3- Operators

1. What are operators?

Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

Example 1:
>>> 2+3 5. Here, + (plus) is the **operator** that performs addition. 2 and 3 are the operands and 5 is the output of the operation.

2. Use of operators:

Operators are used to perform operations on values and variables. Operators can manipulate individual items and returns a result. The data items are referred as operands or arguments.

3. Types of operators:

- Arithmetic operator.
- Comparison operator.
- Logical operator.
- Assignment operator.
- Bitwise Operator.

3.1 Arithmetic Operator:

- Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, division, etc. This are the basic mathematics which we all know. Now let us see how it is implemented in python coding.

- List of Arithmetic Operators:

Operator	Meaning
+	Adding two operands
-	Subtraction of operands
*	Multiply two operands
/	Divide left operand by the right one
%	Modulus - remainder of the division of left operand by the right
//	Floor division – This operator will always give a integer value.

Example :

```
In [2]:
x = 5
y = 4
print('Output')
print('x + y =',x+y)   #Addition of x and y

print('x - y =',x-y)   # substraction of x and y

print('x * y =',x*y)    # Multipliaction

print('x / y =',x/y)    # Division of x by y

print('x // y =',x//y)  #Double

Output
x + y = 9
x - y = 1
x * y = 20
x / y = 1.25
x // y = 1
```

3.2 Comparison Operators:

- Comparison operators are used to compare values. It returns either *True* or *False* according to the condition.
- List of comparison operators:

Operator	Meaning	Example
>	Greater than – (True if left operand is greater than the right)	x > y
<	Less than – (True if left operand is less than the right)	x<y
==	Equal to –(True if both operands are equal)	x==y
!=	Not equal to – (True if operands are not equal)	x != y
>=	Greater than equal to – (True if left operand is greater than or equal to the right)	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Example:

```
In [5]:
1  x = 100
2  y = 120
3
4
5  print('x > y is',x>y)
6
7
8  print('x < y is',x<y)
9
10
11 print('x == y is',x==y)
12
13
14 print('x != y is',x!=y)
15
16
17 print('x >= y is',x>=y)
18
19
20 print('x <= y is',x<=y)
21
```

```
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

3.3 Logical Operators

- The logical operators in Python are used to combine the true or false values of variables (or expressions) so you can figure out their resultant truth value.
- List of Logical 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

Now let us see how it works with the help of logical truth table.

Let us assume 0 as 'False' and 1 as 'True'.

3.3.1 And operator

Truth table:

x	y	X and y
0	0	0
0	1	0
1	0	0
1	1	1

The truth table means if we have both the values as 1 (true) the output will be true(1) else false(0).

Example:

```
In [6]: 1 5 < 7 and 5 > 3
Out[6]: True

In [7]: 1 3 > 3 and 55 > 30
Out[7]: False

In [8]: 1 15 / 3 >= 200 and 3 == 3
Out[8]: False

In [9]: 1 55 == 55 and 3 <= 3
Out[9]: True
```

Let us take a look first output

- Here 5 < 7 is true(1)
- 5 > 3 is also true(1)
- Therefore as per the table if both are true then the output will be true.

Let us take a look Second output:

- Here 3 > 3 is false(0)
- 55 > 30 is true (1)
- Therefore the output will be False

3.3.2 Or Operator

Truth Table :

x	y	x or y
0	0	0
0	1	1
1	0	1
1	1	1

Here if both the values are false then the output will be false else true,

Example:

```
In [16]: 1 3 == 3 or 5 < 3
Out[16]: True

In [13]: 1 15 < 3 or 5 > 3
Out[13]: True

In [14]: 1 12 <= 1 or 5 < 1
Out[14]: False

In [15]: 1 20 + 3 >= 23 or 5 != 5
Out[15]: True
```

First output

- Here 3 == 3 is true
- < 3 is false
- Therefore output is True

3.3.3 Not operator

Truth Table:

Operand	Operator
0	1
1	0

In this table, if the value given is true then the output will be false and if the value is false the output will be true.

Example:

