Operators in Python

1. Arithmetic Operators

Arithmetic Operators in python are + , - ,\* ,/ ,% , // , \*\*

+ means addition

- means subtraction

\* means multiplication

/ means actual quotient

% means remainder or modular division

// means integer quotient or floor division that is it rounds off the actual quotient to lowest integer

\*\* means exponentation that is power forms

Operator Precedence and Associativity

PEMDAS

P --> Parenthisis

E --> Exponentiation

M,D --> Multiplication , Division

A,S --> Addition , Subtraction

\*\* --> 1st priority

\* , / , // , % --> 2nd priority

+ , - --> 3rd priority

Associativity(binding)

If the expression is having operators belong to same level of precedence ( or priority ) we use assosiativiy. It is of two types

a) Left Binding (left to right associativity)

Every arithmetic operator in python follows left binding except \*\* or exponentiation

Ex : - 4 \* 7

b) Right Binding (right to left associativity)

Some examples on precedence and asscociavity:-

#4 + 5 \* 6 + 10 - 16 // 4 \*\* 2 - 5 % 3 \*\* 2

# 4 + 5 \* 6 + 10 - 16 // 16 - 5 % 9

#4 + 30 + 10 - 1 - 5

# 38

print(4 + 5 \* 6 + 10 - 16 // 4 \*\* 2 - 5 % 3 \*\* 2)

#2 + 3 \* 4 \*\* 3 + 5 - 6 // 3 - 7 % 4

#2 + 3 \* 64 + 5 - 6 // 3 - 7 % 4

#2 + 192 + 5 - 2 - 3

#194

print(2 + 3 \* 4 \*\* 3 + 5 - 6 // 3 - 7 % 4)

#3 \*\* 2 + 4 % 3 \* 6 // 2 - 7 + 6 // 3 \* 2 % 2 - 5 + 9 \*\* 2

#9 + 4 % 3 \* 6 // 2 - 7 + 6 // 3 \* 2 % 2 - 5 + 81

#9 + 1 \* 6 // 2 - 7 + 6 // 3 \* 2 % 2 - 5 + 81

#9 + 6 // 2 - 7 + 6 // 3 \* 2 % 2 - 5 + 81

#9 + 3 - 7 + 2 \* 2 % 2 - 5 + 81

#9 + 3 - 7 + 4 % 2 - 5 + 81

#9 + 3 - 7 + 0 - 5 + 81

#81

print(3 \*\* 2 + 4 % 3 \* 6 // 2 - 7 + 6 // 3 \* 2 % 2 - 5 + 9 \*\* 2)

2.Relational or Comparison Operators

- > ---> Greater than

- < ---> Less than

- > = ---> Greater than or equals to

- <= ---> Less than or equals to

- = = ---> equals to

- != ---> not equals to

---> The result of any relational operation will give a boolean value (True,False)

0 --> False

1 --> True

Ex:-

print(12>45) # False

print(12<45) #True

print(12>=45) # False

print(12<=45) #True

print(12==45) # False

print(12!=12) # False

print(45==45) #True

print('thor' > 'ironman') #True

print('Thor' > 'ironman') #False

---> This is due to ASCII code point values which is invented in 1963

ASCII stands for American Standard Code for Information Interchange

ASCII Characterset:-

'a' to 'z' --> 65 to 90

'A' to 'Z' --> 97 to 122

'0' to '9' --> 48 to 57

print('a'<'A') #False

---> ord() function

It takes a character and gives the ASCII Code point values corresponds to that character

print(ord('!')) #33

print(ord('@')) #64

print(ord('#')) #35

print(ord('$')) #36

print(ord('%')) #37

print(ord('^')) #94

print(ord('&')) #38

print(ord('\*')) #42

print(ord('\_')) #95

print(ord('-')) #45

print(ord('+')) #43

print(ord(';')) #59

print(ord(':')) #58

print(ord('~')) #126

print(ord('`')) #96

print(ord('<')) #60

print(ord('>')) #62

print(ord('/')) #47

print(ord('?')) #63

print(ord('|')) #124

print(ord(' ')) #32

3.Logical Operators

and,or,not

--->and

It becomes true if both or all the operands are true

--->or

It becomes true if even one of the all operands is true

--->nor

Unary operator

It becomes true if the taken operand is false.

4.Assignment Operators

5. Bitwise Operators

We have 5 types of operators in bitwise they are

bitwise and - &

11 & 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 11& 12 | 1 | 0 | 0 | 0 |

bitwise or - |

11 | 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 11 | 12 | 1 | 1 | 1 | 1 |

bitwise xor - ^

11 ^ 12

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 11 | 1 | 0 | 1 | 1 |
| 12 | 1 | 1 | 0 | 0 |
| 11 ^ 12 | 0 | 1 | 1 | 1 |

Bitwise Left Shift - <<

A<<B

Shifts bits is A towards left for B times

It is equivalent of writing A \* 2 power B

10 <<1

10\*2^1 - -> 20

22 << 3

22\*(2^3) - -> 176

Bitwise Right Shift - >>

A<<B

Shifts bits is A towards right for B times

It is equivalent of writing A // 2 power B

10 <<1

10//2^1 - -> 5

22 >> 3

22//(2^3) - -> 2

6.Membership Operators

It checks if a value or a item is a member of a group (iterable).

Strings,range(),set,list, tuple,dictionary are fall under the category of iterable.

Int, float etc are not iterable.

300

In [3]:



*#Bitwise Operators --> bitwise and & , bitwise or | , bitwise xor ^ , left shift << and right shift <<*

print(10 **&** 20)

print(10 **|** 20)

print(10 **^** 20)

0

30

30

In [4]:



print(10**<<**20)

print(10**>>**20)

10485760

0

In [5]:



print(10**<<**2)

print(10**>>**2)

print(22**<<**3)

print(22**>>**3)

40

2

176

2

In [7]:

7.Identity Operator