Python - Membership Operators

Python Membership Operators

The membership operators in Python help us determine whether an item is present in a given container type object, or in other words, whether an item is a member of the given container type object.

Types of Python Membership Operators

Python has two membership operators: **in** and **not in**. Both return a Boolean result. The result of **in** operator is opposite to that of **not in** operator.

The 'in' Operator

The "**in**" operator is used to check whether a substring is present in a bigger string, any item is present in a list or tuple, or a sub-list or sub-tuple is included in a list or tuple.

Example of Python Membership "in" Operator

In the following example, different substrings are checked whether they belong to the string var="TutorialsPoint". Python differentiates characters on the basis of their Unicode value. Hence "To" is not the same as "to". Also note that if the "in" operator returns True, the "not in" operator evaluates to False.

</> Open Compiler

```
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)
```

It will produce the following **output** –

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

The 'not in' Operator

The "not in" operator is used to check a sequence with the given value is not present in the object like string, list, tuple, etc.

Example of Python Membership "not in" Operator

</> Open Compiler

```
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)
```

It will produce the following **output** –

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

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

Membership Operator with Lists and Tuples

You can use the "in/not in" operator to check the membership of an item in the list or tuple.

</> Open Compiler

```
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)
```

It will produce the following **output** –

```
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
```

In the last case, "d" is a float but still it compares to True with 10 (an **int**) in the list. Even if a number expressed in other formats like binary, octal or hexadecimal are given the membership operators tell if it is inside the sequence.

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

Example

However, if you try to check if two successive numbers are present in a list or tuple, the in operator returns False. If the list/tuple contains the successive numbers as a sequence itself, then it returns True.

</> Open Compiler

```
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)
```

It will produce the following **output** –

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

Membership Operator with Sets

Python's membership operators also work well with the set objects.

</> Open Compiler

```
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)
```

It will produce the following **output** –

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

Membership Operator with Dictionaries

Use of in as well as not in operators with dictionary object is allowed. However, Python checks the membership only with the collection of keys and not values.

</> Open Compiler

```
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)
```

It will produce the following **output** –

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