Identity Operator

Identity operator is a binary operator in python
This operator is used to compare identity of two values or objects.
“is” keyword represents identity operator in python

What is identity?
Identity is an integer value which is used to identify memory location.
Memory address (OR) value address (OR) object address is called identity.

How find identity of value or object?
id() this function returns identity or address of object

id(value/variable)

```
>>> a=10
>>> id(a)
140703247369416
>>> b=20
>>> id(b)
140703247369736
```

Identity operator is used to find two variables pointing same memory location or same value in memory

Example:

```
>>> a=10
>>> id(a)
140703247369416
>>> b=20
>>> id(b)
140703247369736
>>> c=a
>>> id(c)
140703247369416
>>> a is b
False
>>> a is c
True
```

What difference is between == and is operator in python?

In Python, == checks for value equality (whether two objects have the same content), while is checks for identity (whether two variables point to the same object in memory).

Python Data Types

Python data types are classified into categories

1. Mutable types
2. Immutable Types

Immutable Data Type

Immutable objects values cannot change in same memory location, the changes are done in new memory location.
Immutable data types are represents immutable objects

The following data types are called immutable data types

1. All scalar data types are immutable
    a. Int
    b. Float
    c. Complex
    d. Bool
    e. NoneType
2. Collection Types
    a. Sequences
        i. Tuple
        ii. String
        iii. Range
        iv. Bytes
    b. Sets
        i. Frozenset

Mutable objects values can be changed in same memory location.
The following data types are mutable data types and used for creating mutable objects

Collections

1. Sequences
    a. List

    b. Bytearray
2. Sets
    a. Set
3. Mapping
    a. Dict

Immutable are sharable, it mean one memory location value or one object shared by number of variables, when it hold same value.

Example:
#Proof of Concept (POC)

```
a=10
b=10
c=10
d=5+5
print(a,b,c,d)
print(id(a),id(b),id(c),id(d))
f1=1.5
f2=1.5
f3=1.5
print(f1,f2,f3)
print(id(f1),id(f2),id(f3))
x=100
print(id(x))
x=200
print(id(x))
```

Output

```
10 10 10 10
140703247369416 140703247369416 140703247369416
140703247369416
1.5 1.5 1.5
2616903006480 2616903006480 2616903006480
140703247372296
140703247375496
```

Example:

```
# Mutable of are not sharable
# POC (Proof Of Concept)
a=[10,20,30]
```

```
b=[10,20,30]
print(a,b)
print(id(a),id(b))
a[0]=99
print(a)
print(id(a))
```

Output

```
[10, 20, 30] [10, 20, 30]
2825965882880 2825922788416
[99, 20, 30]
2825965882880
```

Example:

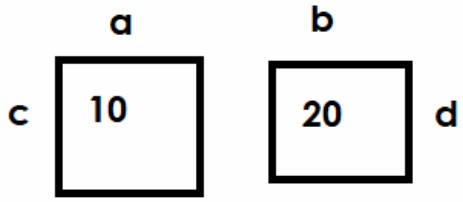
```
a=[10,20,30]
b=[10,20,30]
print(a==b)
print(a is b)
x=1.5
y=1.5
print(x==y)
print(x is y)
c=a
print(a is c)
```

Output

```
True
False
True
True
True
```

How many variables and objects are created in the following code?

```
a=10
b=20
c=b-a
d=a+c
```



4 Variables ==> a,b,c,d
Objects ==> 2 objects

How may variables and objects created in the following code?

```
a=[10,20,30]
b=[40,50,60]
c=[10,20,30]
```

List is mutable and mutables are not sharable

Variables ==> 3 variables a,b,c
Objects ==> 3 objects

Bitwise Operators

Bitwise Operators are used to perform operations on bits or binary data.

Python support the following bitwise operators

Operators	**Description**
>>	Right shift operator
<<	Left shift operator
&	Bitwise and operator
|	Bitwise or operator
^	Bitwise xor operator
~	Bitwise not operator