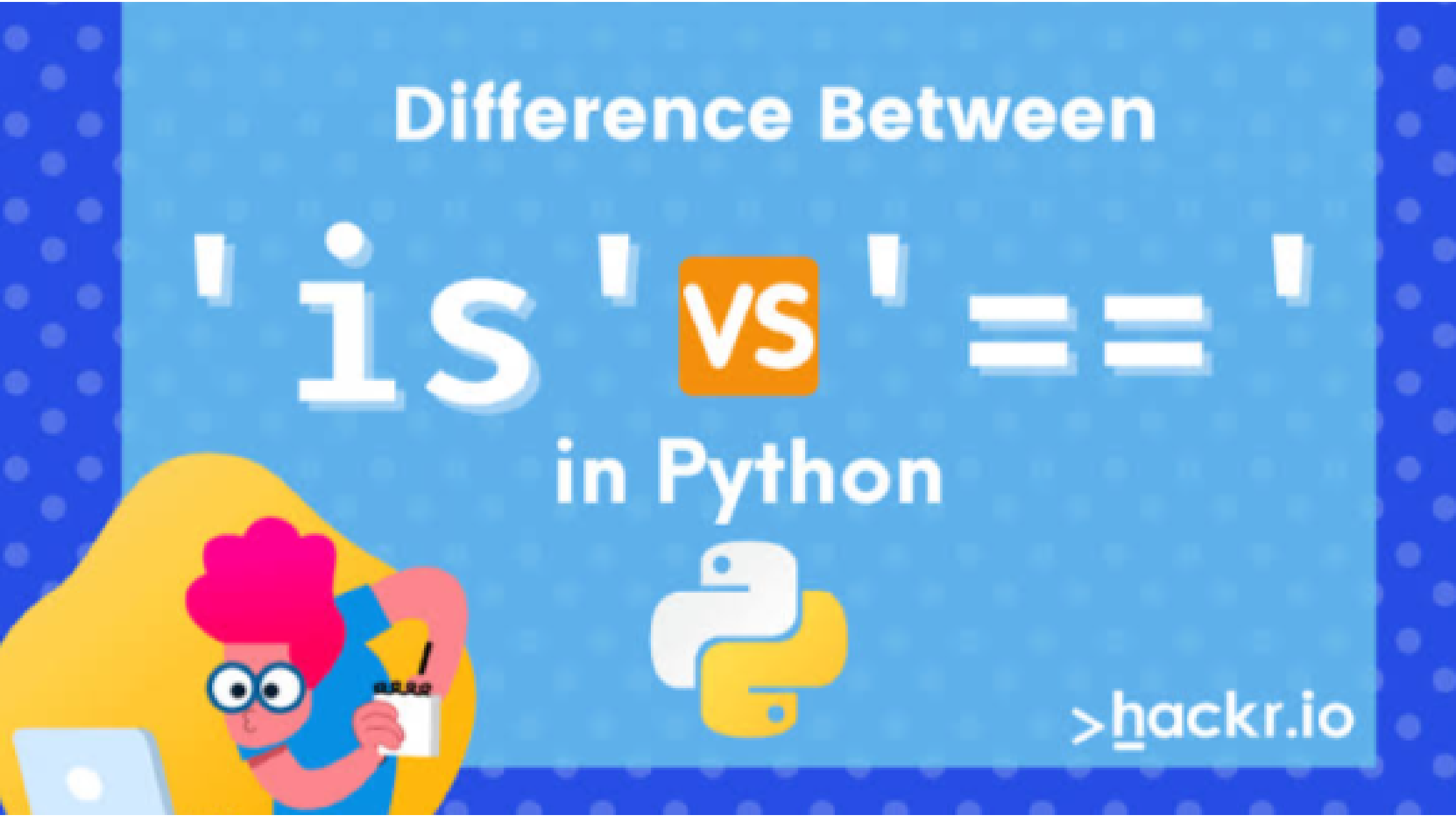
Python


Difference Between
'is' VS '=='
in Python
>hackr.io

Difference between == and is operator in Python

When comparing objects in Python, the identity operator is frequently used in contexts where the equality operator == should be. In reality, it is almost never a good idea to use it when comparing data.

What is == Operator?

To compare objects **based on their values**, Python's equality operators (==) are employed. It calls the left object's __eq__() class method, which specifies the criteria for determining equality. However, these constraints are typically written so that the equality operator == returns True if two objects, have the same value, and returns False if both have different value

What is the 'is' Operator?

Python identity operators (is, is not) are used to compare objects **based on their identity**. When the variables on either side of an operator point at the exact same object, the "**is**" operator's evaluation is true. Otherwise, it would provide us with a false assessment.

Identity vs Equality operators

```
# Equality operator
a=10
b=10

Case 1:
# Return True because both a and b have the same value
print(a==b)
True

Case 2:
# Return True because both a and b is pointing to the same object
print(id(a))
2813000247664

print(id(b))
2813000247664

a is b
True

Case 3:
# Here variable a is assigned to new variable c,
# which holds same object and same memory location
c=a
id(c)
2813000247664

a is c
True
```

Example 1:

Python3

```
list1 = []
list2 = []
list3 = list1

# case 1
if (list1 == list2):
    print("True")
else:
    print("False")

# case 2
if (list1 is list2):
    print("True")
else:
    print("False")

# case 3
if (list1 is list3):
    print("True")
else:
    print("False")

# case 4
list3 = list3 + list2

if (list1 is list3):
    print("True")
else:
    print("False")
```

Output:

```
True

False

True

False
```

Explanation:

- *The output of the first case*, if the condition is “True” as both list1 and list2 are empty lists
- *The output of the second case*, if the condition shows “False” because two empty lists are at different memory locations. Hence list1 and list2 refer to different objects. We can check it with id() function in python which returns the “identity” of an object.
- *The output of the third case*, if the condition is “True” as both list1 and list3 are pointing to the same object.
- *The output of the fourth case*, if the condition is “False” because the concatenation of two lists always produces a new list.

Example 2:

This shows that list1 and list2 refer to different objects.

Python3

```
list1 = []
list2 = []

print(id(list1))
print(id(list2))
```

Output:

```
139877155242696

139877155253640
```

The difference between == and is operators in Python

Parameters	is Operator	== Operator
Name	The 'is' is known as the identity operator.	The '==' is known as the equality operator.
Uses	When the variables on either side of an operator point at the exact same object, the **is** operator's evaluation is true. Otherwise, it will evaluate as False.	When the variables on either side have the exact same value, the **==** operator evaluation is true. Otherwise, it will evaluate as False.