Python - Operadores Aritméticos

Em Python, os números são o tipo de dados usado com mais frequência . Python usa os mesmos símbolos para operações aritméticas básicas com as quais todo mundo está familiarizado, ou seja, "+" para adição, "-" para subtração, "*" para multiplicação (a maioria das linguagens de programação usa "*" em vez de "x" como usado em matemática/álgebra) e "/" para divisão (novamente para o "÷" usado em matemática).

Além disso, Python define mais alguns operadores aritméticos. São eles "%" (Módulo), "**" (Expoente) e "//" (Divisão de piso).

Operadores Aritméticos Python

Os operadores aritméticos são operadores binários no sentido de que operam em dois operandos. Python suporta totalmente aritmética mista. Ou seja, os dois operandos podem ser de dois tipos de números diferentes. Em tal situação, o Python amplia o mais estreito dos operandos. Um objeto inteiro é mais estreito que um objeto float e float é mais estreito que um objeto complexo. Conseqüentemente, o resultado da operação aritmética de int e float é um float. O resultado de float e um complexo é um número complexo; da mesma forma, a operação em um número inteiro e um objeto complexo resulta em um objeto complexo.

Diferentes operadores aritméticos em Python

A seguir está a tabela que lista todos os operadores aritméticos disponíveis em Python:

Operador	Nome	Exemplo
+	Adição	uma + b = 30
-	Subtração	uma – b = -10
*	Multiplicação	uma * b = 200
/	Divisão	b / a = 2
%	Módulo	b % a = 0

**	Expoente	a**b =10**20
//	Divisão de Piso	9//2 = 4

Vamos estudar esses operadores com exemplos.

Operador de adição (+)

Este operador pronunciado como mais é um operador aritmético básico. Ele adiciona os dois operandos numéricos de cada lado e retorna o resultado da adição.

Exemplo: adição de dois inteiros

No exemplo a seguir, as duas variáveis inteiras são os operandos do operador "+".

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

Ele produzirá a seguinte **saída** -

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

Exemplo: adição de números inteiros e flutuantes

A adição de inteiro e float resulta em um float.

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

Ele produzirá a seguinte **saída** -

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

Exemplo: adição de dois números complexos

O resultado da adição de float a complex é um número complexo.

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

Ele produzirá a seguinte **saída** -

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

Operador de subtração (-)

Este operador, conhecido como menos, subtrai o segundo operando do primeiro. O número resultante é negativo se o segundo operando for maior.

Exemplo: Subtração de dois inteiros

O primeiro exemplo mostra a subtração de dois inteiros.

```
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)
```

Resultado -

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

Exemplo: subtração de números inteiros e flutuantes

A subtração de um inteiro e de um float segue o mesmo princípio.

```
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)
```

Ele produzirá a seguinte **saída** -

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

Exemplo: Subtração de dois números complexos

Na subtração envolvendo um complexo e um float, o componente real está envolvido na operação.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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)
```

Operador de multiplicação (*)

O símbolo * (asterisco) é definido como um operador de multiplicação em Python (como em muitas linguagens). Ele retorna o produto dos dois operandos de cada lado. Se algum dos operandos for negativo, o resultado também será negativo. Se ambos forem negativos, o resultado é positivo. Alterar a ordem dos operandos não altera o resultado

Exemplo: Multiplicação de dois inteiros

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

Ele produzirá a seguinte **saída** -

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

Exemplo: multiplicação de números inteiros e flutuantes

Na multiplicação, um operando flutuante pode ter uma notação de ponto decimal padrão ou uma notação científica.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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
```

Exemplo: Multiplicação com Número Complexo

Para a operação de multiplicação envolvendo um operando complexo, o outro operando multiplica tanto a parte real quanto a parte imaginária.

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

Ele produzirá a seguinte **saída** -

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

Operador de Divisão (/)

O símbolo "/" geralmente é chamado de barra. O resultado do operador de divisão é o numerador (operando esquerdo) dividido pelo denominador (operando direito). O número resultante é negativo se algum dos operandos for negativo. Como o infinito não pode ser armazenado na memória, Python gera ZeroDivisionError se o denominador for 0.

O resultado do operador de divisão em Python é sempre flutuante, mesmo que ambos os operandos sejam inteiros.

Exemplo: Divisão de Dois Inteiros

```
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)
```

Ele produzirá a seguinte **saída** -

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

Exemplo: Divisão com Números Flutuantes

Na divisão, um operando flutuante pode ter uma notação de ponto decimal padrão ou uma notação científica.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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
```

Exemplo: Divisão com Número Complexo

Quando um dos operandos é um número complexo, ocorre a divisão entre o outro operando e ambas as partes do objeto do número complexo (real e imaginário).

