untitled7

September 12, 2024

0.1 PYTHON OPERATORS:

What are operators in python?

In python programming, operators in general arr used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into different types of python operators.

- operators : These are the special symbols. example : -, +, *, /, etc.
- operand : It is the value on which the operater is applied.

Types of operators in python :

There are “7” types of operaters in python :

1.Arthematic operator.

2.Assignment operator.

3.Comparison operator.

4.Logical operator.

5.Identity operator.

6.Membership operator.

7.Bitwise operator.

Arithmetic operator :

Python Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, and division.

In python 3.x the result of division is a floating - point while in pyhton 2.x division of integers was an integer. To obtain an integer result in python 3.x floored (// integer) is used.

Addition : ” + ”

Subtraction : ” - ”

Multiplication : ” * ”

Division : ” / ”

Modulus : ” % ”

Exponent : ” ** ”

Integer : ” // ”

[ ]:
```
#addition
a = 5
b = 7
print(a + b)
```

12

[ ]:
```
#subtraction
a = 79
b = 19
print(a - b)
```

60

[ ]:
```
#multiplication
a = 98
b = 46
print(a * b)
```

4508

[ ]:
```
#Division
a = 45
b = 60
print(a / b)
```

0.75

[ ]:
```
#modulus
a = 100
b = 32
print(a % b)
```

4

[ ]:
```
#exponent
a = 67
b = 59
print(a ** b)
```

54753685784643014746547901871872468050559164194801265744596165473795460502968956
3388152202313966849957495803

[ ]:
```
#Integer
a = 97
b = 67
```

```
print(a // b)
```

```
1
```

Assignment operator :

The python operator are used to perform operations on value and variables. These are the special symbols that carry out arithmetic, logical, and bitwise computations. The value the operator operates on is known as the operand. Here, we will cover different assignment operators in python.

Assignment operator : " = "

Addition assignment operator : " += "

Subtraction assignment operator : " -= "

Multiplication assignment : " *= "

Division assignment : " /= "

Modulus assignment operator : " %= "

Floor Division assignment operator : " //= "

Exponentiation assignment operator : " **= "

Bitwise AND assignment operator : " &= "

Bitwise OR operator : " |= "

Bitwise XOR assignment operator : " ^= "

Bitwise Right shift assignment operator : " >>= "

Bitwise Left shift assignment operator : " <<= "

[ ]:
```
#Assignment operator
#Assignment values using
#Assignment operator

a = 6
b = 4
c = a + b

#output
print(c)
```

```
10
```

[ ]:
```
#addition assignment operator
a = 3
b = 8
a += b
print(a)
```

11

[ ]:
```
#subtraction assignment operator
a = 67
b = 56
a -= b
print(a)
```

11

[ ]:
```
#multiplication assignment
a = 99
b = 36
a *= b
print(a)
```

3564

[ ]:
```
#Division assignment
a = 78
b = 34
a /= b
print(a)
```

2.2941176470588234

[ ]:
```
#modulus assignment
a = 55
b = 29
a %= b
print(a)
```

26

[ ]:
```
#floor division assignment operator
a = 56
b = 40
a //= b
print(a)
```

1

[38]:
```
#exponentiation assignment operator
a = 76
b = 10
a **= b
print(a)
```

6428888932339941376

[ ]:
```
#Bitwise AND operator
a = 55
b = 86
a &= b
print(a)
```

22

[ ]:
```
#Bitwise OR operator
a = 30
b = 48
a  |= b
print(a)
```

62

[ ]:
```
#Bitwise XOR operator
a = 78
b = 34
a ^= b
print(a)
```

108

[ ]:
```
#Bitwise RIght operator
a = 39
b = 45
a >>= b
print(a)
```

0

[ ]:
```
#Bitwise Left shift operator
a = 91
b = 27
a <<= b
print(a)
```

12213813248

Comparision operator

The python operators can be used with various data types, including numbers, strings, booleans, and more. In python, comparison operators are used to copmare the values of two operands. When comparing steings, the comparision is based on the alphanetical order of their characters. Be cautions when comparing floating - point numbers due to potential precision issues.

