## PRECEDENCE OF OPERATORS

**Operator:**

An operator is a special symbol that is used to perform particular mathematical or logical computations like addition, multiplication, comparison and so on. The value of operator is applied to be called operands. For e.g., in the expression 4 + 5, 4 and 5 are operands and + is an operator. The following tokens are operators in Python:

| | | | | | | |
|---|---|---|---|---|---|---|
| + | - | * | ** | / | // | % |
| << | >> | & | \| | ^ | ~ | <> |
| < | > | <= | >= | == | != | |

**Operator precedence:**

When an expression contains more than one operator, precedence is used to determine the order of evaluation. The order of evaluation depends on rules of precedence.

**Rules of precedence**

i) Parenthesis has the highest precedence.

***Example:*** *5**(9-3)

Here expression in parenthesis is evaluated first.

*5**(9-3) = 5 * (6) = 30

ii) Exponentiation has next highest precedence.

***Example:*** *1*+2**3

*1*+2**3 = 1 + 8 = 9

iii) Multiplication and division have next higher precedence than addition and subtraction.

***Example:*** 3*2-1

3*2-1 = 6 – 1 = 5

iv) Operators with same precedence are evaluated from left to right (Except exponentiation).

The following table summarizes the operator precedence in Python, from the highest precedence to the lowest precedence. Operators in the same box have the same precedence and group from left to right (except for comparisons statements).

| Operator | Description | Associativity |
|---|---|---|
| (expressions...)<br>[expressions...]<br>{ key: value...}<br>'expressions...' | Binding or tuple display<br>list display<br>dictionary display<br>string conversion | left to right |
| x[index]<br>x[index:index]<br>x(arguments...)<br>x.attribute | Subscription<br>Slicing<br>Call<br>Attribute reference | left to right |
| ** | Exponentiation | right-to-left |
| +x , -x<br>~x | Unary plus and Unary minus<br>Bitwise NOT | left to right |
| *<br>/<br>//<br>% | Multiplication<br>Division<br>Floor division<br>Remainder | left to right |
| +, - | Addition and Subtraction | left to right |
| <<, >> | Bitwise Left Shift and Right Shift | left to right |
| & | Bitwise AND | left to right |
| ^ | Bitwise XOR | left to right |
| \| | Bitwise OR | left to right |
| in, not in<br>is, is not<br><,<=,>,>=,<>,!=,== | Membership tests<br>Identity tests<br>Comparisons | Chain from left to right |
| not | Boolean NOT | left to right |
| and | Boolean AND | left to right |

| | | |
|---|---|---|
| or | Boolean OR | left to right |

→The acronym **PEMDAS** is useful way to remember the rules. That is,

P (parenthesis first)

E (Exponent)

MD (multiplication and division)

AS (addition and subtraction)

→Arithmetic evaluation is carried out using two phases from left to right. During the first phase highest priority operators are evaluated. The second phase lowest priority operators are evaluated.

***Example:***

6+4/2 is 8, not 5.

***Example:***

```
>>>X, Y, Z=2, 3, 4
>>>value=X-Y/3+Z*3-1
>>>print(“Result=”,value)
    Result=12
```

**COMMENTS**

Comments are the non-executable statements explain what the program does. For large programs it often difficult to understand what is does. The comment can be added in the program code with the symbol #.

Comment contains information for persons reading the program. Comment lines are ignored during program execution. Comment lines have no effect on the program results.

***Example:***

```
print ‘Hello, World!’  # print the message Hello, World!; comment
v=7                    # creates the variable v and assign the value 7; comment
```

**Types of comments**

i) Single line comments

ii) Multi line comments

**i) Single line comments -** Single line comments describes information in one line(short),and they start with the symbol #.Everything from the # to the end of the line is ignored, it has no effect on the execution of the program.

***Example:***

```
>>>#This is a print statement
>>>print(“Hello python”)
       Hello python
```

**ii) Multi line comments -** Multi line comments describes information in detail. Here, more thanone line of comments can be added to the program in between the triple quotes(“ ” ”).

***Example:***

```
”””This is print
statement This line
print “Hello
python”
 Written by ABC,
April 2017”””
```