Relational Operators

17 June 2025 15:24

Definition:Relational operators are used to **compare two values**. The result of a relational operation is always a **Boolean value** (True or False).

Types of Relational Operators:

1. == → Equal to
2. != → Not equal to
3. > → Greater than
4. < → Less than
5. >= → Greater than or equal to
6. <= → Less than or equal to

Note:

1. For **single value data types** (like int, float, bool, str), comparison is done based on **value only**.
2. For **collection types** (like list, tuple, dict, set), comparison checks:
   - **Value**
   - **Data type**
   - **Order/Position** (except in sets and dictionaries)

1.==(Equal to Operator):-Used to check whether two values are **equal**.

Syntax:- Operand1==Operand2

Examples:-

```
0 == 0.0              # True (int vs float but same value)
7 == 7.0              # True
[1,2,3] == (1,2,3)    # False (list vs tuple)
'hai' == 'HAI'        # False (case-sensitive)
(1,2,3) == (1,3,2)    # False (different order)
{1,2,3} == {1,3,2}    # True (set is unordered)
{'a':10} == {'a':10}  # True (same keys and values)
```

2.**!= (Not Equal to Operator):-** Used to Checks whether two values are not equal.

Syntax:- Operand1!=Operand2

Example:-

```
10 != 10          # False
10 != 20          # True
[1,2] != [2,1]    # True
'a' != 'A'        # True
(1,2) != (1,2)    # False
```

3.> (Greater Than):- used to Checks if the left operand is greater than the right.

Syntax:- Operand1>Operand2

Note:

- Python compares strings **character by character** using their **ASCII values**.

ASCII Values of Uppercase and Lower Case Characters:-

A-Z ⟶ 65-88,
a-z ⟶ 97-120,

Rule

ASCII Values of Uppercase and Lower Case Characters:-

A-Z ⟶ 65-88,

a-z ⟶ 97-120,

Example:-

```
10 > 5          #Output:- True
5 > 10          #Output:- False
'a' > 'A'       # Output:-True (based on ASCII)
[2,3] > [1,5]   #Output:- True (compares element-wise)

hai'>'hello'    #Output:-False
(2+3j)>2
Output:-Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    (2+3j)>2
TypeError: '>' not supported between instances of 'complex' and 'int'
(2+3j) >(1+1j)  #Output:- Error
'python'>'abc'                  #Output:-True
'ABC'>'abc'                     #Output:-False
'abcdef'>'abcdem'               #Output:-False
[10,20,30]>[10,20,30]           #Output:-False
[1,2,3,4]>[1,2,3,4]             #Output:-False
[100,7,3]>[23,1000,200,500]     #Output:-True
(1,2,3,4)>(1,2,3)               #Output:-True
{10,20,30,40}>{1,2,3}           #Output:-False
```

Rule

Collection1 > Collection2
Val1,val2,val3…>v1,v2,v3
If Val1 >v1 ----->True
If val1<v1------>False
If val1==v1------>then it goes to compare other values

- ✧ When we use the **greater than (>) operator** between two collections (like lists or tuples), Python **compares elements one by one**, in order, from left to right.

Note 1: Dictionary and Complex Number Comparisons

"In case of dictionary and complex it not possible, it will give the error as output."

Explanation:

- Python does **not allow comparison** like >, < between:
  - **Dictionaries**: e.g., {'a': 10} > {'b': 20} → Error
  - **Complex numbers**: e.g., (2+3j) > (1+2j) → Error

These comparisons raise a Type Error because Python doesn't define an ordering for these types.

Note 2: Set Comparisons Using >(great Ethan)

"In case of set, if all the values of 2nd collection are in 1st collection, it will give output as True. Otherwise, it will give False."

```
Examples:- {10,20,30,40}>{1,2,3}      #Output:-False
           {12,34,56,75}>{12,34}      # Output:-True
```

4. < (Less Than): Used to check if the left operand is smaller than the right operand**.**

Syntax:-Operand1 < Operand2

Note:

- Python compares strings **character by character** using their **ASCII values**.

