**Expressions & Operators in Python**

In Python, **expressions** are combinations of values, variables, and operators that Python interprets and evaluates to produce a value. **Operators** are special symbols that perform operations on operands (values or variables).

**1. Expressions**

An **expression** is any combination of literals, variables, operators, and function calls that Python can evaluate to produce a value.

**Examples of Expressions:**

5 + 3 # Arithmetic expression, evaluates to 8

x \* 10 # Expression using a variable

a > b # Comparison expression, evaluates to True or False

"Hello " + "World" # String concatenation expression

**2. Operators**

Operators allow us to perform operations on variables and values. Python has different categories of operators:

**A. Arithmetic Operators**

These operators perform mathematical operations.

| **Operator** | **Description** | **Example (a = 10, b = 3)** | **Result** |
| --- | --- | --- | --- |
| + | Addition | a + b | 13 |
| - | Subtraction | a - b | 7 |
| \* | Multiplication | a \* b | 30 |
| / | Division | a / b | 3.33 |
| // | Floor Division | a // b | 3 |
| % | Modulus (Remainder) | a % b | 1 |
| \*\* | Exponentiation | a \*\* b | 1000 |

**B. Comparison (Relational) Operators**

These operators compare values and return a Boolean result (True or False).

| **Operator** | **Description** | **Example (a = 10, b = 5)** | **Result** |
| --- | --- | --- | --- |
| == | Equal to | a == b | False |
| != | Not equal to | a != b | True |
| > | Greater than | a > b | True |
| < | Less than | a < b | False |
| >= | Greater than or equal to | a >= b | True |
| <= | Less than or equal to | a <= b | False |

**C. Logical Operators**

Used for combining conditional statements.

| **Operator** | **Description** | **Example (x = True, y = False)** | **Result** |
| --- | --- | --- | --- |
| and | Returns True if both conditions are true | x and y | False |
| or | Returns True if at least one condition is true | x or y | True |
| not | Reverses the boolean value | not x | False |

**D. Bitwise Operators**

Operate on binary numbers (bit-level operations).

| **Operator** | **Description** | **Example (a = 5, b = 3)** | **Result** |
| --- | --- | --- | --- |
| & | Bitwise AND | a & b (0101 & 0011) | 0001 (1) |
| ` | ` | Bitwise OR | `a |
| ^ | Bitwise XOR | a ^ b (0101 ^ 0011) | 0110 (6) |
| ~ | Bitwise NOT | ~a (inverts bits) | -6 |
| << | Left Shift | a << 1 | 10 |
| >> | Right Shift | a >> 1 | 2 |

**E. Assignment Operators**

Used to assign values to variables.

| **Operator** | **Example (x = 10)** | **Equivalent to** |
| --- | --- | --- |
| = | x = 10 | x = 10 |
| += | x += 5 | x = x + 5 |
| -= | x -= 5 | x = x - 5 |
| \*= | x \*= 5 | x = x \* 5 |
| /= | x /= 5 | x = x / 5 |
| //= | x //= 5 | x = x // 5 |
| %= | x %= 5 | x = x % 5 |
| \*\*= | x \*\*= 5 | x = x \*\* 5 |

**F. Identity Operators**

Used to compare memory locations of objects.

| **Operator** | **Description** | **Example (a = 10, b = 10, c = [1,2,3])** | **Result** |
| --- | --- | --- | --- |
| is | Returns True if both variables refer to the same object | a is b | True |
| is not | Returns True if they refer to different objects | a is not c | True |

**G. Membership Operators**

Used to check if a value exists in a sequence (list, tuple, string, etc.).

| **Operator** | **Description** | **Example (x = [1,2,3,4])** | **Result** |
| --- | --- | --- | --- |
| in | Returns True if value is in sequence | 2 in x | True |
| not in | Returns True if value is not in sequence | 5 not in x | True |

**3. Order of Precedence**

Python follows **precedence rules** when evaluating expressions.

**Precedence Table (Highest to Lowest)**

| **Precedence** | **Operator Category** | **Operators** |
| --- | --- | --- |
| 1 (Highest) | Parentheses | () |
| 2 | Exponentiation | \*\* |
| 3 | Unary Operators | +x, -x, ~x |
| 4 | Multiplication, Division | \*, /, //, % |
| 5 | Addition, Subtraction | +, - |
| 6 | Bitwise Shift | <<, >> |
| 7 | Bitwise AND | & |
| 8 | Bitwise XOR | ^ |
| 9 | Bitwise OR | ` |
| 10 | Comparison Operators | ==, !=, >, <, >=, <= |
| 11 | Logical NOT | not |
| 12 | Logical AND | and |
| 13 (Lowest) | Logical OR | or |

**Example of Precedence**

x = 5 + 2 \* 3 # Multiplication happens first, then addition

print(x) # Output: 11

y = (5 + 2) \* 3 # Parentheses change precedence

print(y) # Output: 21

**Summary**

* **Expressions** are any valid combinations of variables, values, and operators.
* **Operators** perform operations on values and are categorized into Arithmetic, Comparison, Logical, Bitwise, Assignment, Identity, and Membership operators.
* **Operator precedence** determines the order in which operations are performed.