SNS COLLEGE OF ENGINEERING

Kurumbapalayam(Po), Coimbatore – 641 107
Accredited by NAAC-UGC with 'A' Grade
Approved by AICTE, Recognized by UGC & Affiliated to Anna University, Chennai

Department of Artificial Intelligence and Data Science

Course Name – Computational Thinking and Python Programming

I Year / I Semester

Unit 2-DATA, EXPRESSIONS, STATEMENTS

OPERATORS:

1. Operators are the constructs which can manipulate the value of operands.
2. Consider the **expression *4 + 5 = 9.*** Here, **4 and 5 are called operands** and **+ is called operator**

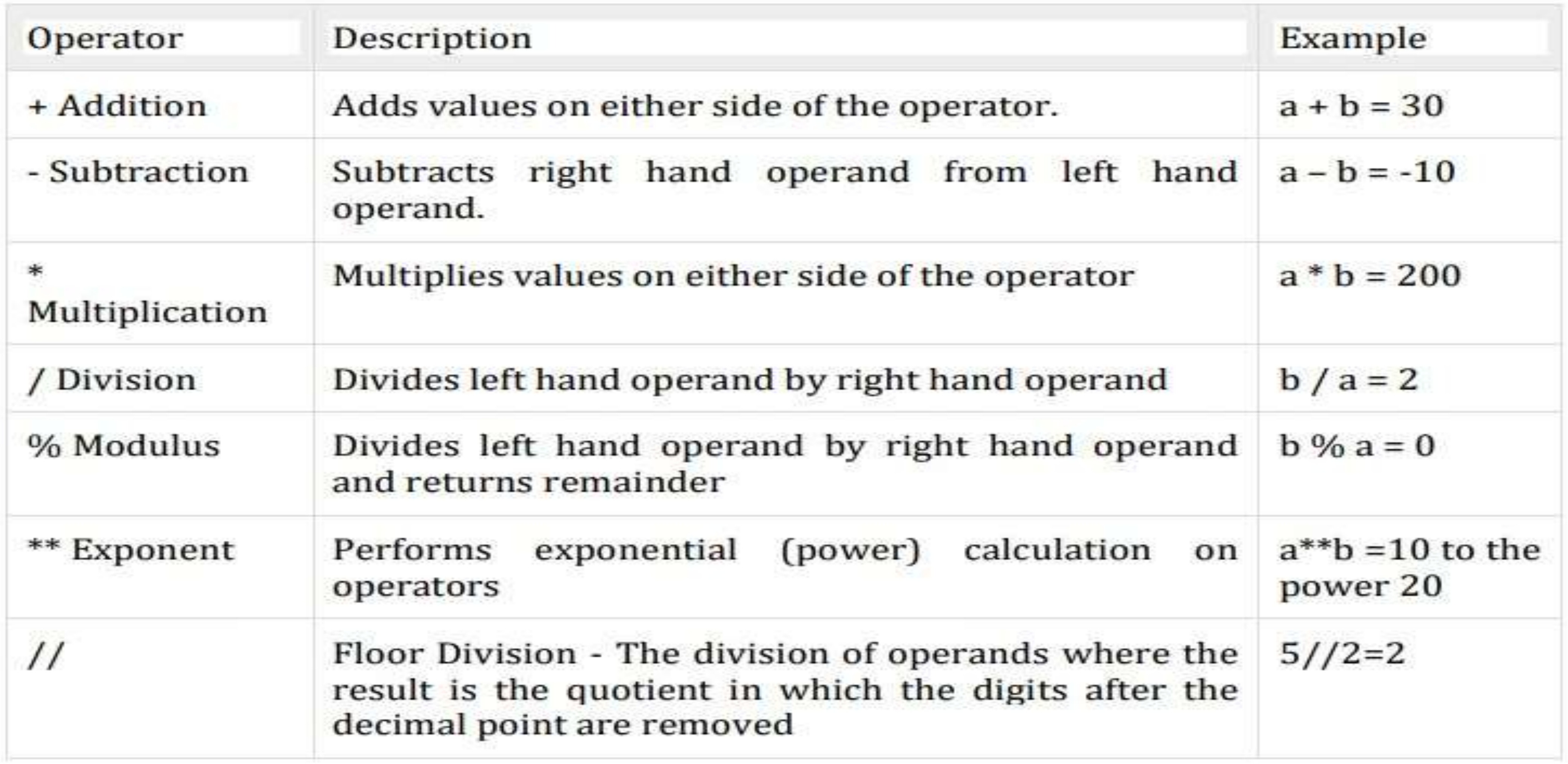
Types of Operators:

-Python language supports the following types of operators

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic operators:

They are used to perform **mathematical operations** like addition, subtraction, multiplication etc. **Assume, a=10 and b=5**



Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed	5//2=2

Examples

```
a=10
b=5
print("a+b=",a+b)
print("a-b=",a-b)
print("a*b=",a*b)
print("a/b=",a/b)
print("a%b=",a%b)
print("a//b=",a//b)
print("a**b=",a**b)
```

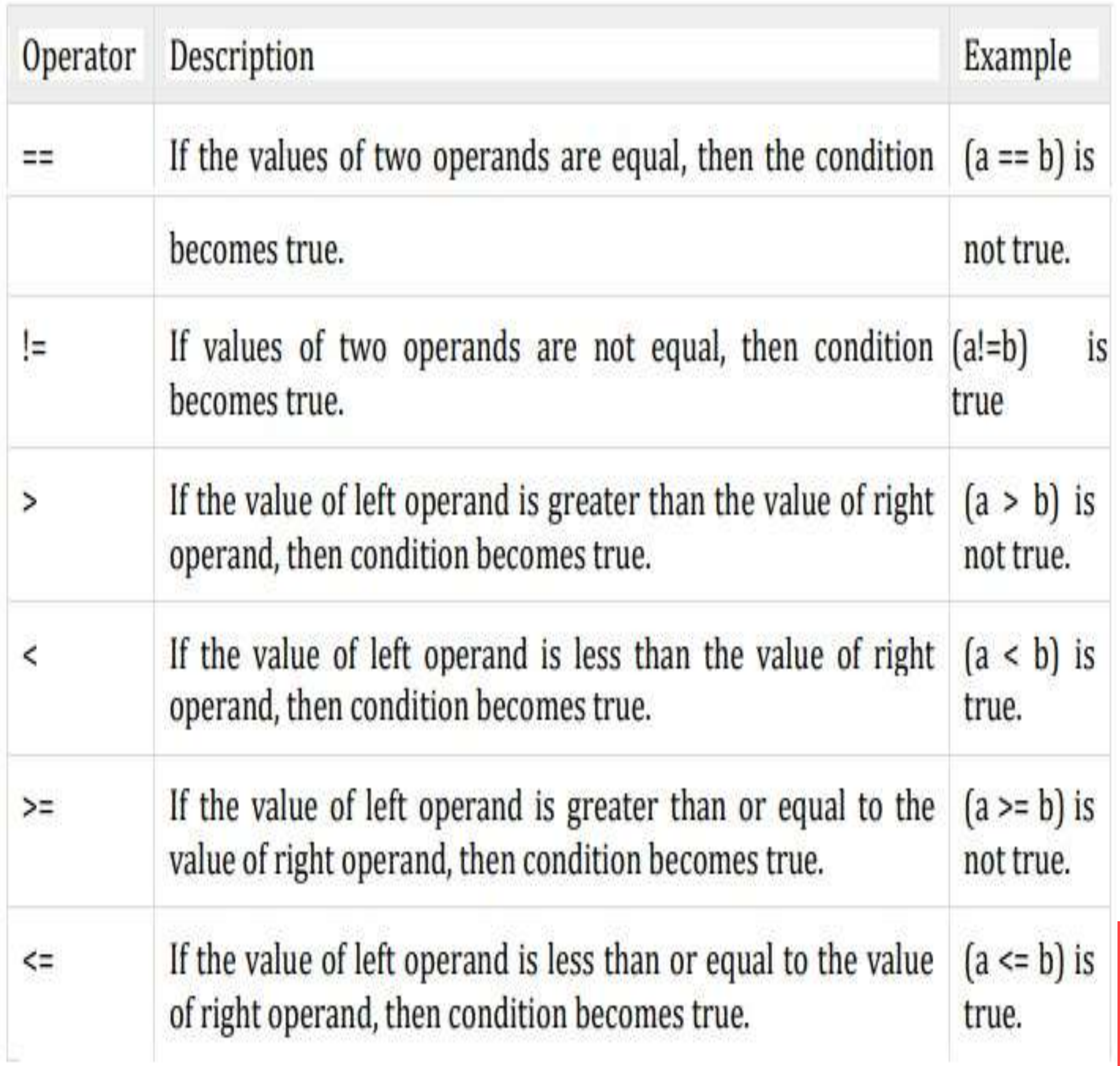
Output:

```
 a+b= 15
a-b= 5
a*b= 50
a/b= 2.0
a%b= 0
a//b= 2
a**b= 100000
```