ASCII Values:

- A-Z → 65–88
- a-z → 97–120

Examples:

```
10 < 5          # Output: False
5 < 10          # Output: True
'a' < 'A'       # Output: False (based on ASCII)
```

```
[2, 3] < [5, 6]                  # Output: True (compares element-wise)

'hai' < 'hello'                  # Output: True
(2+3j) < 2                       #  Error (TypeError)
(2+3j) < (1+1j)                  #  Error (TypeError)

'python' < 'abc'                 # Output: False
'ABC' < 'abc'                    # Output: True
'abcdef' < 'abcdem'              # Output: True

[10, 20, 30] < [10, 20, 30]      # Output: False
[1, 2, 3, 4] < [1, 2, 3, 4]      # Output: False
[100, 7, 3] < [1000, 200]        # Output: True
(1, 2, 3) < (1, 2, 4)            # Output: True
{10, 20, 30, 40} < {1, 2, 3}     # Output: False
```

Note 1: Dictionary and Complex Number Comparisons

In case of dictionary and complex, it is not possible. It will give an error as output.

Explanation:

Python **does not allow** comparisons like < or > between:

- **Dictionaries:**
  {'a': 10} < {'b': 20} → TypeError
- **Complex Numbers:**
  (2+3j) < (1+1j) → TypeError

Note 2: Set Comparisons Using < (Less Than)

In case of sets, if all the values of the 1st collection are **not fully present** in the 2nd collection, output is **False**.

```
{10, 20, 30, 40} < {1, 2, 3}      # Output: False
{12, 34} < {12, 34, 56, 75}       # Output: True
```

5. <= (Less Than or Equal To):

Used to check if the **left operand is less than or equal to the right operand**.

Syntax:- Operand1 <= Operand2

Note:

- Python compares values **character by character** or **element by element** depending on the data type.
- For **strings**, comparisons are based on **ASCII values**.

ASCII Values of Characters:

- **A–Z** → 65–88
- **a–z** → 97–120

Example:-

```
10 <= 10                 # Output: True
5 <= 10                  # Output: True
15 <= 10                 # Output: False

'a' <= 'A'               # Output: False  (ASCII of 'a'=97, 'A'=65)
'A' <= 'a'               # Output: True

[2, 3] <= [2, 3]         # Output: True
[1, 2, 3] <= [1, 2, 4]   # Output: True
[5, 6] <= [2, 3]         # Output: False
```

```
'hai' <= 'hello'          # Output: True
'python' <= 'abc'         # Output: False
'ABC' <= 'abc'            # Output: True
'abcdef' <= 'abcdem'      # Output: True

(1, 2, 3) <= (1, 2, 3)    # Output: True
(1, 2) <= (1, 2, 0)       # Output: True
(2, 3) <= (1, 2)          # Output: False

{1, 2} <= {1, 2, 3}       # Output: True
{1, 2, 3} <= {1, 2, 3}    # Output: True
{1, 2, 3, 4} <= {1, 2, 3} # Output: False
```

Note 1: Dictionary and Complex Number Comparisons

In case of **dictionary** and **complex numbers**, it is not possible.
Python will raise a **TypeError**.

Note 2: Set Comparisons Using <= (Less Than or Equal To)

In case of **sets**, A <= B returns **True** if all elements of set A are in set B.

- ❖ Returns **True** even if both sets are **equal** (not proper subset).
- ❖ Returns **False** if A has any element not in B.

6. >= (Greater Than or Equal To):

Used to check if the **left operand is greater than or equal to the right operand**.

```
Syntax:Operand1 >= Operand2
```

Examples:-

```
10 >= 5             # Output: True
10 >= 10            # Output: True
5 >= 10             # Output: False

'a' >= 'A'          # Output: True  (97 >= 65)
'abc' >= 'abc'      # Output: True
'abc' >= 'abcd'     # Output: False

[1, 2, 3] >= [1, 2, 3]    # Output: True
[1, 2, 4] >= [1, 2, 3]    # Output: True
[1, 2, 2] >= [1, 2, 3]    # Output: False

'hello' >= 'hai'    # Output: True
'abc' >= 'ABC'      # Output: True
'Python' >= 'python'  # Output: False

(1, 2, 3) >= (1, 2, 3)    # Output: True
(1, 2, 4) >= (1, 2, 3)    # Output: True
(1, 2, 1) >= (1, 2, 3)    # Output: False
```

Note: it is not supported for complex and dictionary.

Note :it work like a greater than.