Operators in python:

- Operator is one which performs some activity. There are many types of operators
  - Arithmetic Operators
  - Relational or Comparison Operators
  - Logical operators
  - Bitwise operator
  - Assignment operator
  - Special operators

1. Arithmetic Operators : there are 7 arithmetic operator supported in python.

Op	Operator	Description	Example	Output
+	Addition	Adds the operands	>>>print(a+b)	300
-	Subtraction	Subtracts the operands	>>>print(a-b)	-100
*	Multiplication	Multiplies the operands	>>>print(a*b)	20000
/	Division	Divides operands gives the quotient and result is always float.	>>>print(17/3) >>>print(12/5.0) >>>print(9/4)	5.666666666666667 2.4 2.25
%	Modulo or Remainder	Remainder after division	>>>print(5%3) >>>print(5.0%3) >>>print(5.5%3)	2 2.0 2.5
//	Floor Division	After division quotient is adjusted to the integer less than or equal to result. For float division it returns result in float. and for integer operands it return integer output.	>>>print(13//5) >>>print(13.0//5.0) >>>print(-19//5) >>>print(-20/3.0)	2 2.0 -4 -7.0
**	Exponential or power	It returns X^y where x, y are operands	>>>print(10**3) >>>print(10**-2)	10^3=1000 10^{-2}=0.01

Note : let a= 100 and b=200 for the above examples

1. + operator applicable for str type also . string concatenation operator

   'lbrce' + 3 // Error because to work + as concatenation both operands must be strings

   'lbrce' + '3' // lbrce3

   'lbrce'+str(3) // lbrce3

2. * operator can also applicable to str then it is called String multiplication operator or repetition operator.
   'lbrce' * 3 // 'lbrce lbrce lbrce'
   'lbrce' * '3' // error because one operand must be integer only.
   'lbrce' * 3.0 // error only integer is accepted as numeric argument.
   2 * 'lbrce' // 'lbrcelbrce'

3. x/0 or x%0 always return ZeroDivisionError

2. Equality & Relational Operators : relational operators also called comparison operators are used to compare two operands and determine the relation between them. Always return Boolean result.

Op	Operator	Description	Examples	Output
==	Equality	Return True if two operands are exactly equal. Compatible types are compared if we have same value it returns True.	>>>10==True >>>False==False >>>'lbrce'=='lbrce' >>>10==20==30 >>>10==5+5==3+7 >>>'a'==97 >>>(10+2j)== (10+2j) >>>10==10.0 >>>1==True	False True True False True False True True True
!=	Not Equal	Return True if two operands are not equal.	>>>print(a!=b)	True
>	Greater than	Return True if 'a' is more than 'b' where 'a' is left and 'b' is right operand.	>>>print(a>b)	False
<	Less than	Return True if 'a' is less than 'b' where 'a' is left and 'b' is right operands.	>>>print(a<b)	True
>=	Greater than or equal to	Return True if 'a' is more than or equal to 'b' where 'a' is left and 'b' is right operands.	>>>print(a>=b)	False
<=	Less than or equal to	Return True if 'a' is less than or equal to 'b' where 'a' is left and 'b' is right operands.	>>>print(a<=b)	True

Note : a=10 , b=20 for the above table

- Relational operator can also applicable to str operands also. They are compared based on alphabetical order i.e., based on their unicode values. A= 65 , and a=97
- Let a='durga' , b='ravi'

```
print("a>b is",a>b)      //False
print("a>=b is",a>=b)    //False
print("a<b is",a<b)      //True
print("a<=b is",a<=b)    //True
print("a==b is",a==b)    //False
print("a!=b is",a!=b)    //True
```

- Let a=True , b=False // True is bigger and False is smaller because True□1 and False□0

```
print("a>b is",a>b)      // True
print("a>=b is",a>=b)    //True
print("a<b is",a<b)      //False
print("a<=b is",a<=b)    //False
print("a==b is",a==b)    //False
```

print(“a!=b is”,a!=b) //True

- a=10
  b=20
  if(a>b):
  print(“ a is greater than b”)
  else :
  print(“ a is not greater than b”)

- indentation is so important in python to identify the blocks
- Chaining of relational operators
  10<20<30<40 //it compare all operators if at least one operator is False the result is False
  10<20<35>3 // True
  10<20<30<40<50>35 // False

3. Logical Operators :

For Boolean types : the result is boolean
and if both arguments are true the result is True
or if atleast one operand is true the result is True
not if the argument is True then result is False vice versa

True and False // False
True or False // True
Not True // False

For non-boolean types : the result is not boolean
0 False
Non-Zero True
Empty string False

1. x and y (x , y are not Boolean type)
if ‘x’ evaluates to false then result is ‘x’ otherwise returns ‘y’.
10 and 20 20
0 and 20 0
1 and ‘lbrce’ ‘lbrce
0 and ‘lbrce’ 0

2. x or y (x , y are not Boolean type)
if ‘x’ evaluates to True then result is ‘x’ otherwise returns ‘y’.
10 or 20 10
0 or 20 20
0 or True True

