

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Chapter-2\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2. 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 operators

5. Assignment operators

6. Special operators

……………………………………………………………………………………………

1. Arithmetic Operators:

+ ==>Addition

- ==>Subtraction

\* ==>Multiplication

/==>Division operator

% ===>Modulo operator

// ==>Floor Division operator

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

………………………………………………………………………………………………

Eg: test.py:

1) a=10

2) b=2

3) print('a+b=',a+b)

4) print('a-b=',a-b)

5) print('a\*b=',a\*b)

6) print('a/b=',a/b)

7) print('a//b=',a//b)

8) print('a%b=',a%b)

9) print('a\*\*b=',a\*\*b)

---------------------------------------------------------------------------------

Output:

1) Python test.py or py test.py

2) a+b= 12

3) a-b= 8

4) a\*b= 20

5) a/b= 5.0

6) a//b= 5

7) a%b= 0

8) a\*\*b= 100

……………………………………………………………………………………………

Eg:

1) a = 10.5

2) b=2

3) a+b= 12.5

4) a-b= 8.5

5) a\*b= 21.0

7) a/b= 5.25

8) a//b= 5.0

9) a%b= 0.5

10) a\*\*b= 110.25

……………………………………………………………………………………………..

Eg:

10/2==>5.0   
10//2==>5

10.0/2===>5.0

10.0//2===>5.0

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 at least 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.

………………………………………………………………………………………………

1) >>> "Prasanna"+10

Ans:-TypeError: must be str, not int

3) >>> "Prasanna"+"10"

Ans:- 'Prasanna10'

If we use \* operator for str type then compulsory one argument should be int and other argument should be str type.

2\*"Prasanna"

"Prasanna"\*2

2.5\*"Prasanna" ==>TypeError: can't multiply sequence by non-int of type 'float'

" Prasanna "\*" Prasanna "==>TypeError: can't multiply sequence by non-int of type 'str'

+====>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:

……………………………………………………

>,>=,<,<=

…………………………………………………………………………………………..

Eg 1:

1) a=10

2) b=20

3) print("a > b is ",a>b)

4) print("a >= b is ",a>=b)

5) print("a < b is ",a<b)

6) print("a <= b is ",a<=b)

………………………………………………………………………………..

O/P:-

………………….

8) a > b is False

9) a >= b is False

10) a < b is True

11) a <= b is True

We can apply relational operators for str types also

………………………………………………………………………………………………Eg 2:

1) a=" Prasanna "

2) b=" Prasanna "

3) print("a > b is ",a>b)

4) print("a >= b is ",a>=b)

5) print("a < b is ",a<b)

6) print("a <= b is ",a<=b)

O/P:-

a > b is False

a >= b is True

a < b is False

a <= b is True

Eg:

1. print(True>True)

False

2) print(True>=True)

True

1. print(10 >True)

True

1. print(False > True)

False

5) print(10>' Prasanna ')

TypeError: '>' not supported between ninstances of 'int' and 'str'

……………………………………………………………………………………………

Eg:

1) a=10

2) b=20

3) if(a>b):

print("a is greater than b")

else:

print("a is not greater than b")

………………………………………………………………………..

Output:-a is not greater than b

……………………………………………………………………………………………

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

……………………………………………………………………………………………

Eg:

1) 10<20 ==>True

2) 10<20<30 ==>True

3) 10<20<30<40 ==>True

4) 10<20<30<40>50 ==>False

………………………………………………………………………………………………

\*\*\*\*\*\*\*\*\*\*\*\*equality operators: == , != \*\*\*\*\*\*\*\*\*\*\*\*\*\*:-

……………………………………………………………………………………………

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

1. >>> 10==20 ==>False
2. >>> 10!= 20🡺 True
3. >>> 10==True 🡺 False
4. >>> False==False🡺 True
5. >>> " prasanna"==" prasanna "🡺 True
6. >>> 10==" prasanna " 🡺 False

…………………………………………………………………………………………..

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

……………………………………………………………………………………………

Eg:

1) >>> 10==20==30==40 🡺 False

2) >>> 10==10==10==10 🡺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 argument 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

……………………………………………………………………………………………..

Eg:

10 and 20

0 and 20

…………………………………………………………………………………………..

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

10 or 20 ==> 10

0 or 20 ==> 20

……………………………………………………………………………………………..

not x:

……………………………………………………………………………………………

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

………………………………………………………………………………………….

not 10 ==>False

not 0 ==>True

…………………………………………………………………………………………..

Eg:

1) " prasanna" and " prasannasoft" ==> prasannasoft

2) "" and " prasanna" ==>""

3) " prasanna" and "" ==>""

4) "" or " prasanna" ==>" prasanna"

5) " prasanna " or ""==>" prasanna"

6) not ""==>True

7) not "prasanna" ==>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.

…………………………………………………………………………………………….

&,|,^,

………………………………………………………………………………………

print(4&5) ==>valid

print(10.5 & 5.6) ==>

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

………………………………………………………………………………………..

print(True & True) ==>valid

………………………………………………………………………………………..

