Python - Arithmetic Operators

Python Arithmetic Operators

Python arithmetic operators are used to perform mathematical operations such as addition, subtraction, multiplication, division, and more on numbers. Arithmetic operators are binary operators in the sense they operate on two operands. Python fully supports mixed arithmetic. That is, the two operands can be of two different number types. In such a situation.

Types of Arithmetic Operators

Following is the table which lists down all the arithmetic operators available in Python:

Operator	Name	Example
+	Addition	a + b = 30
-	Subtraction	a – b = -10
*	Multiplication	a * b = 200
/	Division	b / a = 2
%	Modulus	b % a = 0
**	Exponent	a**b =10**20
//	Floor Division	9//2 = 4

Let us study these operators with examples.

Learn **Python** in-depth with real-world projects through our **Python certification course**. Enroll and become a certified expert to boost your career.

Addition Operator

The addition operator represents by + symbol. It is a basic arithmetic operator. It adds the two numeric operands on the either side and returns the addition result.

Example to add two integer numbers

In the following example, the two integer variables are the operands for the "+" operator.

```
a=10
b=20
print ("Addition of two integers")
print ("a =",a,"b =",b,"addition =",a+b)
```

It will produce the following **output** –

```
Addition of two integers
a = 10 b = 20 addition = 30
```

Example to add integer and float numbers

Addition of integer and float results in a float.

```
a=10
b=20.5
print ("Addition of integer and float")
print ("a =",a,"b =",b,"addition =",a+b)
```

It will produce the following **output** –

```
Addition of integer and float
a = 10 b = 20.5 addition = 30.5
```

Example to add two complex numbers

The result of adding float to complex is a complex number.

```
a=10+5j
b=20.5
print ("Addition of complex and float")
print ("a=",a,"b=",b,"addition=",a+b)
```

It will produce the following **output** –

```
Addition of complex and float
a= (10+5j) b= 20.5 addition= (30.5+5j)
```

Subtraction Operator

The subtraction operator represents by - symbol. It subtracts the second operand from the first. The resultant number is negative if the second operand is larger.

Example to subtract two integer numbers

First example shows subtraction of two integers.

</> Open Compiler

```
a=10
b=20
print ("Subtraction of two integers:")
print ("a =",a,"b =",b,"a-b =",a-b)
print ("a =",a,"b =",b,"b-a =",b-a)
```

Result –

```
Subtraction of two integers
a = 10 b = 20 a-b = -10
a = 10 b = 20 b-a = 10
```

Example to subtract integer and float numbers

Subtraction of an integer and a float follows the same principle.

</> Open Compiler

```
a=10
b=20.5
print ("subtraction of integer and float")
```

```
print ("a=",a,"b=",b,"a-b=",a-b)
print ("a=",a,"b=",b,"b-a=",b-a)
```

It will produce the following **output** –

```
subtraction of integer and float
a= 10 b= 20.5 a-b= -10.5
a= 10 b= 20.5 b-a= 10.5
```

Example to subtract complex numbers

In the subtraction involving a complex and a float, real component is involved in the operation.

</> Open Compiler

```
a=10+5j
b=20.5
print ("subtraction of complex and float")
print ("a=",a,"b=",b,"a-b=",a-b)
print ("a=",a,"b=",b,"b-a=",b-a)
```

It will produce the following **output** –

```
subtraction of complex and float
a= (10+5j) b= 20.5 a-b= (-10.5+5j)
a= (10+5j) b= 20.5 b-a= (10.5-5j)
```

Multiplication Operator

The * (asterisk) symbol is defined as a multiplication operator in Python (as in many languages). It returns the product of the two operands on its either side. If any of the operands negative, the result is also negative. If both are negative, the result is positive. Changing the order of operands doesn't change the result

Example to multiply two integers

</> Open Compiler

```
a=10
b=20
print ("Multiplication of two integers")
print ("a =",a,"b =",b,"a*b =",a*b)
```

It will produce the following **output** –

```
Multiplication of two integers
a = 10 b = 20 a*b = 200
```

Example to multiply integer and float numbers

In multiplication, a float operand may have a standard decimal point notation, or a scientific notation.

</> Open Compiler

```
a=10
b=20.5
print ("Multiplication of integer and float")
print ("a=",a,"b=",b,"a*b=",a*b)

a=-5.55
b=6.75E-3
print ("Multiplication of float and float")
print ("a =",a,"b =",b,"a*b =",a*b)
```

It will produce the following **output** –

```
Multiplication of integer and float
a = 10 b = 20.5 a-b = -10.5
Multiplication of float and float
a = -5.55 b = 0.00675 a*b = -0.037462499999999996
```

Example to multiply complex numbers

For the multiplication operation involving one complex operand, the other operand multiplies both the real part and imaginary part.

```
a=10+5j
b=20.5
print ("Multiplication of complex and float")
print ("a =",a,"b =",b,"a*b =",a*b)
```

Open Compiler

It will produce the following **output** –

```
Multiplication of complex and float
a = (10+5j) b = 20.5 a*b = (205+102.5j)
```

Division Operator

The "/" symbol is usually called as forward slash. The result of division operator is numerator (left operand) divided by denominator (right operand). The resultant number is negative if any of the operands is negative. Since infinity cannot be stored in the memory, Python raises ZeroDivisionError if the denominator is 0.

