Identity Operators (Applicable in Python only)

Additional examples of all the data types in Python

In [10]:
```
a=None # None is keyword
b=None
print(a,type(a),id(a))
```

```
None <class 'NoneType'> 140728651757520
```

In [12]:
```
a=None
b=None
print(a,type(a),id(a))
```

```
None <class 'NoneType'> 140728651757520
```

In [14]:
```
a is b
```

Out[14]:
```
True
```

In [16]:
```
a is not b
```

Out[16]:
```
False
```

In [36]:
```
a={10:"Apples", 20:"Mangos", 30:"Kiwi"}
b={10:"Apples", 20:"Mangos", 30:"Kiwi"}
print(a,type(a),id(a))
```

```
{10: 'Apples', 20: 'Mangos', 30: 'Kiwi'} <class 'dict'> 2961079483648
```

In [38]:
```
print(b,type(b),id(b))
```

```
{10: 'Apples', 20: 'Mangos', 30: 'Kiwi'} <class 'dict'> 2961079752384
```

In [40]:
```
a is b
```

Out[40]:
```
False
```

In [42]:
```
a is not b
```

Out[42]:
```
True
```

In [44]:
```
#forzenset:
a={10,20,30}
b={10,20,30}
print(a,type(a),id(a))
print(b,type(b),id(b))
```

```
{10, 20, 30} <class 'set'> 2960931694720
{10, 20, 30} <class 'set'> 2961065472544
```

In [46]:
```
a is b
```
Out[46]:
```
False
```

In [48]:
```
a is not b
```
Out[48]:
```
True
```

In [50]:
```
a=frozenset({10,20,30})
b=frozenset({10,20,30})
print(a,type(a),id(a))
print(b,type(b),id(b))
```
```
frozenset({10, 20, 30}) <class 'frozenset'> 2961065468064
frozenset({10, 20, 30}) <class 'frozenset'> 2961065470080
```

In [52]:
```
a is b
```
Out[52]:
```
False
```

In [54]:
```
a is not b
```
Out[54]:
```
True
```

In [58]:
```
#list
a=[10,20,30]
b=[10,20,30]
print(a,type(a),id(a))
print(b,type(b),id(b))
```
```
[10, 20, 30] <class 'list'> 2961079712960
[10, 20, 30] <class 'list'> 2961079705472
```

In [60]:
```
a is b
```
Out[60]:
```
False
```

In [62]:
```
a is not b
```
Out[62]:
```
True
```

In [64]:
```
#tuple
a=(10,20,30)
b=(10,20,30)
print(a,type(a),id(a))
print(b,type(b),id(b))
```
```
(10, 20, 30) <class 'tuple'> 2961065365632
(10, 20, 30) <class 'tuple'> 2961079700096
```

In [66]:
```
#range
a=range(10,20,2)
b=range(10,20,2)
```

```
print(a,type(a),id(a))
print(b,type(b),id(b))
```

```
range(10, 20, 2) <class 'range'> 2961047586736
range(10, 20, 2) <class 'range'> 2961039765936
```

In [68]:
```
a is b
```

Out[68]:
```
False
```

In [70]:
```
a is not b
```

Out[70]:
```
True
```

In [2]:
```
#bytes
a=bytes([10,20,30,40])
b=bytes([10,20,30,40])
print(a,type(a),id(a))
print(b,type(b),id(b))
```

```
b'\n\x14\x1e(' <class 'bytes'> 1277794729824
b'\n\x14\x1e(' <class 'bytes'> 1277782737856
```

In [4]:
```
a is b
```

Out[4]:
```
False
```

In [6]:
```
a is not b
```

Out[6]:
```
True
```

In [14]:
```
#bytesarry
ba1=bytearray([10,20,30,40])
ba2=bytearray([10,20,30,40])
print(ba1,type(ba1),id(ba1))
print(ba2,type(ba2),id(ba2))
```

```
bytearray(b'\n\x14\x1e(') <class 'bytearray'> 1277826735728
bytearray(b'\n\x14\x1e(') <class 'bytearray'> 1277826735856
```

In [16]:
```
a is b
```

Out[16]:
```
False
```

In [18]:
```
a is not b
```

Out[18]:
```
True
```

In [20]:
```
#string data type
a="MRIIRS"
b="MRIIRS"
print(a,type(a),id(a))
print(b,type(b),id(b)) #Same memory space
```

```
MRIIRS <class 'str'> 1277773072976
MRIIRS <class 'str'> 1277773072976
```

In [22]:
```
a="CDOE"
b="CDEO"
print(a,type(a),id(a))
print(b,type(b),id(b)) #different memory space
```
```
CDOE <class 'str'> 1277795519296
CDEO <class 'str'> 1277795524576
```

In [24]:
```
a="123"
b="123"
print(a,type(a),id(a))
print(b,type(b),id(b)) #Same memory space
```
```
123 <class 'str'> 1279895254960
123 <class 'str'> 1279895254960
```

In [26]:
```
#complex
a=2+3j
b=2+3j
print(a,type(a),id(a))
print(b,type(b),id(b)) #different memory space
```
```
(2+3j) <class 'complex'> 1277826562224
(2+3j) <class 'complex'> 1277826559472
```

In [28]:
```
a is b
```
Out[28]:
```
False
```

In [30]:
```
a is not b
```
Out[30]:
```
True
```

In [38]:
```
a=True #keyword
b=True
print(a,type(a),id(a))
print(b,type(b),id(b))
```
```
True <class 'bool'> 140727794072448
True <class 'bool'> 140727794072448
```