Comparison (Relational) Operators:

-

- Comparison operators are used to compare values.
- It either returns True or False according to the condition. **Assume, a=10 and b=5**



Operator	Description	Example
==	If the values of two operands are equal, then the condition	(a == b) is
	becomes true.	not true.
!=	If values of two operands are not equal, then condition becomes true.	(a!=b) is true
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Example

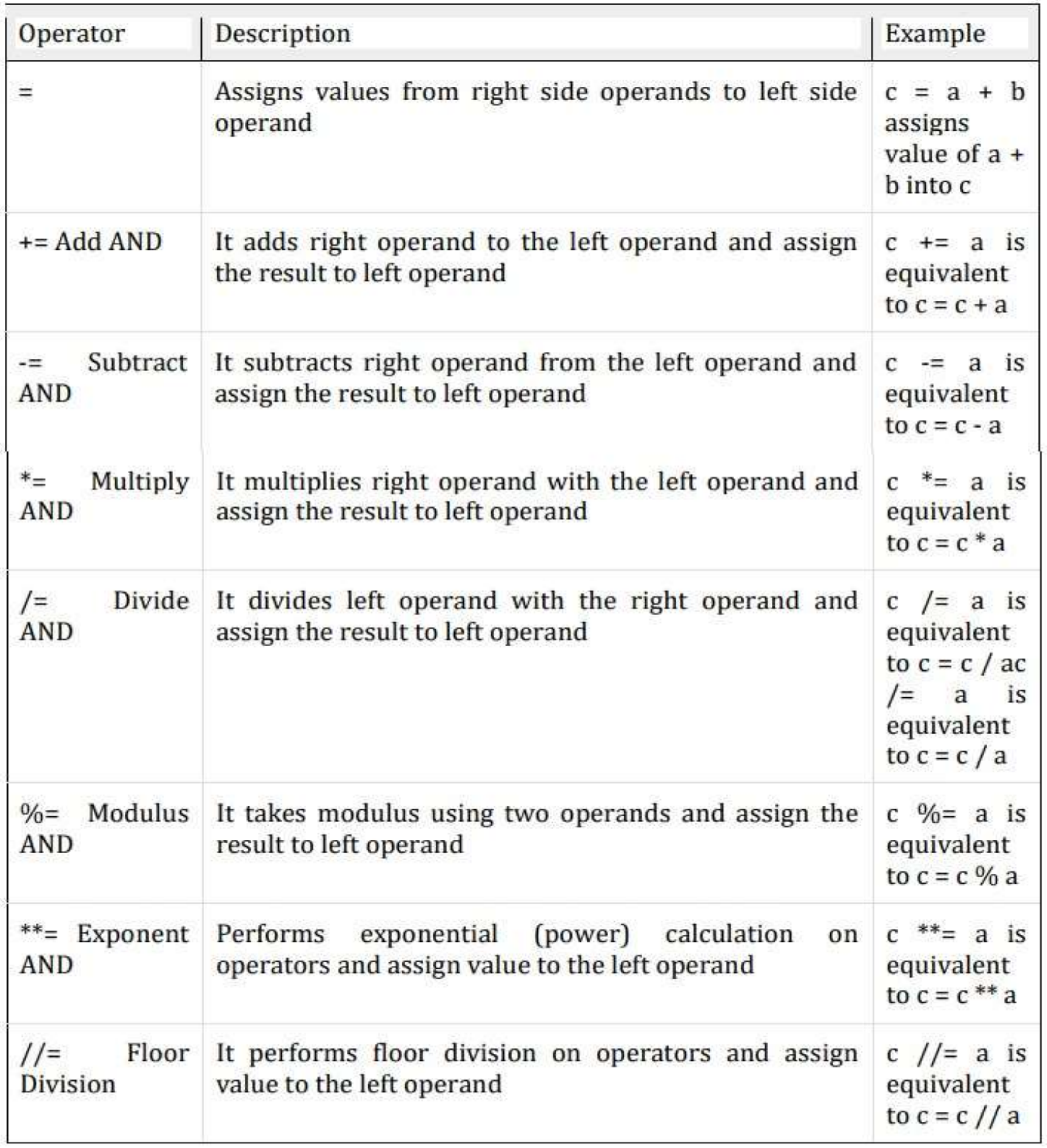
```
a=10
b=5
print("a>b=>",a>b)
print("a>b=>",a<b)
print("a==b=>",a==b)
print("a!=b=>",a!=b)
print("a>=b=>",a<=b)
print("a>=b=>",a>=b)
```