Types of comparison operators in python

Python equality operator : " == "

Inequality operator : " != "

greater than : " > "

Less than : " < "

Greater than or equal to : " >= "

Less than or equal to : " <= "

[ ]:
```
#python equality operator
a = 57
b = 45
c = 57
print(a == b)
print(a == c)
```

```
False
True
```

[ ]:
```
#Inequality operator
a = 77
b = 45
c = 77
print(a != b)
print(a != c)
```

```
True
False
```

[ ]:
```
#Greater than
a = 69
b = 34
c = 69
print(a > b)
print(a > c)
```

```
True
False
```

[ ]:
```
#less than
a = 67
b = 89
c = 67
print(a < b)
print(a < c)
```

```
True
False
```

[ ]:
```
#Greater than or equal to
a = 55
b = 66
c = 55
print(a >= b)
print(a >= c)
```

```
False
True
```

[36]:
```
#less than or equal to
a = 34
b = 64
c = 39
print(a <= b)
print(a <= c)
```

```
True
True
```

Logical operator

python logical operaters are used to combine conditinal staements, allowing you to perform operations based on multiple conditions. These python operaters, alongside arithemetic operaters, are special symbols used to carry out compulations on value and varibles.

In python, Logical operaters are used on conditional statements.They perform Logical AND, LOgical OR, Logical NOT operations.

[41]:
```
#Logica AND
a = 50
b = 45
print (a>10 and b>40)
print (a>60 and b>39)
print (a>49 and b>44)
print (a>51 and b>46)
print (a<100 and b<50)
print (a>55 and b<30)
print (a!=70 and b!=60)
print (a==50 and b==45)
```

```
True
False
True
False
True
False
True
True
```

[42]:
```
#Logical OR
x = 40
y = 21
print (x>30 or y<30)
print (x<45 or y>45)
print (x!=50 or y!=20)
print (x==40 or y==21)
print (x>=50 or y<=15)
print (x<=30 or y>=24)
```

```
True
True
True
True
False
False
```

[43]:
```
#Logical NOT
a = 88
b = 67
print(not(x>90 and y<50))
print(not(x<87 and y>68))
print(not(x!=67 and y!=67))
print(not(x==88 and y==99))
print(not(x<=50 and y>=30))
print(not(x>=40 and y<=57))
```

```
True
True
False
True
True
False
```

Identity operators

The python identity operators are used to compare the objects if oth the objects are actually of the same data type and share the same memory location. There are different identity such as:

- IS Operator
- IS NOT Operator

Python IS Operator :

The IS operator evalutes to true if the variables on either side of the operator point to the same object in the memory and false otherwise.

[45]:
```
# IS OPERATOR
```

```
a = 5
b = 15
c = 20
d = 44
e = 59
x = "HI BRO"
y = "HOW ARE YOU"
z = "BE NICE"
print(a is b)
print(b is not a)
print(c is not a)
print(a is c)
print(x is y)
print(y is not z)
print(z is not y)
print(y is a)
print(c is not b)
print(x is c)
print(d is not e)
```

```
False
True
True
False
False
True
True
False
True
False
True
```

Membership Operator

The python membership operators tset for the membership of an object in a sequence, such as strings, lists, or tuples. Python offers two membership operators to check or validate the membership of a value. They are as follows:

- IN Operator
- NOT IN Operator

The IN operator returns True if the value or variable is found in the sequence, otherwise false.

The NOT IN operator returns true if the value of varible is not found in the sequence, otherwise false.

[47]:
```
a = ["apple", "pineapple", "capsicum"]
b = [1,2,3,4,5,6,7,9]
z = "faheem shaik"
```

```
print("banana" in a)
print("faheem" in z)
print(3 in b)
print(8 in b)
print("apple" in a)
print("shaik" in z)
print("Faheem" in z)
print("t" in a)
print("pineapple" in a)
print("capsicum" in z)
```

```
False
True
True
False
True
True
False
False
True
False
```

BITWISE OPERATOR

Python bitwise operator are used to perform bitwise calculations on integers. This integers are first convert into binary and then operations are performed on each bit or corresponding pair of bits, hence the name bitwise operators.

Bitwise And : " & "

Bitwise Or : " | "

Bitwise Not : " ~ "

Bitwise Xor : " ^ "

Bitwise Left shift : " << "

Bitwise Right shift : " >> "

[48]:
```
'''
The & operator compares the each bit and set to 1 if both are 1, otherwise 0
51 & 18                        276 & 28

27 =>  1 1 0 1 1          51 =>  0 1 1 0 0 1 1
18 =>  1 0 0 1 0          18  => 0 0 1 0 0 1 0
--------------------      -------------------------
18 <=  1 0 0 1 0          18  <= 0 0 1 0 0 1 0
//18 is output//             //18 is output//
'''
```

```
print(27 & 18)
print(51 & 18)
```

```
18
18
```