Python Programming

RGM College of Engineering & Technology (Autonomous)

Department of Computer Science & Engineering

Academic Year : 2020-2021

PYTHON OPERATORS-2

Guido Van Rossum

Learning Mantra

If you really strong in the basics, then remaining things will become so easy.

Agenda:

- Relational Operators (or) Comparison Operators
- Equality Operators

2. Relational Operators (or) Comparison Operators

Following are the relational operators used in Python:

1. Less than (<)
2. Greater than (>)
3. Less than or Equal to (<=)
4. Greater than or Equal to (>=)

i) We can apply relational operators for number types

Eg:

```
a = 10
b = 20
print('a < b is', a<b)        ➔True
print('a <= b is', a<=b)      ➔True
print('a > b is', a>b)        ➔False
print('a >= b is', a>=b)      ➔False
```

ii) We can apply relational operators for 'str' type also, here comparison is performed based on ASCII or Unicode values.

How to know the Unicode or ASCII value of any character?

- By using **ord() function**, we can get the ASCII value of any character.

Eg:

print(ord('a')) ➔97

print(ord('A')) ➔65

- If you know the ASCII value and to find the corresponding character, you need to use the **chr() function**.

print(chr(97)) ➔a

print(chr(65)) ➔A

Eg:

```
s1 = 'karthi'
s2 = 'sahasra'
print(s1<s2)        ➔True
print(s1<=s2)       ➔True
print(s1>s2)        ➔False
print(s1>=s2)       ➔False
```

iii) We can apply relational operators even for boolean types also.

Eg:

print(True > False) ➔True

print(True >= False) ➔True

print(True < False) ➔False

print(True <= False) ➔False

print(10 > 'karthi') ➔ **TypeError:** '>' not supported between instances of 'int' and 'str'

Eg:

```
a = 10
b = 20
if a>b:
    print('a is greater than b')
else:
    print('a is not greater than b')
```

Output:

```
a is not greater than b
```

iv) Chaining of relational operators:

- Chaining of relational operators is possible.
- In the chaining, if all comparisons returns True then only result is True.
- If at least one comparison returns False then the result is False.

Eg:

```
print(10<20)          ➔True
print(10<20<30)       ➔True
print(10<20<30<40)    ➔True
print(10<20<30<40>50) ➔False
```

3. Equality Operators:

- Equality operators are used to check whether the given two values are equal or not.
- The following are the equality operators used in Python.
  1. Equal to (==)
  2. Not Equal to (!=)

Eg:

print(10==20) ➔False

print(10!=20) ➔True

print(1==True) ➔True

print(10==10.0) ➔True

print('karthi'=='karthi') ➔True

We can apply Equality operators for any type even for incompatible types also.

Eg:

print(10=='karthi') ➔False

print(10=='10') ➔False

Note:

- Chaining concept is applicable for equality operators.
- If at least one comparison returns False then the result is False. otherwise the result is True.

Eg:

print(10==20==30==40) ➔False

print(10==10==10==10) ➔True

What is the Difference between '==' and 'is' operators?

is Operator:

- '**is**' operator meant for reference or address comparison.

When **a is b** returns true?

Ans: Whenever 'a' and 'b' pointing to the same object, then only 'a is b' returns true, which is nothing but reference comparison (or) Address comparison.

== Operator:

- '==' is meant for content comparison.

Eg:

```
l1 = [10,20,30]
l2 = [10,20,30]
print(id(l1))          ➔2689910878664
print(id(l2))          ➔2689910921864
print(l1 is l2)        ➔False      # Reference Comparison
print(l1 == l2)        ➔True       # Content Comparison
l3 = l1                ➔ l3 is also pointing to l1
print(id(l3))          ➔2689910878664
print(l1 is l3)        ➔True
print(l1 == l3)        ➔True
```

Any question?

If you try to practice programs yourself, then you will learn many things automatically

Spend few minutes and then enjoy the study

Thank You