Output:

```
a>b=> True
a>b=> False
a==b=> False
a!=b=> True
a>=b=> False
a>=b=> True
```

<u>**Assignment Operators:**</u>

-Assignment operators are used in Python to assign values to variables.



Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a

Example

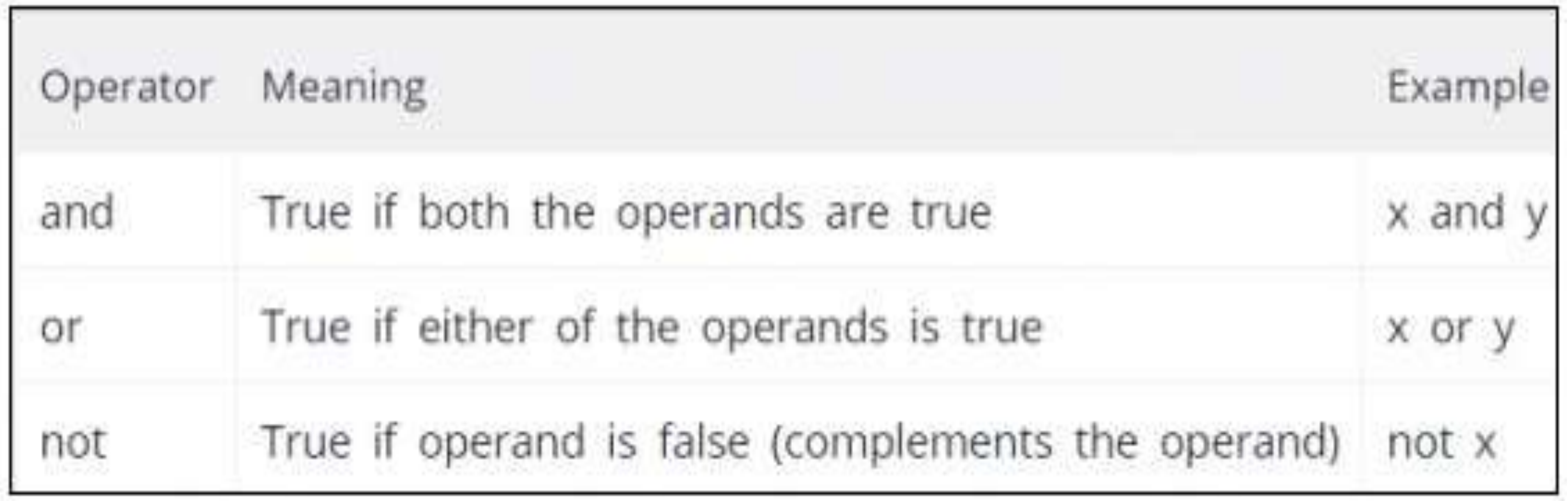
```
a = 21
b = 10
c = 0
c = a + b
print("Line 1 - Value of c is ", c)
c += a
print("Line 2 - Value of c is ", c)
c *= a
print("Line 3 - Value of c is ", c)
c /= a
print("Line 4 - Value of c is ", c)
c= 2 c %= a
print("Line 5 - Value of c is ", c) c **= a
print("Line 6 - Value of c is ", c) c //= a
print("Line 7 - Value of c is ", c)
```

