OPERATORS IN PYTHON

**Operator**: It is defined as a symbol which is responsible for a particular operation between two operands.

Python supports different types of operators:

1. Arithmetic Operators
2. Assignment Operators
3. Logical Operators
4. Comparison Operators
5. Bitwise Operators
6. Identity Operators
7. Membership Operators

**1.Arithmetic Operators:** They are used to perform Mathematical operations between two operands.i.e Addition,Subtraction,Multiplication,Division etc.

**Use Case**: used to perform basic arithmetic operators

| Operator | Description | Syntax | Example(x=5,y=2) |
| --- | --- | --- | --- |
| + (Addition) | Add two Operands | ( x+y ) | 5+2=10 |
| - (Subtraction) | Subtract two Operands | ( x-y ) | 5-2=3 |
| \* (Multiplication) | multiply two Operands | ( x\*y ) | 5\*2=10 |
| / (Division) | Divide first operand by second | ( x/y ) | 5/2=2.5 |
| \*\* (Power) | First Operand raised to power of second | (x\*\*y ) | 5\*\*2=25 |
| % (Modulus) | Remainder when first Operand is Divide by second | ( x%y ) | 5%2=1 |
| // (floor Division) | Divide first operand by second | ( X//y ) | 5//2=2 |

Example:

**INPUT:**

a = 18

b = 12

# Addition of numbers

su= a + b

#Subtraction of numbers

sub = a - b

# Multiplication of numbers

mul = a \* b

# Division(float) of numbers

div1= a / b

# Division(floor) of numbers

div2 = a // b

# Modulo of both numbers

mod=a%b

# Power of the numbers

power = a \*\* b

print("Addition of numbers:",su)

print("Subtraction of numbers:", sub)

print("Multiplication of numbers:", mul)

print("Division(float) of numbers:", div1)

print("Division(floor) of numbers:", div2)

print("Modulo of both numbers:",mod)

print("Power of the numbers:", power)

Output

Addition of numbers: 30

Subtraction of numbers: 6

Multiplication of numbers: 216

Division(float) of numbers: 1.5

Division(floor) of numbers: 1

Modulo of both numbers: 6

Power of the numbers: 1156831381426176

**ASSIGNMENT OPERATORS**

**Assignment operators:** They are used to assign values of right expression to the left operand .

**Use case : used to manipulate variables, most commonly used in loops.**

| Operator | Description | Syntax(take s and t) |
| --- | --- | --- |
| (=) | Assign value of right side of expression to left side operand | s=t |
| (+=) | Add right operand from left operand and then assign to left operand | s+=t (or) s=s+t |
| (-=) | Subtract right operand from left operand and then assign to left operand | s-=t (or) s=s-t |
| (\*=) | Multiply right operand from left operand and then assign to left operand | s\*=t (or) s=s\*t |
| (/=) | Divide left operand with right operand and then assign to left operand | s/=t (or) s=s/t |
| (%=) | Takes modulus using left and right operands and assign the result to left operand | s%=t (or) s=s%t |
| (\*\*=) | Calculate exponent value and assign value to left operand | s\*\*=t (or) s=s\*\*t |
| (//=) | Divide left operand with right operand and then assign the value(floor) to left operand | s//=t (or) s=s//t |

**Example:**

#assigning the value

s = 10

t = 5

print(s)

# adding and assigning

s+=t

print(s)

#subtracting and assigning

t-=s

print(t)

# multiply and assigning

t \*=t

print(t)

#divide and assigning

t/=s

print(t)

OUTPUT:  
10

15

-10

100

6.666666666666667

**Comparison operators**

Comparison operators compare the values of the two operands and return a True or False i.e Boolean Value

**Use case :to evaluate conditions and make logical decisions**

| Operator | Description | Syntax |
| --- | --- | --- |
| (==) Equal to | True if both operands are equal | S==T |
| (!=) Not Equal to | True if operands are not equal | S!=T |
| ( >) Greater than | True if the left operand is greater than the right | S>T |
| ( <) Less than | True if the left operand is Less than the right | S<T |
| ( >=) Greater than or equal to | True if the left operand is greater than or equal to the right | S>=T |
| ( <=) Less than or equal to | True if the left operand is less than or equal to the right | S<=T |

**Example:**

INPUT:

s = 10

t = 2

#validate for equality

print( s ==t)

#validate for unequality

print ( s !=t)