In [34]:
```
a is b
```
Out[34]:
```
True
```

In [36]:
```
a is not b
```
Out[36]:
```
False
```

In [40]:
```
#float data type:
a=1.2
b=1.2
print(a,type(a),id(a))
print(b,type(b),id(b))
```

```
1.2 <class 'float'> 1277794082352
1.2 <class 'float'> 1277794086000
```

In [42]:
```python
#int data type:
#If the int value from 0 to 256 present objects with same value then whose address
a=10
b=10
print(a,type(a),id(a))
print(b,type(b),id(b)) #Same memory address
```

```
10 <class 'int'> 140727795198680
10 <class 'int'> 140727795198680
```

In [44]:
```python
# int:
# If the int value from 0 to 256 present objects with same value then whose address
a=300
b=300
print(a,type(a),id(a))
print(b,type(b),id(b)) #different memory space
```

```
300 <class 'int'> 1277826559056
300 <class 'int'> 1277826560048
```

# int data type: If the int value from 0 to 256 present objects with same value then whose address is same, otherwise different.

In [48]:
```python
a=255
b=255
print(a,type(a),id(a))
print(b,type(b),id(b)) #Same memory address
```

```
255 <class 'int'> 140727795206520
255 <class 'int'> 140727795206520
```

In [52]:
```python
a is b
```

Out[52]:
```
False
```

In [54]:
```python
a is not b
```

Out[54]:
```
True
```

In [50]:
```python
a=257
b=257
print(a,type(a),id(a))
print(b,type(b),id(b))
```

```
257 <class 'int'> 1277826561968
257 <class 'int'> 1277826560880
```

In [56]:
```python
a is b
```

```
Out[56]:  False
```

```
In [58]:  a is not b
```

```
Out[58]:  True
```

```
In [60]:  a=-5
          b=-5
          print(a,type(a),id(a))
          print(b,type(b),id(b))
```

```
-5 <class 'int'> 140727795198200
-5 <class 'int'> 140727795198200
```

```
In [62]:  a is b
```

```
Out[62]:  True
```

```
In [64]:  a is not b
```

```
Out[64]:  False
```

```
In [66]:  a=-2
          b=-2
          print(a,type(a),id(a))
          print(b,type(b),id(b))
```

```
-2 <class 'int'> 140727795198296
-2 <class 'int'> 140727795198296
```

```
In [74]:  #Both a and b were assigned the value 4000 in the same line.
          #In this case, Python optimizes and points both a and b to the same object in memor
```

When we assign any range of integer,float,complex,values using multiline assignment with the same value in different objects, those objects share the same address.”

```
In [68]:  a,b=4000,4000
          print(a,type(a),id(a))
          print(b,type(b),id(b))
```

```
4000 <class 'int'> 1277826560816
4000 <class 'int'> 1277826560816
```

```
In [70]:  a is b
```

```
Out[70]:  True
```

```
In [72]: a is not b
Out[72]: False
```

“When we create sequence, list, set, or dictionary types using single-line or multi-line assignment with the same values in different objects, they will have different addresses.

```
In [13]: a=[10,20,30]
         b=[10,20,30]
         print(a,type(a),id(a))
         print(b,type(b),id(b))

[10, 20, 30] <class 'list'> 1697935566912
[10, 20, 30] <class 'list'> 1697935561152
```

```
In [15]: a is b
Out[15]: False
```

```
In [17]: a is not b
Out[17]: True
```

```
In [19]: a,b=[10,20,30],[10,20,30]
         print(a,type(a),id(a))
         print(b,type(b),id(b))

[10, 20, 30] <class 'list'> 1697935561408
[10, 20, 30] <class 'list'> 1697935491200
```

```
In [21]: a is b
Out[21]: False
```

```
In [23]: a is not b
Out[23]: True
```

```
In [27]: a,b={10:"CDOE"},{10:"CDOE"}
         print(a,type(a),id(a))
         print(b,type(b),id(b))

{10: 'CDOE'} <class 'dict'> 1697935563456
{10: 'CDOE'} <class 'dict'> 1697975175488
```

```
In [29]: a is b
Out[29]: False
```

```
In [31]: a is not b
Out[31]: True
```

short hand operator

In [ ]:
```
#a = a + b  # Normal Expression====  var1 = var1 op var2
We can write the above Expression by using Short Hand Operator
var1 = var1 op var2 ========> var1 op= var2

a = 10
b = 20
a = a + b  #Normal Expression -------- Short Notation a += b here += is called Shor
```

In [37]:
```
a=10
b=20
a+=b
print(a)
```
```
30
```

In [39]:
```
a=10
b=20
a-=b
print(a)
```
```
-10
```

In [41]:
```
a=10
b=20
a*=b
print(a)
```
```
200
```

In [43]:
```
a=2
b=3
c=4
k=a+b*c
print(k)
```
```
14
```

In [47]:
```
a=2
b=3
c=4
a+=b*c
print(a)
```
```
14
```

In [49]:
```
a=10
b=2
a=a/b
print(a)
```
```
5.0
```

```
In [51]: a=10
         b=2
         a/=b
         print(a)

         5.0

In [53]: a=10
         b=2
         a/=b
         print(a)

         5.0

In [55]: a=10
         b=2
         a=b/a
         print(a)

         0.2

In [57]: a=10
         b=2
         a>>=b
         print(a)

         2

In [61]: a=10
         b=3
         a=a|b
         print(a)

         11

In [63]: a=10
         b=3
         a|=b
         print(a)

         11
```