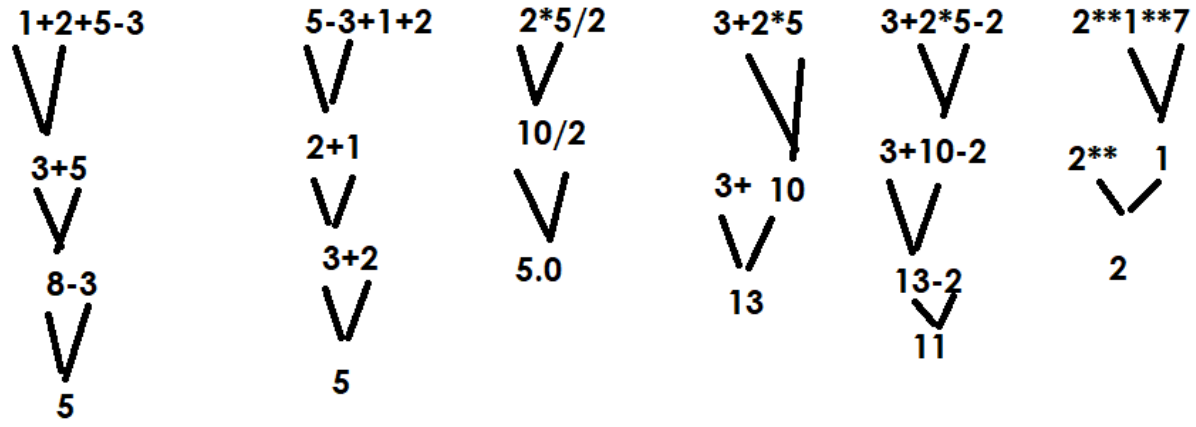
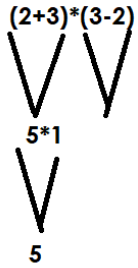
Precedence of Arithmetic Operators

**	Exponent
*,/,//,%	Multiply,float div,floor div, modulo
+,-	Addition,subtraction

The operators within same box are given same precedence
The operators within same box are evaluated from left to right
But ** operator is evaluated from right to left



Precedence of operator can be change using (), () is given highest precedence.



Relational Operators

Relational Operators are used to compare values.
Relational Operators are binary operators and required 2 operands

Relational operators after comparing values it returns boolean value (True/False)
An expression having relational operators is called boolean expression.

Operator	**Description**
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

In python relational operator are used with scalar data types and collection data types.

```
>>> 10>5
True
>>> 10>20
False
>>> 10>=10
True
>>> 10>=20
False
>>> 10>=5
True
>>> 10<5
False
>>> 5<10
True
>>> 5<=5
True
>>> 5<=7
```

```
True
>>> 5<=3
False
```

How to find ASCII value of input character?
ord() is a predefined function which returns ASCII value of input character.
Syntax: ord(char)

How to find character value of input ASCII value?
chr() is a predefined function which returns character value of input ASCII value

Syntax: chr(ascii-value)

```
>>> 'A'<'B'
True
>>> 'B'>'A'
True
>>> ord('A')
65
>>> ord('B')
66
>>> ord('C')
67
>>> ord('a')
97
>>> ord('b')
98
>>> ord('z')
122
>>> 'b'>='a'
```

```
True
>>> chr(65)
'A'
>>> chr(66)
'B'
>>> chr(97)
'a'
>>> chr(98)
'b'
```

Example:

```
# Write a program to input uppercase letter and
# convert into lowercase

s=input("Input Alphabet in Uppercase :")
s1=chr(ord(s)+32)
print(s,s1)
```

Example:

```
# Write a program to input lowercase letter and
# convert into uppercase

s=input("Input alphabet in lowercase :")
s1=chr(ord(s)-32)
print(s,s1)
```

Example:

```
>>> n1=10
>>> n1
10
>>> n1==10
```

```
True
>>> n1=20
>>> n1
20
>>> n1==10
False
>>> "naresh"=="naresh"
True
>>> "naresgh"=="NARESH"
False
>>> 10!=20
True
>>> 10!=10
False
```

Example:

```
>>> a=10
>>> b=5
>>> c=2
>>> a>b>c
True
>>> b>a>c
False
>>> b<a>c
True
```

Example:

```
# Write an boolean expression to find input value
# is between 1 and 10pow10

num=int(input("Enter any value "))
b=1<=num<=10**10
```

print(b)

Output
Enter any value 10
True

Enter any value -10
False

Enter any value 10000000000000000000000000000000000000
False

Logical Operators

Logical operators are used to combine two or more boolean expressions.
Logical operators are represented using 3 keywords

1. and
2. or
3. not

and operator

"and" is a keyword which represents logical and operator in python
Truth table of "and" operator, it is a binary operator and required 2 operands to perform operation.

Opr1	**Opr2**	**Opr1 and Opr2**
True	True	True

True	False	False
False	True	False
False	False	False

```
>>> True and True
True
>> True and False
False
>>> False and True
False
>>> False and False
False
>>> 10>2 and 10>5
True
>>> 10>2 and 10>20
False
>>> 10<2 and 10>5
False
>>> False and True
False
>>> True and False
False
>>> True and True
True
>>> True and False and True
False
>>> 100 and 200
200
>>> 0 and 100
0
>>> 100 and 0
0
```

```
>>> 100 and 200 and 300
300
>>> 100 and 0 and 300
0
>>> "python" and "java"
'java'
>>> "" and "python"
''
```

or operator

“or” keyword represents logical operator in python
Truth table of “or” operator

Opr1	**Opr2**	**Opr1 or Opr2**
True	True	True
True	False	True
False	True	True
False	False	False

```
>>> True or True
True
>>> False or True
True
>>> True or False
True
>>> False or False
False
>>> 100 or 200
100
>>> 0 or 200
200
```

```
>>> 0 and 200
0
>>> 100 or 200 or 300
100
>>> "python" or "java"
'python'
>>> "" or "python"
'python'
```

Precedence of logical operators

1. not
2. and
3. or

```
>>> 100 and 200 or 300
200
>>> 100 or 200 and 300
100
>>> 100 and 200 or 0
200
```

not operator

“not” keyword which represents logical operator in python

Truth table of “not” operator

Opr1	not opr1
True	False
False	True

```
>>> not True
```

```
False
>>> not False
True
>>> not 10>20
True
>>> not 20>10
False
>>> 100 and 200 and not 300
False
>>> not 0
True
>>> not 100
False
```