```
In [18]: not True
Out[18]: False

In [19]: not False
Out[19]: True

In [20]: not 5 > 3
Out[20]: False

In [21]: not (5 > 3 and 5 > 2)
Out[21]: False

In [22]: not (5 > 3 and 5 < 33)
Out[22]: False

In [23]: not (5 < 3 and 5 < 33)
Out[23]: True
```

Let's se the last example: **not (5 < 3 and 5 < 33)** and why it returns True.
Simply evaluate the expression in the parentheses first:

- **5 < 3 :** 5 isn't smaller than 3, so this expression is false.
- **5 < 33:** 5 is smaller than 33, so this expression is true.

From the truth table, for the **and operator** above, we know that False and True is False.

So the expression 5 < 3 and 5 < 33 will give False. Now, we just need to apply the **not operator**, so False becomes True.

3.4 Assignment Operators

- Assignment operators are used in Python to assign values to variables.
- a = 5 is a simple assignment operator that assigns the value 5 on the right to the variable a on the left.
- There are various compound operators in Python like a += 5 that adds to the variable and later assigns the same. It is equivalent to a = a + 5.

List of Assignment operators:

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 / 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:

In [25]:
```
a = 21
b = 10
c = 0
c //= a
print ("Value of c is ", c)
```

```
Value of c is  0
```

In [28]:
```
a = 2
b = 10
c = 25
c += a
print ("Value of c is ", c )
```

```
Value of c is  27
```

In [27]:
```
a = 21
b = 10
c = 25
c *= a
print ("Value of c is ", c )
```

```
Value of c is  525
```

3.5 Bitwise operators

a. Compliment (~)

Example:

```
In [29]: 1 ~9
Out[29]: -10
```

- ~9 will give you output as -10 Now how -10 lets see
- First we know the binary number of 9 is 1001 but we have to use (8 4 2 1) format so we will write the binary number as 0000 1001.
- We have,
  0000 1001 (binary number of 9)
- Now we will find the negation of above binary number.
- For negation 0 will become 1 and 1 will become 0.
- After negation we have 1111 0110 (2's complement) as answer which is -10.

- Now for better under standing we will calculate binary number of -10 Binary number of 10 is 0000 1010

- Now by 2's compliment 0000 1010
  1111 0101 (Negation of 10)
  + 1
  11110110 binary of -10
- This is how we calculate the complement of any number

b. Bitwise And (symbol is &)

Example:

```
In [32]: 1 10 & 12
Out[32]: 8
```

- 10 & 12 will give output as 8 Explanation
  10 – 0000 1010
  12 – 0000 1100

- Now by Truth Table of AND

  0000 1010
  0000 1100
  1000 (This is binary number of 8)

c. Bitwise OR (symbol is |)

Example:

```
In [33]: 1 10|12
Out[33]: 14
```

- 10|12 will give you output as 14 Explanation
  10 – 0000 1010
  12 – 0000 1100

- Now by Truth Table of OR
  0000 1010
  0000 1100
  1110 s binary number of 14)

d. Bitwise XOR(symbol is ^)

Truth Table:

x	y	x^y
0	0	0
0	1	1
1	0	1
1	1	0

Example:

```
In [34]: 1 10 ^ 12
Out[34]: 6
```

- 10 ^ 12 will give you output as 6 Explanation
  10 – 0000 1010
  12 – 0000 1100

- Now by Truth Table of OR 0000 1010
  0000 1100
  00000110 (This is binary number of 6)

e. Left shift (symbol is <<)

Example:

```
In [35]: 1 12<<2

Out[35]: 48
```

- 12<<2 this will give you output as 48 Binary number of 12 is 1100
- After applying left shift operator It will be shifted by two bits i.e 110000 which is equivalent to 48.

f. Right Shift (symbol is >>)

Example:

```
In [36]: 1 12>>2

Out[36]: 3
```

- 12>>2 this will give you output as 3 Binary number of 12 is 1100
- Always assume a dot after the binary digit i.e 1100.
- After applying left shift operator It will be shifted by two bits i.e 0011 which is equivalent to 3.

Video link:

1) **https://youtu.be/v5MR5JnKcZI**
2) https://youtu.be/PyfKCvHALj8

Reference:

1) https://www.programiz.com/python-programming/operators
2) https://www.journaldev.com/26737/python-bitwise-operators

s