Implicit conversion
Arithmetic operations done on different data types, python return result in broader type.

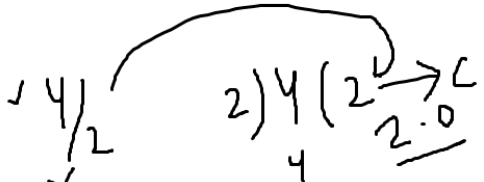
1. Int → III
2. Float ➔ II
3. Complex ➔ I

complex>float>int
int+int ➔ int
int+float ➔ float
int+float+complex ➔ complex
float+complex ➔ complex
float+float+float ➔ complex
complex+complex ➔ complex

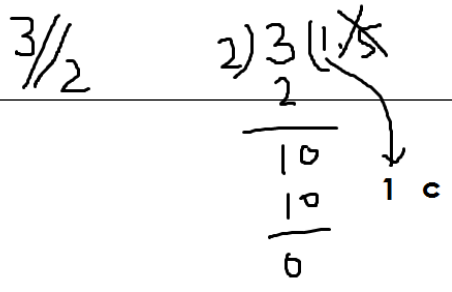
Example:

```
>>> 10+1.5
11.5
>>> 10+20
30
>>> 1.5+2.5
4.0
>>> 1+2j+1+1j
(2+3j)
>>> 1+1+2j
(2+2j)
>>> 1j+1+2
(3+1j)
>>> 1.5+2j+1
(2.5+2j)
>>> True+1+2j+1.5
(3.5+2j)
>>> int(1+2.5)
3
>>>
```

/	Float division operator, it will divide two number and return result in float type



	>>> res1=5/2
	>>> print(res1)
	2.5
	>>> res2=4/2
	>>> print(res2)
	2.0
	>>> res3=4/0
	Traceback (most recent call last):
	File "<pyshell#13>", line 1, in <module>
	res3=4/0
	ZeroDivisionError: division by zero
	>>>
	
	Example:
	# write a program to find simple interest
	# ptr/100
	
	amt=float(input("Enter amount"))
	t=int(input("Enter time"))
	r=float(input("Enter rate"))
	si=amt*t*r/100
	print("Simple Interest",si)

//	Floor division operator This operator divide two numbers and return result in integer 

	>>> res=5//2
	>>> print(res)
	2
	>>> res=5.0//2
	>>> print(res)
	2.0
	>>> res=5.0/2
	>>> print(res)
	2.5
	>>> res=5.0//2
	>>> print(res)
	2.0
	>>>
	
	The result is always rounded towards minus infinity: 1//2 is 0, (-1)//2 is -1, 1//(-2) is -1, and (-1)//(-2) is 0.

%	Modulo operator, this operator divide two numbers and gets remainder
	>>> res=5%2
	>>> print(res)
	1
	>>> res=4%2
	>>> print(res)
	0
	>>>

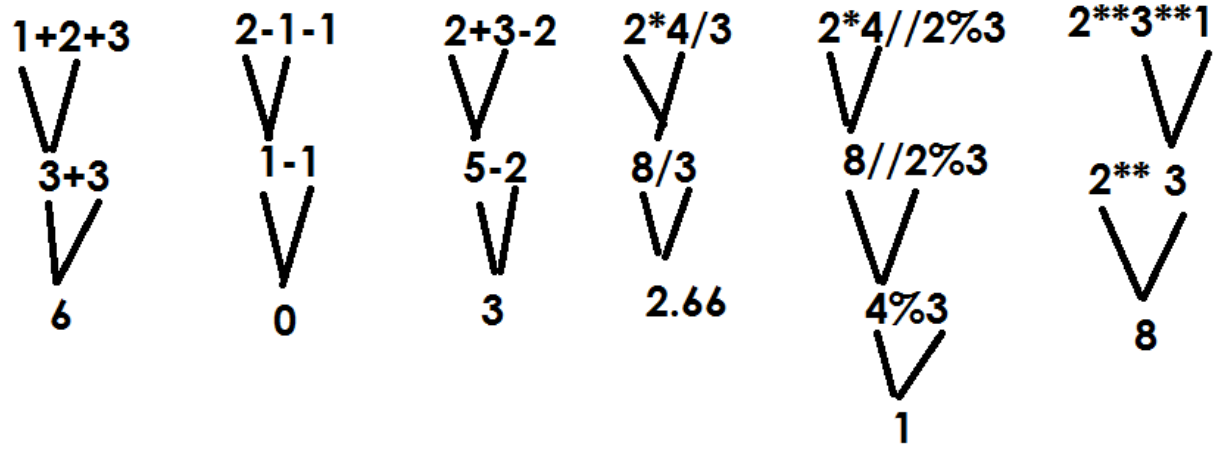
**	Exponent operator or power of operator
	>>> 5**2
	25
	>>> 5**-2
	0.04
	>>>

