Python operator falls into 7 categories

1) Arithmetic operator
2) Relational operator
3) Assignment operator
4) Logical operator
5) Membership operator
6) Identity operator
7) Bitwise operator)

1) Arithmetic operator :- There python Arithmetic operator include python operator for basic mathematical operation.

a) Addition (+) :- Adds the values on either side of the operator.

Ex :- >>> 3+4 O/P :- 7

b) Substraction (-) :- Substracts the value on the right from the one on the left

Ex :- >>> 3-4

O/P :- -1

c) Multiply (*) :- Multiplies the values on either sides of the operator

>>> 3*4

O/P :- 12

d) Division (/) :- Divides the values on the left side by the one on the right. Notice that division results in a floating-point value.

e) Exponentiation (**) :-

Raises the first number to the power of the second.

Ex:- >>> 3**4

O/P:- 8

f) Floor division (//) :-

Divides and returns the integer values of the quotient. It dumps the digits after the decimal

Ex:- >>> 3//4

>>> 4//3

O/P:- 1

g:- Modulus (%)

Divides and returns the value of the remainder

>>> 3%4

output :- 3

(9) Relational operator :- Carries out the comparision b/w operands. They tell us whether an operand is greater than the other, lesser, equal or a combination of these

a) Less than (<) :- This operator checks if the value on the left of the operators is less than one on the right.

ex: >>>3<4

O/P :- True

b) Greater than (>) :- It checks if the value on the left of the operator is greater than the one on the right

Ex :- >>>3>4

O/P :- False

c) Less than (or) Equal to (<=) :- It checks if the value on the left of the operator is lesser than (or) equal to the one on the right.

Ex:- >>> 7<=7

O/p:- True

d) Greater than (or) equal to (>=) :-

It checks if the value on the left of the operator is greater than (or) equal to the one the right.

Ex:- >>> 0>=0

O/p:- True.

e) Equal to (==) :- if the true value on the left of the operator is equal to the one on the right. 1 is equal to the Boolean value True, but 2 is n't

16 SUNDAY

NOTES

Also, 0 is equal to false

Ex: >>> 3==3.0

O/P:- True

f) Not equal (!=0) :- If the value on the left of the operator is not equal to the one on the right. The python operator <> does the same job.

Ex:- >>> 1!=1.0

O/P:- False

③ Assignment operator :- An assignment operator assigns a value to a variable. It may manipulates the value by a factor before assignment.it

a) Assign (=) :- Assign a value to the Expression on the left. Notice that == is used for comparing, but = is used for assigning

Ex:- >>> a=7
>>> print(a)

14 NOVEMBER

O/P:- 7

b) Add and Assign (+=):- Adds the values on either side and assigns it to the Expressions on the left. a+=10 is the same as a=a+10

Ex:- >>> a+=2
>>> print (a)
O/P:- 9

c) Subtract and Assign (-=)

Subtract the value on the right from the value on the left. Then it assign it to the expression on the left.

Ex:- >>> a-=2
>>> print (a)
O/P:- 7

d) Divide and Assign (/=)

Divide the values on the left by the one on the right. Then it assign it to the expression on the left.

NOTES

Ex:- >>> a/=7

>>> print(a)

O/P:- 1.0

e) Multiply and Assign (*=)

Multiplies the values on either sides. Then it assigns it to the expression on the left

ex:- >>> a*=8

print(a)

O/P:- 8.0

f) Modulus and Assign (%=)

perform modulus on the values on either side. Then it assigns it to the expression on the left

Ex:- >>> a%=3

print(a)

O/P:- 2.0

(g) Exponent and Assign (**=)

performs exponentiation on the values on either side. Then assigns it to the expression on the left.

Ex:- >>> a **= 5
print(a)

O/P:- 32.0

(h) Floor - Divide and Assign (//=)

performs floor-division on the values on either side. Then assigns it to the expression on the left.

Ex:- >>> a//=3
print(a)

O/P:- 10.0

(i) Logical operator :-

There are conjunction that you can use to combine more than one condition. We have three python logical operator - and, or, and not that under Python operator

(a) and:- If the conditions on both sides of the operator are true, then the expression as a whole is true.

ex:- >>> a=7>7 and 2>=1
>>> print(a)
O/p:- False

(b) or:- the expression is false only if both the statement around the operator are false. otherwise, it is true

ex:- >>> a=7>7 or 2>=1
print(a)
O/p:- True.

and returns the first false value or the last value; 'or' returns the first True value or the last value

ex:- >>> 7 and 0 or 5
output:-5

(c) not:- This inverts the Boolean value of an expression. It converts True to False, and False to True. As you can see below, the Boolean value for 0 is False. So, not

inverts it to true

Ex:- >>> a=not(0)
print(a)

O/P:- True

(5) Membership operator:-

These operator test whether a value is a member of a sequence. The sequence may be a list, a string or a tuple. we have two membership Python operator "In" and "not In"

(a) In:-

This check if a value is a member of a sequence.

Ex:- Pets=['dog','cat','ferret']
'for' in pets

O/P:- False

(b) not in:- unlike 'in', 'not in' check if a value is not a member of a sequence

Ex:- 'pot' not in 'disappointment'

o/p:- True.

(6) Identity operator :-

Then operator test if the two operands share an identity. we have two identity operator - 'is' and 'is not'.

(a) is :- If two operands have the same identity, it returns True

Ex :- (1) >>> 2 is 20 (2) >>> '2' is '2'
o/p is False o/p:- True

(b) is not :- 2 is a number, and '2' is a string. so, it returns a True to that

Ex:- >>> 2 is not '2'

o/p:- True.

(7) Bitwise operator :- Operate bit by bit

(a) Binary And (&) :- It performs bit by bit AND operation on two values

Ex:- (1) 2 & 3 (2) 3 & 4
o/p:- 3 o/p:- 0

(b) Binary OR (|) :- It performs bit by bit OR on the two value

Ex :- 2 | 3

O/P :- 3

(c) Binary XOR (^) :- It performs bit by bit XOR (exclusive OR) on the two values.

ex :- 2 ^ 3

O/p :- 1

(d) Binary one's Complement (~)

It returns the one's complement of a number's binary.

Ex :- >>> ~-3

O/p : 2

(e) Binary Left-shift (<<) :-

It shifts the value of the left operand the number of places to the left that the right operand

Specifies

ex:- >>> 2<<2

o/p:- 8

(f) Binary Right-shift(>>):-

It shifts the value of the left operand the number of places to the right that the right operand specifies.

ex:- >>> 3. 3>>2

3>>1

o/p:- 1