The result of division operator in Python is always a float, even if both operands are integers.

Example to divide two numbers

```
a=10
b=20
print ("Division of two integers")
print ("a=",a,"b=",b,"a/b=",a/b)
print ("a=",a,"b=",b,"b/a=",b/a)
```

Open Compiler

It will produce the following **output** –

```
Division of two integers
a= 10 b= 20 a/b= 0.5
a= 10 b= 20 b/a= 2.0
```

Example to divide two float numbers

In Division, a float operand may have a standard decimal point notation, or a scientific notation.

```
a=10
b=-20.5
print ("Division of integer and float")
print ("a=",a,"b=",b,"a/b=",a/b)
a=-2.50
b=1.25E2
print ("Division of float and float")
print ("a=",a,"b=",b,"a/b=",a/b)
```

It will produce the following **output** –

```
Division of integer and float
a= 10 b= -20.5 a/b= -0.4878048780487805
Division of float and float
a= -2.5 b= 125.0 a/b= -0.02
```

Example to divide complex numbers

When one of the operands is a complex number, division between the other operand and both parts of complex number (real and imaginary) object takes place.

```
a=7.5+7.5j
b=2.5
print ("Division of complex and float")
print ("a =",a,"b =",b,"a/b =",a/b)
print ("a =",a,"b =",b,"b/a =",b/a)
```

It will produce the following **output** –

```
Division of complex and float
a = (7.5+7.5j) b = 2.5 a/b = (3+3j)
a = (7.5+7.5j) b = 2.5 b/a = (0.16666666666666666-0.16666666666666666j)
```

If the numerator is 0, the result of division is always 0 except when denominator is 0, in which case, Python raises ZeroDivisionError wirh Division by Zero error message.

</> Open Compiler

```
a=0
b=2.5
print ("a=",a,"b=",b,"a/b=",a/b)
print ("a=",a,"b=",b,"b/a=",b/a)
```

It will produce the following **output** –

```
a= 0 b= 2.5 a/b= 0.0
Traceback (most recent call last):
  File "C:\Users\mlath\examples\example.py", line 20, in <module>
    print ("a=",a,"b=",b,"b/a=",b/a)
                          ~^~
ZeroDivisionError: float division by zero
```

Modulus Operator

Python defines the "%" symbol, which is known aa Percent symbol, as Modulus (or modulo) operator. It returns the remainder after the denominator divides the numerator. It can also be called Remainder operator. The result of the modulus operator is the number that remains after the integer quotient. To give an example, when 10 is divided by 3, the quotient is 3 and remainder is 1. Hence, 10%3 (normally pronounced as 10 mod 3) results in 1.

Example for modulus operation on integers

If both the operands are integer, the modulus value is an integer. If numerator is completely divisible, remainder is 0. If numerator is smaller than denominator, modulus is equal to the numerator. If denominator is 0, Python raises ZeroDivisionError.

</> Open Compiler

```
a=10
b=2
print ("a=",a, "b=",b, "a%b=", a%b)
a=10
b=4
print ("a=",a, "b=",b, "a%b=", a%b)
print ("a=",a, "b=",b, "b%a=", b%a)
a=0
b=10
print ("a=",a, "b=",b, "a%b=", a%b)
print ("a=", a, "b=", b, "b%a=",b%a)
```

It will produce the following **output** –

```
a= 10 b= 2 a%b= 0
a= 10 b= 4 a%b= 2
a= 10 b= 4 b%a= 4
a= 0 b= 10 a%b= 0
Traceback (most recent call last):
  File "C:\Users\mlath\examples\example.py", line 13, in <module>
    print ("a=", a, "b=", b, "b%a=",b%a)
                              ~^~
ZeroDivisionError: integer modulo by zero
```

Example for modulus operation on floats

If any of the operands is a float, the mod value is always float.

</> Open Compiler

```
a=10
b=2.5
print ("a=",a, "b=",b, "a%b=", a%b)
a=10
b=1.5
print ("a=",a, "b=",b, "a%b=", a%b)
a=7.7
b=2.5
print ("a=",a, "b=",b, "a%b=", a%b)
a=12.4
```

```
b=3
print ("a=",a, "b=",b, "a%b=", a%b)
```

It will produce the following **output** –

```
a= 10 b= 2.5 a%b= 0.0
a= 10 b= 1.5 a%b= 1.0
a= 7.7 b= 2.5 a%b= 0.20000000000000018
a= 12.4 b= 3 a%b= 0.40000000000000036
```

Python doesn't accept complex numbers to be used as operand in modulus operation. It throws TypeError: unsupported operand type(s) for %.

Exponent Operator

Python uses ** (double asterisk) as the exponent operator (sometimes called raised to operator). So, for a**b, you say a raised to b, or even bth power of a.

If in the exponentiation expression, both operands are integer, result is also an integer. In case either one is a float, the result is float. Similarly, if either one operand is complex number, exponent operator returns a complex number.

