3> "*" This operator is used to perform two operations.

1> Multiplication.

2> Repeating sequence number of times.

Examples:-

Eg:- n1 = 5
    n2 = 2
    n3 = n1 * n2
    print (n1, n2, n3)
    O/p   5 2 10

Eg:- f1 = 1.5
    f2 = 2.5
    f3 = f1 * f2
    print (f1, f2, f3)
    O/p 1.5 2.5 3.25

Eg:- n1 = 5 * 1.5
    >>) print (n1)
    O/p 7.5

Eg:- n2 = 5 * true
    >>) print (n2)
    O/p 5

             sequence (must be)
                      integer (must be).
Eg:- list1 = [0].
    >>> list1 = list1 * 10
    >>> print (list1)
    O/p :- [0, 0, 0, 0, 0, 0, 0, 0, 0, 0].

Eg:- S1 = "abc"
    S2 = 5 * S1
      print (S2)

abababababab.

    >>> print ("-" * 30)
    ...

    >>> print (30 * "*")

    >>> list2 = [5] * 20
    >>> print (list2)

[5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5,
  5, 5, 5, 5, 5, 5, 5, 5, 5]

WAP to find area of rectangle.
area = l*b.

Soln:-
```
l = float(input("enter value of l"))
b = float(input("enter value of b"))
area = l*b
print(f' area of rectangle with l={l:.2f} and b={b:.2f} is
        { area:.2f}')
```
→ 2 decimal places.

O/p:- enter value of l 1.2
      enter value of b 1.5
      area of rectangle with l = 1.20 and b = 1.50 is 1.80.

/ division operator or float division operator.
This operator divide two numbers and return value as
float type.

Example:① q1 = 5/2
>>> print(q1)
→ 2.5

② >>> q2 = 4/2
>>> print(q3)
→ 2.0

③ q3 = 4/2.0
>>> print(q3)
→ 2.0

④ q4 = 4/0
Zero Division Error : division by zero.

⑤ q5 = 0/5
>>> print(q5)
0.0

W.A.P to find simple interest.

$$\# \; Si = \underset{\text{simple interest}}{P t r / 100} \; \underset{\text{time}}{\text{rate.}}$$

pxiciple amount

Soln :- P float (input ("Amount"))
   t = int (input ("time"))
   r = float (input ("rate"))

$$Si = Ptr/100$$

doln:- P = float (input("Amount"))
   t = int (input ("Time"))
   r = float (input ("Rate"))

$$Si = ptr/100.$$

print(f""" Amount {P}
   Time {t}
   Rate {r}
   Simple Interest { si:. 2f}""")

Examples:# W.A.P to input roolno, name, 3 subject marks
   # calculate total and avg marks.

roll no = int (input ("Enter the roll no"))
name = input (" enter the name")
subj1 = ~~int~~ int (input ("enter the subj1 Marks"))
Sub2 = ~~input~~ int (" (" " " " sub2 " "))
Sub3 = int (" (" " " subj3 "))

total = sub1 + sub 2+ sub3

avg = total/3.

print (f""" Roll no {roll no} \t Name {name}
   subj1 {subj1} \t Subj2 {subj2} \t subj3 {subj3}
   Total {total} \ Avg {avg :. 2f}""")

Q:-

```
a = int (input())
b = int (input())
print (a+b)
print (a-b)
print (a*b)
```

// floor division operator or integer division.

① >>> r1 = 5/2
>>> print(r1)
2

③ r3 = 5//2.0
>>> print(r3)
2.0

② r2 = 5/2
>>> print(r2)
2.5

④ r4 = 5/2.0
>>> print(r4)
2.5

Examples:- # W.A.P to delete last digit of number.

```
num = 456
print(num)
num = num//10
print(num)
```

O/P ⟹ 456
45

% Modulo operator:

This operator divides two numbers and return remainder.

♡ >>> rem1 = 5%3
>>> print(rem1)
2

⚡ rem2 = 4%2
>>> print(rem2)
0.

Examples:- # W.A.P to read last digit of number.

```
num = 456
last digit = num % 10
print (num)
print (last digit)
```

o/p ⟹ 456
       6

** Power of operator or exponent operator.

(1)
```
>>> res1 = 5 ** 2
>>> print (num)
    print (last digit)
>>> print ( res1)
    25
```

o/p 456

(2)
```
res2 = 5 ** 0
>>> print ( res2)
    1.
```

(3)
```
res 3 = 10 ** -1
>>> print (res3)
    0.1.
```

Precedence of Operators:

The following table summarises the operator precedence in python, from heigest precednce (most binding) to lowest precedence (least binding). Opeators in the same box have the same precedence. (unless the syntax is explicitly given, operators are binary. Operators in the same box group left to right (except for exponentiation and conditional expressions, which group from right to left).

Operator	Description
(expression...), [expressions...], {key: value...}, {expressions...}	Binding or parenthesized expression, list display, dictionary display, set display.
X[index], X[index:index], X(arguments...), X.attribute	Subscription, slicing, call, attribute reference.
await X	Await expression.
**	Exponentiation <u>5</u>
+x, -X, ~X	positive, -ve, bitwise NOT.
*, @, /, //, %	Multiplication, matrix multiplication, division, floor division, remainder.
+, -	Addition and substraction.
<<, >>	shifts.
&	Bitwise AND
^	Bitwise XOR.
|	Bitwise OR.
in, not in, is, is not, <, <=, 7, >	comparisons, including membership tests and identity tests.
not X	Boolean NOT.
and	Boolean AND
or	Boolean OR

if - else	conditional expression
lambda	Lambda expression.
:=	Assignment expression.