Output

```
Line 1 - Value of c is 31
Line 2 - Value of c is 52
Line 3 - Value of c is 1092
Line 4 - Value of c is 52.0
Line 5 - Value of c is 2
Line 6 - Value of c is 2097152
Line 7 - Value of c is 99864
```

Logical Operators:

-Logical operators are the and, or, not operators.



Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Example

```
a = True
b = False
print('a and b is',a and b)
print('a or b is',a or b)
print('not a is',not a)
```

Output

```
x and y is False
x or y is True
not x is False
```

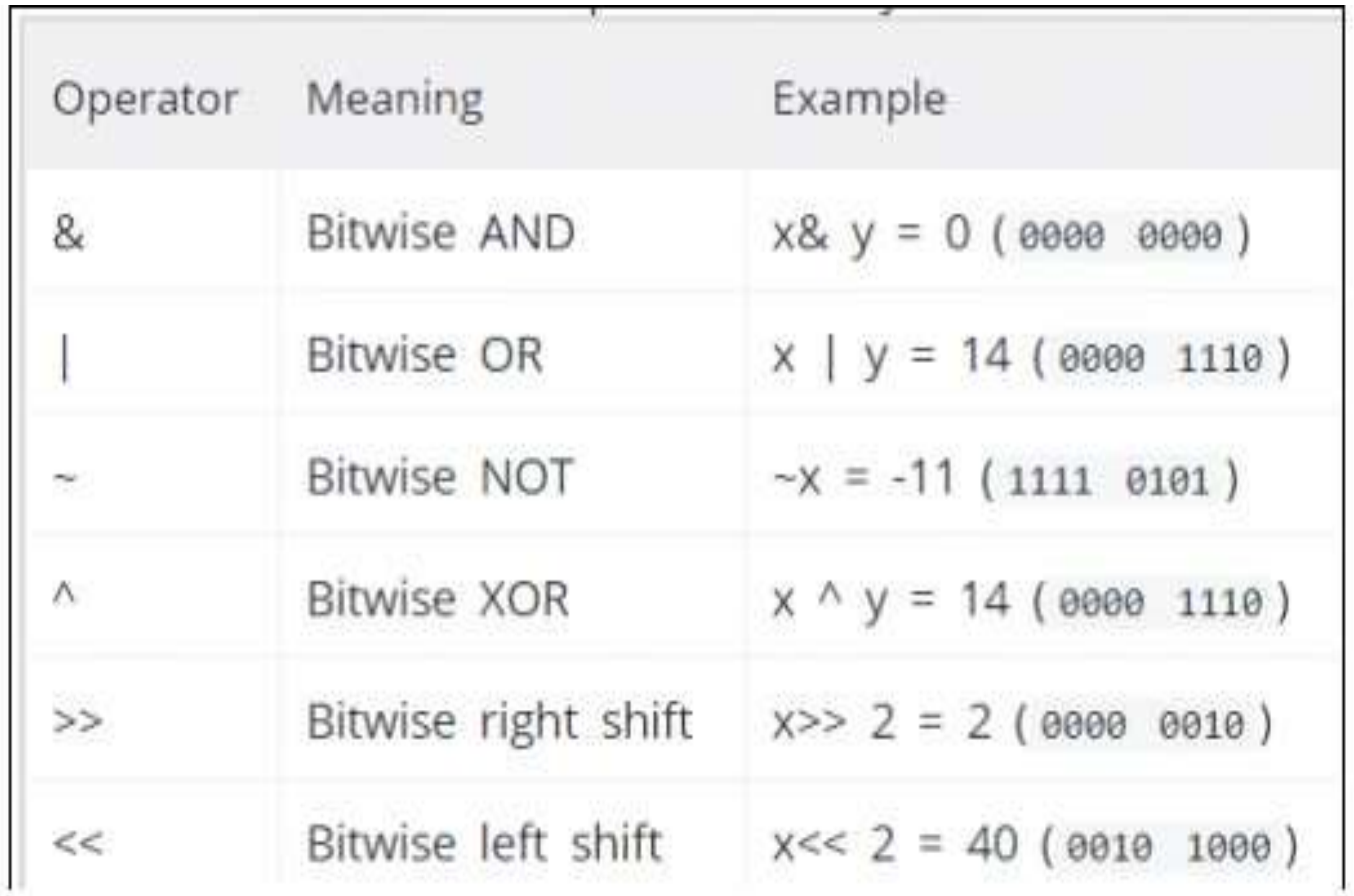
Bitwise Operators:

A **bitwise operation** operates on one or more **bit** patterns at the level of individual Bits

Example:

Let x = 10 (0000 1010 in binary) and

y = 4 (0000 0100 in binary)



Operator	Meaning	Example
&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x>> 2 = 2 (0000 0010)
<<	Bitwise left shift	x<< 2 = 40 (0010 1000)

Example

```
a = 60      # 60 = 0011 1100
b = 13     # 13 = 0000 1101
c = 0
c = a & b;      # 12 = 0000 1100
print "Line 1 - Value of c is ", c
c = a | b;  # 61 = 0011 1101
print "Line 2 - Value of c is ", c
c = a ^ b; # 49 = 0011 0001
print "Line 3 - Value of c is ", c
c = ~a;    # -61 = 1100 0011
print "Line 4 - Value of c is ", c
c = a << 2;     # 240 = 1111 0000
print "Line 5 - Value of c is ", c
c = a >> 2;     # 15 = 0000 1111
print "Line 6 - Value of c is ", c
```

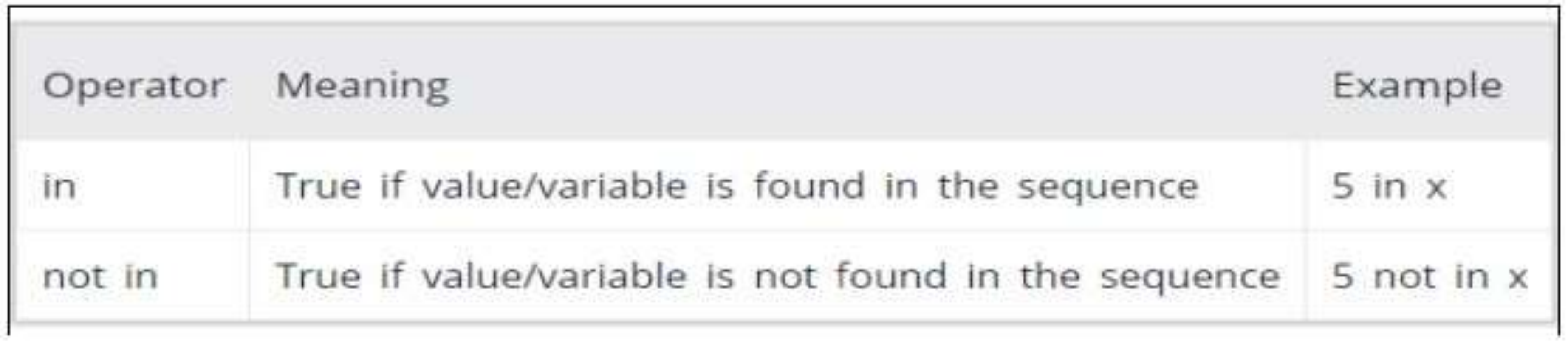
Output

```
Line 1 - Value of c is 12
Line 2 - Value of c is 61
Line 3      - Value of c is 49
Line 4      - Value of c is -61
Line 5      - Value of c is 240
Line 6      - Value of c is 15
```

Membership Operators:

-

1. Evaluates to find a value or a variable is in the specified sequence of string, list, tuple, dictionary or not.
2. Let, **x=[5,3,6,4,1].** To check particular item in list or not**, in and not in** operators are used.