# validate for greater than

print (s > t)

#validate for less than

print (s < t)

#validate for greater than or equal to

print (s >= t)

#validate for less than or equal to

print (s <= t)

OUTPUT:

False

True

True

False

True

False

**BITWISE OPERATORS**

**Bitwise Operators:** performs bit by bit operations on two operands.

**Use case : mostly used in low level programming like hardware manipulation and optimizing the algorithms**

| Operator | Description | Syntax(X, Y) |
| --- | --- | --- |
| & (BITWISE AND) | Resulting bit will be 1 only if both the bits are 1 else 0 | X&Y |
| | (BITWISE OR) | Resulting bit will be 1 if either of the bits is 1 | X|Y |
| ~ (BITWISE NOT) | Resulting will be compliment of the given operand | ~X |
| ^ (BITWISE XOR) | Resulting will be 1 when both the bits are different otherwise 0 | X^Y |
| >> (RIGHT SHIFT) | Shift the left operand bits towards the right for right operand number of times | X>>Y |
| << (LEFT SHIFT) | Shift the left operand bits towards the left for the right operand number of times | X<<Y |

**Example:**

**INPUT:**

s = 10

t = 12

print("The binary value of s:",bin(s))

print("The binary value of t:", bin(t) )

#Bitwise AND

print("Result of BITWISE AND is ", s&t, ":" ,bin(s&t))

#Bitwise OR

print("Result of BITWISE OR is ",s|t, ":" ,bin(s|t))

#Bitwise not

print("Result of BITWISE not is ",~s,":" ,bin(~s))

# Bitwise XOR

print("Result of BITWISE XOR is ",s^t,":" ,bin(s^t))

#Bitwise Right shift

print("Result of BITWISE Right shift is ",s>>t,":" ,bin(s>>t))

# Bitwise left shift

print("Result of BITWISE left shift is ",s<<t,":" ,bin(s<<t))

**Output:**

The binary value of s: 0b1010

The binary value of t: 0b1100

Result of BITWISE AND is 8 : 0b1000

Result of BITWISE OR is 14 : 0b1110

Result of BITWISE not is -11 : -0b1011

Result of BITWISE XOR is 6 : 0b110

Result of BITWISE Right shift is 0 : 0b0

Result of BITWISE left shift is 40960 : 0b1010000000000000

**LOGICAL OPERATORS**

Logical Operators: used in the expression evaluation to make a decision.

They are three logical operators 1.Logical AND 2.Logical OR 3.Logical NOT

**Use case : used to combine multiple conditions in the decision making process and filtering the data.**

| Operator | Description | Syntax(S,T are operands) |
| --- | --- | --- |
| AND | True if both the operands are False | S and T |
| OR | True when either of the operands is True | S or T |
| NOT | True if the operand is False | not S |

Example:

s=True

t = False

#Logical and print(s and t)

print(s and t)

#Logical or print(s or t)

print(s or t)

#logical not print(not s)

print(not t)

print(not s)

OUTPUT:

False

True

True

False

**IDENTITY OPERATORS**

They are used to check if two values are located in the same part of memory or both belong to the same class or not.

There are two identical operators:

1.is

2.is not

**Use case : used to compare memory locations of the object**

| Operator | Description of operator |
| --- | --- |
| is | True if the two operands share same memory location |
| Is not | True if the two operands share different memory location |

We can verify it using the in built function of python called id ( )

Example:

INPUT:

t = 99

m = 99

#checking for Unique id

print("the location of t is ", id(t) )

print("the location of m is ", id(m) )

print(t is m)

print(t is not m)

v = 60

k = 90

print("the location of v is ", id(v) )

print("the location of k is ", id(k) )

print(v is not k)

print(v is k)

OUTPUT:

the location of t is 140680953024864

the location of m is 140680953024864

True

False

the location of v is 140680953023616

the location of k is 140680953024576

True

False

**MEMBERSHIP OPERATORS**

**Membership operators**:it is used to check whether a value is present in the sequence or not

They are two membership Operators:

1. in
2. not in

Example:

INPUT:

l= [10, 20, 30, 40, 60, 50, 90]

#using in Operator

if 60 in l:

print("60 is Present in list l")

else:

print("60 is Not Present in list l")

#using not in operator

if 80 not in l:

print("80 is not present in list l")

else:

print("80 is present in list l")

OUTPUT:

60 is Present in list l

80 is not present in list l