Python - Operadores de Comparação

Operadores de comparação Python

Os operadores de comparação em Python são muito importantes nas instruções condicionais do Python (**if, else** e **elif**) e nas instruções de loop (while e for loops). Os **operadores de comparação** também são chamados **de operadores relacionais** . Alguns dos operadores mais conhecidos são "<" significa menor que e ">" significa maior que.

Python usa mais dois operadores, combinando o símbolo "=" com esses dois. O símbolo "<=" é para menor ou igual ao operador e o símbolo ">=" é para maior ou igual ao operador.

Diferentes operadores de comparação em Python

Python tem mais dois operadores de comparação na forma de "==" e "!=". Eles são **iguais** e **não iguais aos** operadores. Portanto, existem seis operadores de comparação em Python e eles estão listados abaixo nesta tabela:

<	Menor que	uma<b
>	Maior que	a> b
<=	Menos que ou igual a	uma<=b
>=	Melhor que ou igual a	uma>=b
==	É igual a	uma==b
!=	Não é igual a	uma!=b

Os operadores de comparação são de natureza binária, exigindo dois operandos. Uma expressão que envolve um operador de comparação é chamada de expressão booleana e sempre retorna Verdadeiro ou Falso.

Exemplo

```
a=5
b=7
```

```
print (a>b)
print (a<b)
```

Ele produzirá a seguinte **saída** -

```
False
True
```

Ambos os operandos podem ser literais , variáveis ou expressões do Python. Como o Python suporta aritmética mista, você pode ter qualquer tipo de operando numérico.

Exemplo

O código a seguir demonstra o uso dos **operadores de comparação do Python com números inteiros** -

```
print ("Both operands are integer")
a=5
b=7
print ("a=",a, "b=",b, "a>b is", a>b)
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
Both operands are integer
a= 5 b= 7 a>b is False
a= 5 b= 7 a<b is True
a= 5 b= 7 a==b is False
a= 5 b= 7 a!=b is True
```

Comparação do número flutuante

No exemplo a seguir, um operando inteiro e um operando flutuante são comparados.

Exemplo

```
print ("comparison of int and float")
a=10
```

```
b=10.0
print ("a=",a, "b=",b, "a>b is", a>b)
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
comparison of int and float
a= 10 b= 10.0 a>b is False
a= 10 b= 10.0 a<b is False
a= 10 b= 10.0 a==b is True
a= 10 b= 10.0 a!=b is False
```

Comparação de números complexos

Embora objeto complexo seja um tipo de dados numérico em Python , seu comportamento é diferente dos outros. Python não suporta operadores < e >, mas suporta operadores de igualdade (==) e desigualdade (!=).

Exemplo

```
print ("comparison of complex numbers")
a=10+1j
b=10.-1j
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
comparison of complex numbers
a= (10+1j) b= (10-1j) a==b is False
a= (10+1j) b= (10-1j) a!=b is True
```

Você obtém um TypeError com operadores menores ou maiores que.

Exemplo

```
print ("comparison of complex numbers")
a=10+1j
```

```
b=10.-1j
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a>b is",a>b)
```

Ele produzirá a seguinte **saída** -

```
comparison of complex numbers
Traceback (most recent call last):
  File "C:\Users\mlath\examples\example.py", line 5, in <module>
    print ("a=",a, "b=",b,"a<b is",a<b)
                              ^^^
TypeError: '<' not supported between instances of 'complex' and
'complex
```


Comparação de booleanos

Objetos booleanos em Python são na verdade números inteiros: True é 1 e False é 0. Na verdade, Python trata qualquer número diferente de zero como True. Em Python, a comparação de objetos booleanos é possível. "Falso < Verdadeiro" é Verdadeiro!

Exemplo

```
print ("comparison of Booleans")
a=True
b=False
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a>b is",a>b)
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
comparison of Booleans
a= True b= False a<b is False
a= True b= False a>b is True
```

```
a= True b= False a==b is False
a= True b= False a!=b is True
```


Comparação de tipos de sequência

Em Python, a comparação apenas de objetos de sequência semelhantes pode ser realizada. Um objeto string é comparável apenas a outra string. Uma lista não pode ser comparada com uma tupla , mesmo que ambas tenham os mesmos itens.

Exemplo

```
print ("comparison of different sequence types")
a=(1,2,3)
b=[1,2,3]
print ("a=",a, "b=",b,"a<b is",a<b)
```

Ele produzirá a seguinte **saída** -

```
comparison of different sequence types
Traceback (most recent call last):
  File "C:\Users\mlath\examples\example.py", line 5, in <module>
    print ("a=",a, "b=",b,"a<b is",a<b)
                                ^^^
TypeError: '<' not supported between instances of 'tuple' and 'list'
```

Objetos de sequência são comparados por mecanismo de ordenação lexicográfica. A comparação começa no item no índice 0. Se forem iguais, a comparação passa para o próximo índice até que os itens em determinado índice não sejam iguais ou uma das sequências se esgote. Se uma sequência for uma subsequência inicial da outra, a sequência mais curta será a menor (menor).

Qual dos operandos é maior depende da diferença de valores dos itens do índice onde são desiguais. Por exemplo, 'BAT'>'BAR' é True, pois T vem depois de R na ordem Unicode.

Se todos os itens de duas sequências forem iguais, as sequências serão consideradas iguais.

Exemplo

```
print ("comparison of strings")
a='BAT'
b='BALL'
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a>b is",a>b)
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
comparison of strings
a= BAT b= BALL a<b is False
a= BAT b= BALL a>b is True
a= BAT b= BALL a==b is False
a= BAT b= BALL a!=b is True
```

No exemplo a seguir, dois objetos tupla são comparados -

Exemplo

```
print ("comparison of tuples")
a=(1,2,4)
b=(1,2,3)
print ("a=",a, "b=",b,"a<b is",a<b)
print ("a=",a, "b=",b,"a>b is",a>b)
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
a= (1, 2, 4) b= (1, 2, 3) a<b is False
a= (1, 2, 4) b= (1, 2, 3) a>b is True
a= (1, 2, 4) b= (1, 2, 3) a==b is False
a= (1, 2, 4) b= (1, 2, 3) a!=b is True
```

Comparação de objetos de dicionário

O uso dos operadores "<" e ">" para o dicionário Python não está definido. No caso desses operandos, TypeError: '<' notsupported entre instâncias de 'dict' e 'dict' é relatado.

A comparação de igualdade verifica se o comprimento de ambos os itens do dict é o mesmo. O comprimento do dicionário é o número de pares de valores-chave nele.

Os dicionários Python são simplesmente comparados por comprimento. O dicionário com menos elementos é considerado menos que um dicionário com mais elementos.

Exemplo

```
print ("comparison of dictionary objects")
a={1:1,2:2}
b={2:2, 1:1, 3:3}
print ("a=",a, "b=",b,"a==b is",a==b)
print ("a=",a, "b=",b,"a!=b is",a!=b)
```

Ele produzirá a seguinte **saída** -

```
comparison of dictionary objects
a= {1: 1, 2: 2} b= {2: 2, 1: 1, 3: 3} a==b is False
a= {1: 1, 2: 2} b= {2: 2, 1: 1, 3: 3} a!=b is True
```