```
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)
```

Ele produzirá a seguinte **saída** -

```
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)
```

Se o numerador for 0, o resultado da divisão será sempre 0, exceto quando o denominador for 0; nesse caso, o Python gera ZeroDivisionError com a mensagem de erro Divisão por Zero.

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

Ele produzirá a seguinte **saída** -

```
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
```

Operador Módulo (%)

Python define o símbolo "%", que é conhecido como símbolo de porcentagem, como operador Módulo (ou módulo). Ele retorna o resto depois que o denominador divide o numerador. Também pode ser chamado de operador Restante. O resultado do operador módulo é o número que permanece após o quociente inteiro. Para dar um exemplo, quando 10 é dividido por 3, o quociente é 3 e o resto é 1. Portanto, 10%3 (normalmente pronunciado como 10 mod 3) resulta em 1.

Exemplo: Operação de Módulo com Valores Inteiros

Se ambos os operandos forem inteiros, o valor do módulo será um número inteiro. Se o numerador for completamente divisível, o resto será 0. Se o numerador for menor que o denominador, o módulo será igual ao numerador. Se o denominador for 0, Python gera ZeroDivisionError.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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
```

Exemplo: Operação de Módulo com Valores Flutuantes

Se algum dos operandos for float, o valor do mod será sempre float.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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 não aceita números complexos para serem usados como operando na operação de módulo. Ele lança TypeError: tipos de operandos não suportados para%.

Operador Expoente (**)

Python usa ** (asterisco duplo) como operador expoente (às vezes chamado de operador elevado a). Então, para a**b, você diz a elevado a b, ou mesmo b-ésima potência de a.

Se na expressão de exponenciação ambos os operandos forem inteiros, o resultado também será um número inteiro. Caso um deles seja float, o resultado será float. Da mesma forma, se um dos operandos for um número complexo, o operador expoente retornará um número complexo.

Se a base for 0, o resultado será 0, e se o índice for 0, o resultado será sempre 1.

Exemplo de operador expoente

```
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)
```

Ele produzirá a seguinte **saída** -

```
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
```

Operador de Divisão de Andar (//)

A divisão de piso também é chamada de divisão inteira. Python usa o símbolo // (barra dupla) para esse propósito. Ao contrário do módulo ou módulo que retorna o resto, a divisão de piso fornece o quociente da divisão dos operandos envolvidos.

Se ambos os operandos forem positivos, o operador floor retorna um número com a parte fracionária removida dele. Por exemplo, a divisão mínima de 9,8 por 2 retorna 4 (a divisão pura é 4,9, retire a parte fracionária, o resultado é 4).

Mas se um dos operandos for negativo, o resultado será arredondado de zero (em direção ao infinito negativo). A divisão mínima de -9,8 por 2 retorna 5 (a divisão pura é -4,9, arredondada de 0).

Exemplo de operador de divisão de piso

```
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)
```

Ele produzirá a seguinte **saída** -

```
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
```

Precedência e associatividade de operadores aritméticos Python

Operador(es)	Descrição	Associatividade
**	Operador Expoente	A associatividade do operador Expoente é da **direita para a esquerda** .
%, *, /, //	Módulo, multiplicação, divisão e divisão de piso	A associatividade dos operadores Módulo, Multiplicação, Divisão e Divisão de piso é da **esquerda para a direita** .
+, –	Operadores de adição e subtração	A associatividade dos operadores de adição e

		subtração é da **esquerda para a direita** .

A tabela a seguir mostra a precedência e associatividade dos operadores aritméticos em Python.

Aritmética de números complexos

Os operadores aritméticos se comportam de maneira ligeiramente diferente quando ambos os operandos são objetos de números complexos.

A adição e subtração de números complexos é uma simples adição/subtração dos respectivos componentes reais e imaginários.

```
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)
```

Ele produzirá a seguinte **saída** -

```
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)
```

A multiplicação de números complexos é semelhante à multiplicação de dois binômios em álgebra. Se "a+bj" e "x+yj" são dois números complexos, então sua multiplicação é dada por esta fórmula -

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

Por exemplo,

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

O programa a seguir confirma o resultado -

```
a=6+4j
b=3+2j
```

```
print ("Multplication of complex numbers - a=",a, "b=",b, "a*b=", a*b)
```

Para entender como ocorre a divisão de dois números complexos, devemos usar o conjugado de um número complexo. O objeto complexo do Python possui um método conjugate() que retorna um número complexo com o sinal da parte imaginária invertido.

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

Para dividir dois números complexos, divida e multiplique o numerador e também o denominador pelo conjugado do denominador.

```
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
```

Para verificar, execute o seguinte código -

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

Classe complexa em Python não suporta o operador de módulo (%) e o operador de divisão de piso (//).