Python - Identity Operators

Python Identity Operators

The identity operators compare the objects to determine whether they share the same memory and refer to the same object type (data type).

Python provided two identity operators; we have listed them as follows:

- '**is**' Operator
- '**is not**' Operator

Python 'is' Operator

The '**is**' operator evaluates to True if both the operand objects share the same memory location. The memory location of the object can be obtained by the "id()" function. If the "id()" of both variables is same, the "is" operator returns True.

Example of Python Identity 'is' Operator

Open Compiler

```
a = [1, 2, 3, 4, 5]
b = [1, 2, 3, 4, 5]
c = a

# Comparing and printing return values
print(a is c)
print(a is b)

# Printing IDs of a, b, and c
print("id(a) : ", id(a))
print("id(b) : ", id(b))
print("id(c) : ", id(c))
```

It will produce the following **output** −

```
True
False
```

```
id(a) :  140114091859456
id(b) :  140114091906944
id(c) :  140114091859456
```

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

Python 'is not' Operator

The '**is not**' operator evaluates to True if both the operand objects do not share the same memory location or both operands are not the same objects.

Example of Python Identity 'is not' Operator

Open Compiler

```python
a = [1, 2, 3, 4, 5]
b = [1, 2, 3, 4, 5]
c = a

# Comparing and printing return values
print(a is not c)
print(a is not b)

# Printing IDs of a, b, and c
print("id(a) : ", id(a))
print("id(b) : ", id(b))
print("id(c) : ", id(c))
```

It will produce the following **output** −

```
False
True
id(a) :  140559927442176
id(b) :  140559925598080
id(c) :  140559927442176
```

Python Identity Operators Examples with Explanations

Example 1

Open Compiler

```
a="TutorialsPoint"
b=a
print ("id(a), id(b):", id(a), id(b))
print ("a is b:", a is b)
print ("b is not a:", b is not a)
```

It will produce the following **output** −

```
id(a), id(b): 2739311598832 2739311598832
a is b: True
b is not a: False
```

The list and tuple objects behave differently, which might look strange in the first instance. In the following example, two lists "a" and "b" contain same items. But their id() differs.

Example 2

Open Compiler

```
a=[1,2,3]
b=[1,2,3]
print ("id(a), id(b):", id(a), id(b))
print ("a is b:", a is b)
print ("b is not a:", b is not a)
```

It will produce the following **output** −

```
id(a), id(b): 1552612704640 1552567805568
a is b: False
b is not a: True
```

The list or tuple contains the memory locations of individual items only and not the items itself. Hence "a" contains the addresses of 10,20 and 30 integer objects in a certain location which may be different from that of "b".

Example 3

```
print (id(a[0]), id(a[1]), id(a[2]))
print (id(b[0]), id(b[1]), id(b[2]))
```

It will produce the following **output** –

```
140734682034984 140734682035016 140734682035048
140734682034984 140734682035016 140734682035048
```

Because of two different locations of "a" and "b", the "**is**" operator returns False even if the two lists contain same numbers.