the right.

Let us consider another expression:

```
>>> b = 10/5*4
```

In this case, both * (multiplication) and / (division) operators have same level of precedence. However, the left to right associativity rule performs the division first (10/5 = 2) and then the multiplication (2*4 = 8).

Python Operator Precedence Table

The following table lists all the operators in Python in their decreasing order of precedence. Operators in the same cell under the Operators column have the same precedence.

Sr.No.	Operator & Description

1	**(),[], {}** Parentheses and braces
2	**[index], [index:index]** Subscription, slicing,
3	**await x** Await expression
4	****** Exponentiation
5	**+x, -x, ~x** Positive, negative, bitwise NOT
6	***, @, /, //, %** Multiplication, matrix multiplication, division, floor division, remainder
7	**+, -** Addition and subtraction
8	**<<, >>** Left Shifts, Right Shifts
9	**&** Bitwise AND
10	**^** Bitwise XOR
11	**|** Bitwise OR
12	**in, not in, is, is not, <, <=, >, >=, !=, ==** Comparisons, including membership tests and identity tests
13	**not x** Boolean NOT
14	**and** Boolean AND
15	**or** Boolean OR
16	**if – else** Conditional expression

17	**lambda** Lambda expression
18	**:=** Walrus operator

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

Python Operator Precedence Example

</> Open Compiler

```
a = 20
b = 10
c = 15
d = 5
e = 0

e = (a + b) * c / d       #( 30 * 15 ) / 5
print ("Value of (a + b) * c / d is ",  e)

e = ((a + b) * c) / d     # (30 * 15 ) / 5
print ("Value of ((a + b) * c) / d is ",  e)

e = (a + b) * (c / d);    # (30) * (15/5)
print ("Value of (a + b) * (c / d) is ",  e)

e = a + (b * c) / d;      #  20 + (150/5)
print ("Value of a + (b * c) / d is ",  e)
```

When you execute the above program, it produces the following result –

```
Value of (a + b) * c / d is  90.0
Value of ((a + b) * c) / d is  90.0
Value of (a + b) * (c / d) is  90.0
Value of a + (b * c) / d is  50.0
```