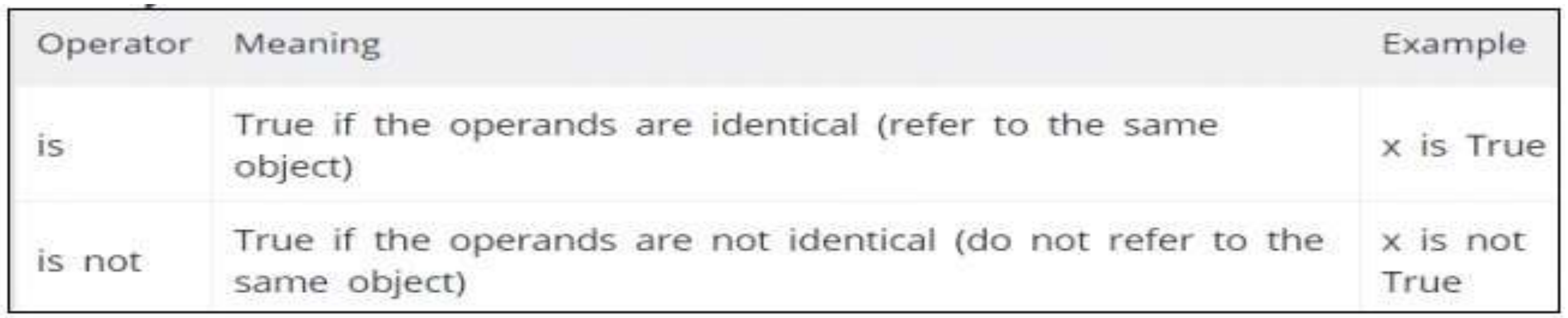


Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Example:
x=[5,3,6,4,1]
>>> 5 in x
True
>>> 5 not in x
False

Identity Operators

They are used to check if two values (or variables) are located on the same part of the memory.



Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Example

```
x = 5
y = 5
x2 = 'Hello'
y2 = 'Hello'
print(x1 is not y1)
print(x2 is y2)
```