If the base is 0, the result is 0, and if the index is 0 then the result is always 1.

Example of exponent operator

</> Open Compiler

```
a=10
b=2
print ("a=",a, "b=",b, "a**b=", a**b)
a=10
b=1.5
print ("a=",a, "b=",b, "a**b=", a**b)
a=7.7
b=2
print ("a=",a, "b=",b, "a**b=", a**b)
a=1+2j
b=4
print ("a=",a, "b=",b, "a**b=", a**b)
a=12.4
```

```
b=0
print ("a=",a, "b=",b, "a**b=", a**b)
print ("a=",a, "b=",b, "b**a=", b**a)
```

It will produce the following **output** –

```
a= 10 b= 2 a**b= 100
a= 10 b= 1.5 a**b= 31.622776601683793
a= 7.7 b= 2 a**b= 59.290000000000006
a= (1+2j) b= 4 a**b= (-7-24j)
a= 12.4 b= 0 a**b= 1.0
a= 12.4 b= 0 b**a= 0.0
```

Floor Division Operator

Floor division is also called as integer division. Python uses // (double forward slash) symbol for the purpose. Unlike the modulus or modulo which returns the remainder, the floor division gives the quotient of the division of operands involved.

If both operands are positive, floor operator returns a number with fractional part removed from it. For example, the floor division of 9.8 by 2 returns 4 (pure division is 4.9, strip the fractional part, result is 4).

But if one of the operands is negative, the result is rounded away from zero (towards negative infinity). Floor division of -9.8 by 2 returns 5 (pure division is -4.9, rounded away from 0).

Example of floor division operator

</> Open Compiler

```
a=9
b=2
print ("a=",a, "b=",b, "a//b=", a//b)
a=9
b=-2
print ("a=",a, "b=",b, "a//b=", a//b)
a=10
b=1.5
print ("a=",a, "b=",b, "a//b=", a//b)
a=-10
```

```
b=1.5
print ("a=",a, "b=",b, "a//b=", a//b)
```

It will produce the following **output** –

```
a= 9 b= 2 a//b= 4
a= 9 b= -2 a//b= -5
a= 10 b= 1.5 a//b= 6.0
a= -10 b= 1.5 a//b= -7.0
```

Precedence and Associativity of Arithmetic Operators

Operator(s)	Description	Associativity
**	Exponent Operator	Associativity of Exponent operator is from **Right to Left**.
%, *, /, //	Modulus, Multiplication, Division, and Floor Division	Associativity of Modulus, Multiplication, Division, and Floor Division operators are from **Left to Right**.
+, –	Addition and Subtraction Operators	Associativity of Addition and Subtraction operators are from **Left to Right**.

The following table shows the precedence and associativity of the arithmetic operators in Python.

Arithmetic Operators with Complex Numbers

Arithmetic operators behave slightly differently when the both operands are complex number objects.

Addition and subtraction of complex numbers

Addition and subtraction of complex numbers is a simple addition/subtraction of respective real and imaginary components.

</> Open Compiler

```
a=2.5+3.4j
b=-3+1.0j
print ("Addition of complex numbers - a=",a, "b=",b, "a+b=", a+b)
print ("Subtraction of complex numbers - a=",a, "b=",b, "a-b=", a-b)
```

It will produce the following **output** −

```
Addition of complex numbers - a= (2.5+3.4j) b= (-3+1j) a+b= (-0.5+4.4j)
Subtraction of complex numbers - a= (2.5+3.4j) b= (-3+1j) a-b= (5.5+2.4j)
```

Multiplication of complex numbers

Multiplication of complex numbers is similar to multiplication of two binomials in algebra. If "a+bj" and "x+yj" are two complex numbers, then their multiplication is given by this formula −

```
(a+bj)*(x+yj) = ax+ayj+xbj+byj2 = (ax-by)+(ay+xb)j
```

For example,

```
a=6+4j
b=3+2j
c=a*b
c=(18-8)+(12+12)j
c=10+24j
```

The following program confirms the result −

```
a=6+4j
b=3+2j
print ("Multplication of complex numbers - a=",a, "b=",b, "a*b=", a*b)
```

To understand the how the division of two complex numbers takes place, we should use the conjugate of a complex number. Python's complex object has a conjugate() method that returns a complex number with the sign of imaginary part reversed.

```
>>> a=5+6j
>>> a.conjugate()
```

```
(5-6j)
```

Division of complex numbers

To divide two complex numbers, divide and multiply the numerator as well as the denominator with the conjugate of denominator.

```
a=6+4j
b=3+2j
c=a/b
c=(6+4j)/(3+2j)
c=(6+4j)*(3-2j)/3+2j)*(3-2j)
c=(18-12j+12j+8)/(9-6j+6j+4)
c=26/13
c=2+0j
```

To verify, run the following code –

</> Open Compiler

```
a=6+4j
b=3+2j
print ("Division of complex numbers - a=",a, "b=",b, "a/b=", a/b)
```

Complex class in Python doesn't support the modulus operator (%) and floor division operator (//).