Operators

Operator is a symbol that performs certain operations.
Python provides the following set of operators

1. **Arithmetic Operators**
2. **Relational Operators or Comparison Operators**
3. **Logical operators**
4. **Bitwise oeprators**
5. **Assignment operators**
6. **Special operators**

1. Arithmetic Operators:

+ ==>Addition
- ==>Subtraction
*** ==>Multiplication**
/ ==>Division operator
% ===>Modulo operator

// ==>Floor Division operator

**** ==>Exponent operator or power operator**

Note: / operator always performs floating point arithmetic. Hence it will always returns float value.

But Floor division (//) can perform both floating point and integral arithmetic. If arguments are int type then result is int type. If atleast one argument is float type then result is float type.

Note:

We can use +,* operators for str type also.
If we want to use + operator for str type then compulsory both arguments should be str type only otherwise we will get error.

+====>String concatenation operator
*** ===>String multiplication operator**

Note: For any number x,

x/0 and x%0 always raises "ZeroDivisionError"

10/0
10.0/0
.....

Relational Operators:

>,>=,<,<=

Note: Chaining of relational operators is possible. In the chaining, if all comparisons returns True then only result is True. If atleast one comparison returns False then the result is False

equality operators:

== , !=

We can apply these operators for any type even for incompatible types also

Note: Chaining concept is applicable for equality operators. If atleast one comparison returns False then the result is False. otherwise the result is True.

Logical Operators:

and, or ,not

We can apply for all types.

For boolean types behaviour:

and ==>If both arguments are True then only result is True
or ====>If atleast one arugemnt is True then result is True
not ==>complement

True and False ==>False
True or False ===>True
not False ==>True

For non-boolean types behaviour:

0 means False
non-zero means True
empty string is always treated as False

x and y:

==>if x is evaluates to false return x otherwise return y
If first argument is zero then result is zero otherwise result is y

x or y:

If x evaluates to True then result is x otherwise result is y

not x:

If x is evalutates to False then result is True otherwise False

Bitwise Operators:

We can apply these operators bitwise.
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.

&,|,^,~,<<,>>

Operator	Description
&	If both bits are 1 then only result is 1 otherwise result is 0
|	If atleast 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

bitwise complement operator(~):

We have to apply complement for total bits.

Note:

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

Assignment Operators:

We can use assignment operator to assign value to the variable.

Eg:
x=10

We can combine asignment operator with some other operator to form compound assignment operator.

Eg: x+=10 ====> x = x+10

The following is the list of all possible compound assignment operators in Python

+=
-=
*=
/=
%=
//=
**=
&=
|=
^=

Ternary Operator:

Syntax:
x = firstValue if condition else secondValue

If condition is True then firstValue will be considered else secondValue will be considered.

Special operators:

Python defines the following 2 special operators

1. Identity Operators
2. Membership operators

1. Identity Operators

We can use identity operators for address comparison.
2 identity operators are available
1. is
2. is not

2. Membership operators:

We can use Membership operators to check whether the given object present in the given collection.(It may be String,List,Set,Tuple or Dict)

in ➔ Returns True if the given object present in the specified Collection
not in ➔ Retruns True if the given object not present in the specified Collection

Operator Precedence:

If multiple operators present then which operator will be evaluated first is decided by operator precedence.

Eg:
print(3+10*2) ➔ 23
print((3+10)*2) ➔ 26

The following list describes operator precedence in Python

() ➔ Parenthesis
** ➔ exponential operator
~,- ➔ Bitwise complement operator,unary minus operator
*,/,%,// ➔ multiplication,division,modulo,floor division
+,- ➔ addition,subtraction
<<,>> ➔ Left and Right Shift
& ➔ bitwise And
^ ➔ Bitwise X-OR
| ➔ Bitwise OR
>,>=,<,<=, ==, != ==>Relational or Comparison operators
=,+=,-=,*=... ==>Assignment operators
is , is not ➔ Identity Operators
in , not in ➔ Membership operators
not ➔ Logical not
and ➔ Logical and
or ➔ Logical or