& ==> 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

print(4&5) ==>4

print(4|5) ==>5

print(4^5) ==>1

……………………………………………………………………………………………

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

……………………………………………………………………………………………

\*\*\*\*\*\*\*\*\*\*Assignment Operators:-\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:-

…………………………………………………………………………………………..

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

Eg: x=10

We can combine assignment 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

->+=

-> -=

->\*=

->/=

-> %=

->//=

-> \*\*=

->&=

->|=

->^=

->-->>=

🡪<<=

Eg:

1) x=10

2) x+=20

3) print(x) ==>30

Eg:

1) x=10

2) x&=5

3) print(x) ==>0

……………………………………………………………………………………………

\*\*\*\*\*\*\*\*\*\*\*Ternary Operator:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:-

……………………………………………………………………………………………

Syntax: x = firstValue if condition else secondValue

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

Eg 1:

1) a,b=10,20

2) x=30 if a<b else 40

3) print(x) #30

Eg 2: Read two numbers from the keyboard and print minimum value

1) a=int(input("Enter First Number:"))

2) b=int(input("Enter Second Number:"))

3) min=a if a<b else b

4) print("Minimum Value:",min)

Output: Enter First Number:10

Enter Second Number:30

Minimum Value: 10

………………………………………………………………………………………………

Note: Nesting of ternary operator is possible.

Q. Program for minimum of 3 numbers

1) a=int(input("Enter First Number:"))

2) b=int(input("Enter Second Number:"))

3) c=int(input("Enter Third Number:"))

4) min=a if a<b and a<c else b if b<c else c

5) print("Minimum Value:",min)

………………………………………………………………………………………………

Q. Program for maximum of 3 numbers

1) a=int(input("Enter First Number:"))

2) b=int(input("Enter Second Number:"))

3) c=int(input("Enter Third Number:"))

4) max=a if a>b and a>c else b if b>c else c

5) print("Maximum Value:",max)

………………………………………………………………………………………………

Eg:

1) a=int(input("Enter First Number:"))

2) b=int(input("Enter Second Number:"))

3) print("Both numbers are equal" if a==b else "First Number is Less than Second Number" if a<b else "First Number Greater than Second Number")

………………………………………………………………………………………………

Output: D:\python\_classes>py test.py

Enter First Number:10

Enter Second Number:10

Both numbers are equal

…………………………………………………………………………………….

D:\python\_classes>py test.py

Enter First Number:10

Enter Second Number:20

First Number is Less than Second Number

…………………………………………………………………………………..

D:\python\_classes>py test.py

Enter First Number:20

Enter Second Number:10

First Number Greater than Second Number

………………………………………………………………………………………………

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*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

Note:-r1 is r2 returns True if both r1 and r2 are pointing to the same object r1 is not r2 returns True if both r1 and r2 are not pointing to the same object

…………………………………………………………………………………………..

Eg:

1) a=10

2) b=10

3) print(a is b) -🡪 True

4) x=True

5) y=True

6) print( x is y)-🡪True

…………………………………………………………………………………………….

Eg:

1) a="prasanna"

2) b="prasanna"

3) print(id(a))

4) print(id(b))

5) print(a is b)

……………………………………………………………………………………………

Eg:

1) list1=["one","two","three"]

2) list2=["one","two","three"]

3) print(id(list1))

4) print(id(list2))

5) print(list1 is list2) 🡪 False

6) print(list1 is not list2)🡪 True

7) print(list1 == list2)🡪 True

……………………………………………………………………………………………

Note: We can use is operator for address comparison where as == operator for content comparison.

…………………………………………………………………………………………..

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

…………………………………………………………………………………………….

Eg:

1) x="hello learning Python is very easy!!!"

2) print('h' in x) 🡪 True

3) print('d' in x)🡪 False

4) print('d' not in x)🡪 True

5) print('Python' in x)🡪 True

………………………………………………………………………

Eg:

1) list1=["sunny","bunny","chinny","pinny"]

2) print("sunny" in list1) True

3) print("prasanna" in list1) False

4) print("prasanna" not in list1) True

………………………………………………………………………………………….

\*\*\*\*\*\*\*\*\*\*\*\*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

& 🡪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

……………………………………………………………………………………………

Eg:

1) a=30

2) b=20

3) c=10

4) d=5

5) print((a+b)\*c/d) 🡪100.0

6) print((a+b)\*(c/d))--🡪100.0

7) print(a+(b\*c)/d) 🡪70.0

9)3/2\*4+3+(10/5)\*\*3-2

11) 3/2\*4+3+2.0\*\*3-2

12) 3/2\*4+3+8.0-2

13) 1.5\*4+3+8.0-2

14) 6.0+3+8.0-2

15) 15.0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*Chapter-3\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*:-

……………………………………………………………………………………………

\*\*\*\*\*\*\*\*\* Input and Output Statements\*\*\*\*\*\*\*\*\*\*\*:-