Python Programming

EDUCATION FOR PEACE

(ESTD-1995)

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

PYTHON OPERATORS-3

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

- Logical Operators
- Bitwise Operators
- Shift Operators

4. Logical operators

Following are the various logical operators used in Python.

1. and
2. or
3. not

You can apply these operators for boolean types and non-boolean types, but the behavior is different.

I. For boolean types:

and ➔If both arguments are True then only result is True

or ➔If at least one argument is True then result is True

not ➔complement

i) 'and' Operator for boolean type:

- If both arguments are True then only result is True.

```
print(True and True)      ➔True
print(True and False)     ➔False
print(False and True)     ➔False
print(False and False)    ➔False
```

ii) 'or' Operator for boolean type:

- If both arguments are True then only result is True.

print(True or True)	→True
print(True or False)	→True
print(False or True)	→True
print(False or False)	→False

iii) 'not' Operator for boolean type:

- Complement (or) Reverse

print(not True)	→False
print(not False)	→True

Eg :

Now we will try to develop a small authentication application with this knowledge.

- We will read user name and password from the keyboard.
- If the user name is karthi and password is sahasra, then that user is valid user otherwise invalid user.

```
userName = input('Enter User Name : ')
password =  input('Enter Password : ')


if userName == 'karthi' and password == 'sahasra':
     print('valid User')
else:
     print('invalid user')
```

```
Enter User Name : karthi
Enter Password : rgm
invalid user
```

II. For non-boolean types behavior:

Note :

- 0 means False
- non-zero means True
- empty strings, list, tuple, set, dict is always treated as False

i) X and Y

Here, X and Y are non boolean types and the result may be either X or Y but not boolean type (i.e., The result is always non boolean type only).

Rules:

- if 'X' is evaluates to false then the result is 'X'.
- If 'X' is evaluates to true then the result is 'Y'.

Eg:

print(10 and 20) ➔20

print(0 and 20) ➔0

print('karthi' and 'sahasra') ➔sahasra

print('' and 'karthi') # first argument is empty string ➔ Empty string

print(' ' and 'karthi') # first argument contains space character, so it is not empty

➔karthi

print('karthi' and '') # second argument is empty string ➔Empty String

print('karthi' and ' ') # second argument contains space character, so it is not empty

➔space

ii) X or Y

Here, X and Y are non boolean types and the result may be either X or Y but not boolean type (i.e., The result is always non boolean type only).

Rules:

- If 'X' is evaluates to false then the result is 'Y'.
- if 'X' is evaluates to true then the result is 'X'.

Eg:

print(10 or 20) ➔10

print(0 or 20) ➔20

print('karthi' or 'sahasra') ➔karthi

print('' or 'karthi') # first argument is empty string ➔karthi

print(' ' or 'karthi') # first argument contains space character, so it is not empty

➔Empty string

print('karthi' or '') # second argument is empty string ➔karthi

print('karthi' or ' ') # second argument contains space character, so it is not empty

➔karthi

iii) not X:

- Even you apply not operator for non boolean type, the result is always boolean type only.
- If X is evaluates to False then result is True otherwise False

Eg:

print(not 'karthi')	➔False
print(not '')	➔True
print(not 0)	➔True
print(not 10)	➔False

5. Bitwise Operators

- We can apply these operators bit by bit.
- These operators are applicable only for **int** and **boolean** types. By mistake if we are trying to apply for any other type then we will get Error.

Following are the various bitwise operators used in Python:

1. Bitwise and (&)
2. Bitwise or (|)
3. Bitwise ex-or (^)
4. Bitwise complement (~)
5. Bitwise left shift Operator (<<)
6. Bitwise right shift Operator(>>)

Eg:

print(10.5 & 20.6)

TypeError: unsupported operand type(s) for &: 'float' and 'float‘

print('karthi' | 'karthi')

TypeError: unsupported operand type(s) fo r |: 'str' and 'str‘

Eg:

```
print(bin(10))              ➔0b1010
print(bin(20))              ➔0b10100
print(10 & 20) # Valid      ➔0
print(10.0 & 20.0)          # In valid
                ➔TypeError: unsupported operand type(s) for &: 'float' and 'float'
```

Eg:

print(True & False) ➔False

print(True | False) ➔True

Behavior of Bitwise Operators

&	➔ If both bits are 1 then only result is 1 otherwise result is 0
|	➔ If at least one bit is 1 then result is 1 otherwise result is 0
^	➔ If bits are different then only result is 1 otherwise result is 0
~	➔ bitwise complement operator, i.e., 1 means 0 and 0 means 1
<<	➔ Bitwise Left shift Operator
>>	➔ Bitwise Right Shift Operator

Eg:

print(4 & 5) # 100 & 101 ➔4

print(4 | 5) # 100 | 101 ➔5

print(4 ^ 5) # 100 ^ 101 ➔1

Bitwise Complement Operator (~):

- We have to apply complement for total bits.

Eg:

print(~4) # 4 ==> 100 ➔-5

- Here, we have to apply complement for total bits, not for three bits (in case of 4).
- In Python minimum 32 bits required to represent an integer.

Note:

- The most significant bit acts as sign bit. 0 value represents +ve number where as 1 represents -ve value.
- Positive numbers will be represented directly in the memory where as Negative numbers will be represented indirectly in 2's complement form.

How you can find two's complement of a number?

- To find Two's complement of a number, first you need to find One's complement of that number and add 1 to it.
- One's complement ==> Interchange of 0's and 1's

Eg:

```
print(~5)          ➔-6
print(~-4) # negative values are stored in the memory in 2's complement form.
                   ➔3
```

6. Shift Operators

Following are the various shift operators used in Python:

1. Left Shift Operator (<<)
2. Right Shift Operator (>>)

1. Left Shift Operator (<<):

- After shifting the bits from left side, empty cells to be filled with zero.

Eg:

print(10<<2) ➔40

2. Right Shift Operator (>>)

- After shifting the empty cells we have to fill with sign bit.(0 for +ve and 1 for -ve)

print(10>>2) ➔2

We can apply bitwise operators for boolean types also.

Eg:

print(True & False)	➔False
print(True | False)	➔True
print(True ^ False)	➔True
print(~True)	➔-2
print(~False)	➔-1
print(True<<2)	➔4
print(True>>2)	➔0

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You