TYPES OF OPERATORS

1. Arithmetic
   * Arithmetic operators are used to perform arithmetic operations between variables.
   * +, -, \*, /, %, \*\*, //
2. Assignment
   * Assignment operators are used to assign values.

|  |  |
| --- | --- |
| = | x=10 |
| += | x+=10 |
| -= | x-=10 |
| \*= | x\*=5 |
| %= | x%=20 |
| \*\*= | x\*\*=10 |
| //= | x//=25 |
| |= | x|=20 |
| ^= | x^=15 |
| &= | x&=12 |

1. Comparison
   * Comparison operators are used to compare two values

|  |  |
| --- | --- |
| == | Equals to |
| != | Not equals to |
| > | Greater than |
| < | Less than |
| >= | Greater than equal to |
| <= | Less than equal to |

* compare = x < y

Compare

1. Logical
   * Logical operators are used to combine conditional statements
     1. Logical and
     2. Logical or
     3. Logical not
   * X =10, y=15
   * if x > y:

print("x greater")

print("not equal")

elif x == y:

print("equal")

elif x < y:

print ("y greater")

print("not eqaul")

1. Membership
   * Membership operators are used to check if sequence is present in object.
     1. IN
        + Returns true if a sequence with specified value is present in object.
        + Example: x IN y
     2. NOT IN
        + Returns true if a sequence with the specified value is not present in object.
        + Example: x IS NOT y
2. Identity
   * Identity operators are used to compare objects.
     1. IS
        + Returns true if both variable are same object.
        + Example: x IS y
     2. IS NOT
        + Returns true if both variables are not same object.
        + Example: X IS NOT Y
3. Bitwise
   * Bitwise operators are used to compare binary numbers.
     1. BITWISE AND (&)
        + Sets each bit to 1 if both bits are 1.
     2. BITWISE OR (|)
        + Sets each bit to 1 if one of the bits is 1.
     3. BITWISE XOR (^)
        + Sets each bit to 1 if only one of bits is 1.
     4. BITWISE NOT (~)
        + Inverts all bits.
     5. Left Shift (<<)
        + Shift left by pushing in zeroes from right and let the leftmost bits fell off.
     6. Right Shift (>>)
        + Shift right by pushing copies of leftmost bit in from left and let rightmost bit fall off.