

Is vs == in python

In Python, `is` and `==` are both comparison operators that can be used to check if two values are equal. However there are some important differences between the two that you should be aware of.

The `is` operator compares the identity of two objects, while the `==` operator compares the values of the objects. This means that `is` will only return `True` if the objects being compared are the exact same object in memory while `==` will return `True` if the objects have the same values.

For Example:

```
c=[1,2,3]
d=[1,2,3]

print(c is d) #False
print(c==d) #True
```

In this case `a` and `b` are two separate lists that have the same values, so `==` returns `True`. However, `a` and `b` are not the same object in memory, so `is` returns `False`.

One important thing to note is that, in Python, strings and integers are immutable, which means that once they are created their value cannot be changed. This means that for strings and integers, `is` and `==` will always return the same result.

```
m1="Munawar"
m2="Munawar"
print(m1 is m2)
print(m1==m2)

int1=5
int2=5
print(int1 is int2)
print(int1==int2)
```

In these cases a, and b are both pointing to the same object in memory, so `is` and `==` both return `True`.

For mutable objects such as lists and dictionaries, `is` and `==` can behave differently. In general, you should use `==` when you want to check if two objects are the same object in memory.

Source Code

```
a=4
b="4"
# both are operators
print(a is b) # exact location of object in memory
print(a==b)  # value

c=[1,2,3]
d=[1,2,3]

print(c is d) #false
print(c==d)  #True

t1=(1,2,3)
t2=(1,2,3)

print(t1 is t2)
print(t1==t2)

n1=None
n2=None
print(n1 is n2)
print(n1 is None)
print(n1==n2)

m1="Munawar"
m2="Munawar"
print(m1 is m2)
print(m1==m2)
```

```
int1=5  
int2=5  
print(int1 is int2)  
print(int1==int2)
```

Thank You