Output

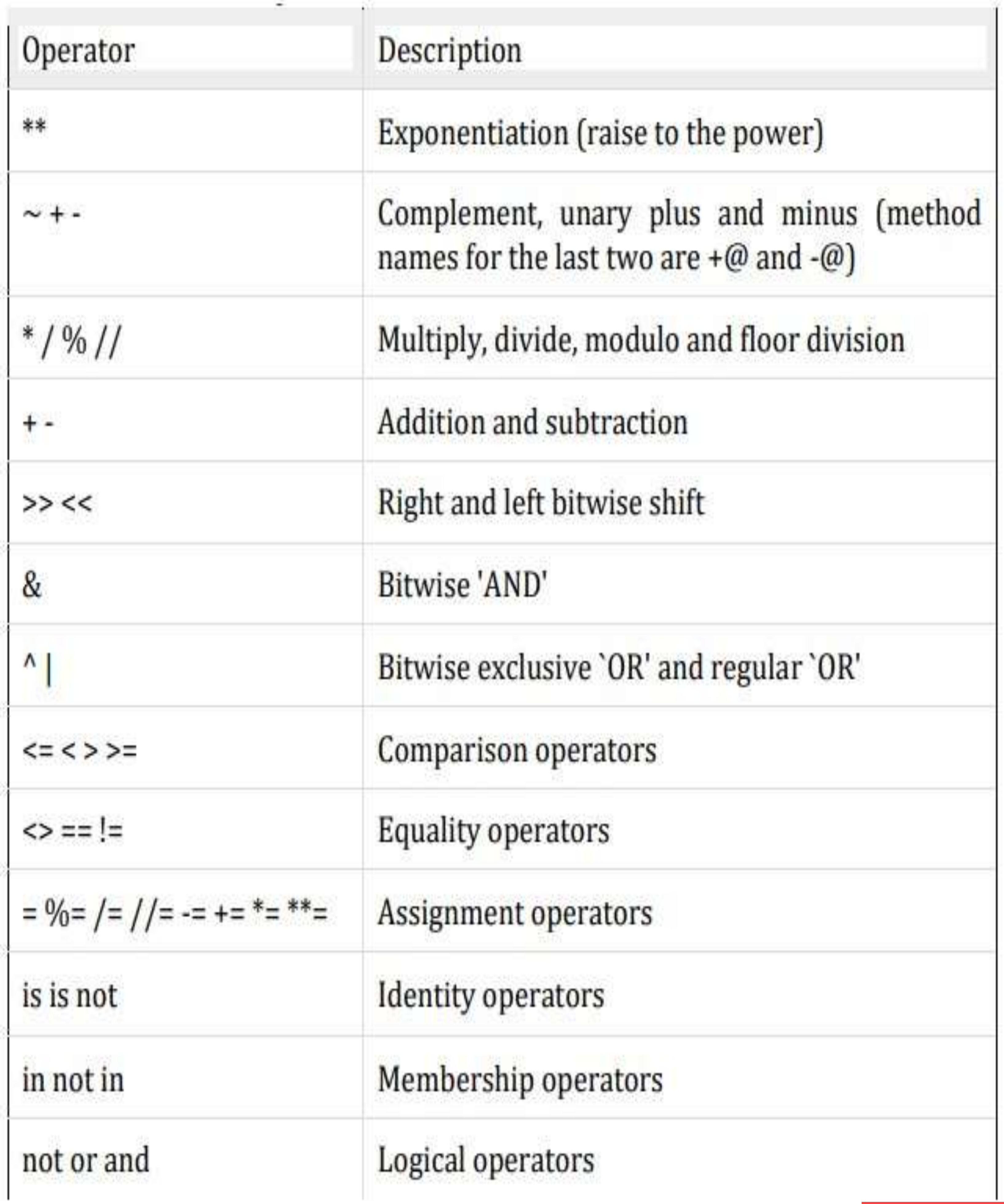
```
False
True
```

OPERATOR PRECEDENCE:

When an expression contains **more than one operator, the order of evaluation** depends on the order of operations.

For mathematical operators, Python follows mathematical convention.

-The acronym **PEMDAS** (Parentheses, Exponentiation, Multiplication, Division, Addition, Subtraction**)** is a useful way to remember the rules:



Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

1. Parentheses have the highest precedence and can be used to force an expression to evaluate in the order you want. Since expressions in parentheses are evaluated first, 2 * (3-1)is 4, and (1+1)**(5-2) is 8.

2. You can also use parentheses to make an expression easier to read, as in (minute * 100) / 60, even if it doesn’t change the result.

3. Exponentiation has the next highest precedence, so 1 + 2**3 is 9, not 27, and 2 *3**2 is 18, not 36.

4. Multiplication and Division have higher precedence than Addition and Subtraction. So 2*3-1 is 5, not 4, and 6+4/2 is 8, not 5.

5. Operators with the same precedence are evaluated from left to right (except exponentiation).

Example:

a=9-12/3+3*2-1
a=?
a=9-4+3*2-1
a=9-4+6-1
a=5+6-1
a=11-1
a=10

A=2*3+4%5-3/2+6
A=6+4%5-3/2+6
A=6+4-3/2+6
A=6+4-1+6
A=10-1+6
A=9+6
A=15

find m=?
m=-43||8&&0||-2
m=-43||0||-2
m=1||-2
m=1

a=2,b=12,c=1
d=a<b>c
d=2<12>1
d=1>1
d=0

a=2,b=12,c=1
d=a<b>c-1
d=2<12>1-1
d=2<12>0
d=1>0
d=1

a=2*3+4%5-3//2+6
a=6+4-1+6
a=10-1+6
a=15