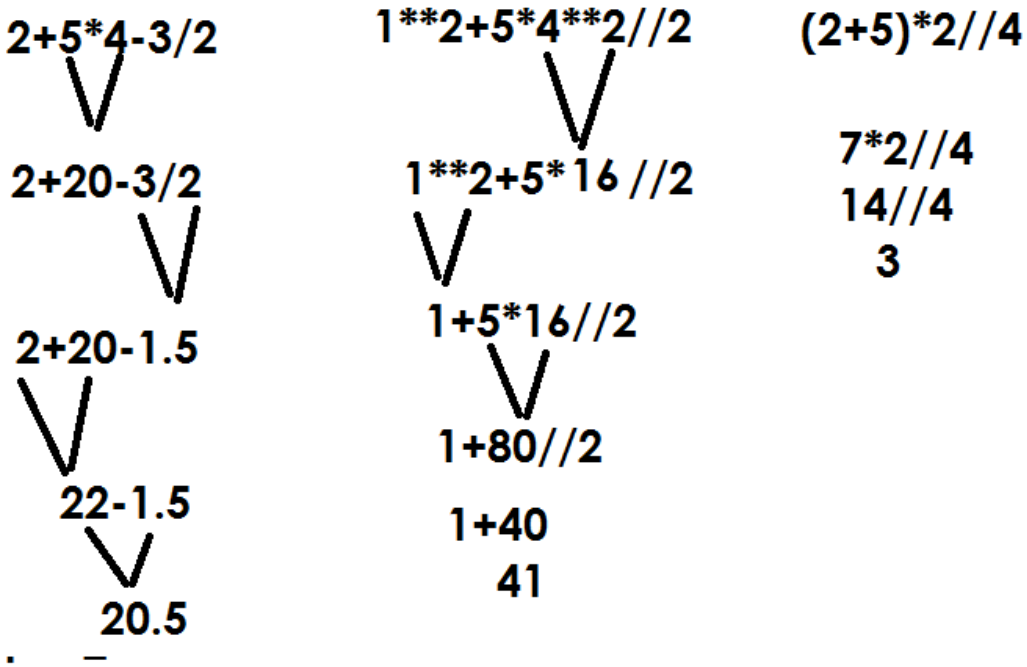
Operator precedence

Operator precedence defines order of executing operators.

The following table summarizes the operator precedence in Python, from highest precedence (most binding) to lowest precedence (least binding). Operators 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, which groups from right to left).

Operator	Description
(expressions...), [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
+x, -x, ~x	Positive, negative, bitwise NOT
*, @, /, //, %	Multiplication, matrix multiplication, division, floor division, remainder
+, -	Addition and subtraction
<<, >>	Shifts
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
in, not in, is, is not, <, <=, >, >=, !=, ==	Comparisons, including membership tests and identity tests
not x	Boolean NOT
and	Boolean AND
or	Boolean OR
if – else	Conditional expression
lambda	Lambda expression
:=	


1+2+3
3+3
6
2-1-1
1-1
0
2+3-2
5-2
3
2*4/3
8/3
2.66
2*4//2%3
8//2%3
4%3
1
2**3**1
2** 3
8
2**1**3
2** 1 2


2+5*4-3/2
2+20-3/2
2+20-1.5
22-1.5
20.5
1**2+5*4**2//2
1**2+5*16 //2
1+5*16//2
1+80//2
1+40
41
(2+5)*2//4
7*2//4
14//4
3