3. not x:

not 0 □ True

not 10 □ False

not ‘ ‘ □ True

4. Bitwise Operators : applicable only for ‘int’ and ‘bool’ data types.

& □ if both bits are ‘1’ then 1 otherwise 0

| □ if at least one bit is 1 then 1 otherwise 0

^ □ X-OR if both bits are different then 1 otherwise 0

~ □ bitwise complement operator , 1 □ 0 and 0 □ 1

<< □ bitwise left shift

>> □ bitwise right shift

4&5 □ 4 100

4 | 5 □5 101

& 100 □ 4 | 101 □ 5 ^ 001□ 1

4^5 □ 1

Bitwise complement operator (~) :

~4 □ -5

4 is represented as 32 bit then

4 □ 0000 0000 0000 0000 0000 0000 0000 0100

MSB □ 0 means positive number , 1 means –ve number

-ve numbers are represented in memory as 2’s complement form

~4 □ 1111 1111 1111 1111 1111 1111 1111 1011

Here MSB is 1 so it is –Ve so number is in 2’s complement form

2’s complement of remaining is 1’s complement + 1

2’s complement □ 1000 0000 0000 0000 0000 0000 0000 0100

+ 1 □ 1000 0000 0000 0000 0000 0000 0000 0101 □ -5

~True □ -2 // because True is 1 so 2’s complement of 1 + 1 = -2

Left shift operator(<<) :

- Shift the bits to left

  print(10<<2)

  right hand side vacant cells will be flled with 0’s

  10 ==> 0000 0000 0000 0000 0000 0000 0000 1010

  When left shift twice we get

  - 0000 0000 0000 0000 0000 0000 0000 0010 1000 □ 40

True << 2 □ 4

Right Shift operator (>>) :

10>>2

Left hand side vacant cells fill with sign bit, which is +ve nos □ 0 , -ve nos □ 1

10 ==> 0000 0000 0000 0000 0000 0000 0000 1010

When right shift twice we get

⇨ 0000 0000 0000 0000 0000 0000 0000 0010 □ 2

True>>2 ? 0

5. Assignment Operators :

x=10

a, b, c, d=10, 20, 30, 40 // a=10 , b=20, c=30, d=40

x += 10 ? x= x+10 // compound assignment operator

compound assignment operators in python :

- ? Increment and decrement operators are not available in python
- ? ++x is interpreted as +(+x) like a sign operator
- ? ---x is –(-(-x)) so if x=10 and ---x = -10
- ? +=, -=, *=, %=, //=, **= , &=, |=, ^=, >>=, <<= are compound assignment operators in python.

```
a=4
a &= 5
print(a)
```

Ternary Operator :

?:

X=(condition)? FirstValue : SecondValue // syntax of c

X=(10<20)?30:40

print(x) // 30 if condition is true it returns firstvalue otherwise return secondvalue

python syntax

X=firstvalue if codition else secondvalue

C=30 if 10<20 else 40 // c=30

a,b=10,20

x=30 if a>b else 40 // x=40

a,b = b,a // swaping without third argument

read two variables from keyboard and print minimum

```
a=input("Enter First Number :")  // read str value from keyboard
a=int(a)
b=int(input("enter Second Number :")
m=a if a<b else b
print(" Minimum Value : ",m)
```

x=firstvalue if condition1 else secondvalue if condition2 else thirdvalue

x=10 if 20<30 else 40 if 50<60 else 70

print(x) // 10

x=10 if 20>30 else 40 if 50<60 else 70

print(x) // 70

read three variables from keyboard and print Maximum value

```
a=int(input("Enter First Number :") )
b=int(input("enter Second Number :")
c= int(input("enter Third Number :")
m=a if a>b and a>c else b if b>c else c
print(" Maximum Value : ",m)
```

```
print("Both  Equal"  if a==b else "First > Second " if a>b else  "First < Second")
```

6. Special operators in Python : there are two special operators in python

1. Identity Operators : for address comparison we use identity operators. There are two identity operator 'is', 'is not'.

```
a=10
b=10
print(a is b)   // 'is' is an identity operator  True
 print( a is not b)   // 'is not' is opposite of 'is'  False
```

r1 is r2 true if r1 and r2 points to same object
r1 is not r2 True if r1 and r2 points to different objects

id(a) address of a

```
l1= [ 10, 20, 30]
l2= [ 10, 20, 30]
print(id(l1))
print(id(l2))
print(l1 is l2)   // False because address of both the lists are different because list is mutable object

print(l1 == l2 )   /// content comparison returns True
```

what is the difference between == and 'is' operators ? content comparison , reference comparison

2. Membership Operators : there are two member ship operators 'in', 'not in'

- L1=[10, 20, 30]

10 is member of L1 or not to check this as : print(10 in L1) // True
60 is in L1 // False

- Let s = " Hello Learning Python is very very easy !!!"

```
print("Hello" in s)   // True
print("z" is not in s)  // True
print("L" in s) // True
print("p" in s )  // False,  small 'p' not there
```

Evaluation of Expression:

Operator Precedence

1. () parenthesis
2. ** exponential
3. ~ , - Unary operator
4. *, / ,%, //
5. +, -
6. <<, >>
7. &
8. ^
9. |
10. > , >=, <, <=, ++, !=
11. Assignment operators =, +=, *=,…etc
12. is , is not
13. in , not in
14. not
15. and
16. or

Ex : a=30, b=20, c=10, d=5
print((a+b)*c/d) // answer is 100.0
(a+(b*c)/d) // 70.0 division always return float values

module : a group of functions, classes, variables etc. import the module and use the corresponding function.
Module is low level component where as library is high level component.