# The combination of values, variables, operators, and function calls is termed as an expression. The Python interpreter can evaluate a valid expression.

# Operator precedence determines the grouping of terms in an expression and decides how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has a higher precedence than the addition operator.

# The effect of a given operator is based on both position and the rules of operator precedence. This concept is shown by the following examples:

## A = 4 + 5 \* 2

# A is equal to 14 since the multiplication operator has a higher precedence than the addition operator. Parentheses can be used to override the default evaluation.

## A = (4 + 5) \* 2

# In this case, A equals 18 because the parentheses have higher operator precedence than the multiplication operator; the expression inside the parentheses is evaluated first, and the result is multiplied by two.

# Position within the expression is used to determine the order of evaluation when two or more operators share the same operator precedence. Consider the following:

## A = 6 / 2 \* 3

# In this case, A equals 9, since the division operator is to the left of the multiplication operator. The sub expression 6 / 2 is evaluated before the multiplication is done, even though the multiplication and division operators have the same precedence. Again, parentheses can be used to override the default evaluation order:

## A = 6 / (2 \* 3)

# In this case, A equals 1, because the expression inside parentheses is evaluated first.

# The operator precedence in Python is listed in the following table. It is in descending order (upper group has higher precedence than the lower ones).

|  |  |
| --- | --- |
| Operators | Meaning |
| ( ) | Parentheses |
| \*\* | Exponent |
| +x, -x, ~x | Unary plus, Unary minus, Bitwise NOT |
| \*, /, //, % | Multiplication, Division, Floor division, Modulus |
| +, - | Addition, Subtraction |
| <<, >> | Bitwise shift operators |
| & | Bitwise AND |
| ^ | Bitwise XOR |
| | | Bitwise OR |
| ==, !=, >, >=, <, <=, is, is not, in, not in | Comparisons, Identity, Membership operators |
| not | Logical NOT |
| and | Logical AND |
| or | Logical OR |