Python - Operadores de associação

Operadores de associação Python

Os operadores de associação em Python nos ajudam a determinar se um item está presente em um determinado objeto do tipo contêiner ou, em outras palavras, se um item é membro de um determinado objeto do tipo contêiner.

Tipos de operadores de associação Python

Python tem dois operadores de associação: **in** e **not in** . Ambos retornam um resultado booleano. O resultado do operador **in** é oposto ao do operador **not in** .

O operador 'in'

O operador " **in** " é usado para verificar se uma substring está presente em uma string maior , se algum item está presente em uma lista ou tupla, ou se uma sublista ou subtupla está incluída em uma lista ou tupla.

Exemplo de operador "in" de associação Python

No exemplo a seguir, diferentes substrings são verificados se pertencem à string var="TutorialsPoint". Python diferencia caracteres com base em seu valor Unicode. Portanto, “To” não é o mesmo que “to”. Observe também que se o operador “in” retornar True, o operador “not in” será avaliado como False.

```
var = "TutorialsPoint"
a = "P"
b = "tor"
c = "in"
d = "To"
print (a, "in", var, ":", a in var)
print (b, "in", var, ":", b in var)
print (c, "in", var, ":", c in var)
print (d, "in", var, ":", d in var)
```

Ele produzirá a seguinte **saída** -

```
P in TutorialsPoint : True
tor in TutorialsPoint : True
in in TutorialsPoint : True
To in TutorialsPoint : False
```

O operador 'não dentro'

O operador "not in" é usado para verificar se uma sequência com o valor fornecido não está presente no objeto como string, list , tuple , etc.

Exemplo de operador Python Membership "not in"

```
var = "TutorialsPoint"
a = "P"
b = "tor"
c = "in"
d = "To"
print (a, "not in", var, ":", a not in var)
print (b, "not in", var, ":", b not in var)
print (c, "not in", var, ":", c not in var)
print (d, "not in", var, ":", d not in var)
```

Ele produzirá a seguinte **saída** -

```
P not in TutorialsPoint : False
tor not in TutorialsPoint : False
in not in TutorialsPoint : False
To not in TutorialsPoint : True
```

Operador de associação com listas e tuplas

Você pode usar o operador "in/not in" para verificar a associação de um item na lista ou tupla.

```
var = [10,20,30,40]
a = 20
b = 10
c = a-b
d = a/2
print (a, "in", var, ":", a in var)
```

```
print (b, "not in", var, ":", b not in var)
print (c, "in", var, ":", c in var)
print (d, "not in", var, ":", d not in var)
```

Ele produzirá a seguinte **saída** -

```
20 in [10, 20, 30, 40] : True
10 not in [10, 20, 30, 40] : False
10 in [10, 20, 30, 40] : True
10.0 not in [10, 20, 30, 40] : False
```

No último caso, "d" é um float, mas ainda assim se compara a True com 10 (um **int**) na lista. Mesmo que um número expresso em outros formatos como binário, octal ou hexadecimal seja fornecido, os operadores de associação informam se ele está dentro da sequência.

```
>>>  0x14 in [10, 20, 30, 40]
True
```

Exemplo

No entanto, se você tentar verificar se dois números sucessivos estão presentes em uma lista ou tupla, o operador in retornará False. Se a lista/tupla contiver os números sucessivos como uma sequência, ela retornará True.

```
var = (10,20,30,40)
a = 10
b = 20
print ((a,b), "in", var, ":", (a,b) in var)
var = ((10,20),30,40)
a = 10
b = 20
print ((a,b), "in", var, ":", (a,b) in var)
```

Ele produzirá a seguinte **saída** -

```
(10, 20) in (10, 20, 30, 40) : False
(10, 20) in ((10, 20), 30, 40) : True
```

Operador de associação com conjuntos

Os operadores de associação do Python também funcionam bem com os objetos definidos .

```
var = {10,20,30,40}
a = 10
b = 20
print (b, "in", var, ":", b in var)
var = {(10,20),30,40}
a = 10
b = 20
print ((a,b), "in", var, ":", (a,b) in var)
```

Ele produzirá a seguinte **saída** -

```
20 in {40, 10, 20, 30} : True
(10, 20) in {40, 30, (10, 20)} : True
```


Operador de associação com dicionários

O uso de operadores in e not in com objeto de dicionário é permitido. No entanto, o Python verifica a associação apenas com a coleção de chaves e não de valores.

```
var = {1:10, 2:20, 3:30}
a = 2
b = 20
print (a, "in", var, ":", a in var)
print (b, "in", var, ":", b in var)
```

Ele produzirá a seguinte **saída** -

```
2 in {1: 10, 2: 20, 3: 30} : True
20 in {1: 10, 2: 20, 3: 30} : False
```