Operators

20 June 2024 04:09 PM

Operators are the elements used for performing the operations on the operands.

Types of Operators:

- 1. Arithmetic Operators (+, -, *, **, /, //, %)
- 2. Relational Operators (and, or, not)
- 3. Relational Operators (==, <, >, <=, >=, !=)
- 4. Bitwise Operators (&, |, ^, ~, >>, <<)
- 5. Assignment Operators (=, += , -= , *= , /= , //= , %= , **=)
- 6. Membership Operators (in, not in)
- 7. Identity Operators (is, is not)
- Arithmetic Operators:

It is used for performing arithmetic operations on given operands.

Ope	erator	Number to number	Number to CDT	CDT to CDT
	+	Addition	Cannot be performed	Concatenation, provided data type should be same cannot be performed on set and dictionaries
	-	Subtraction	Cannot be performed	Only on sets (difference operation)
	*	Multiplication	Concatenation (repetition of specified elements)	Cannot be performed

/ (Normal division Or True division):

Performs the division and gives the output in the form of float

Example: 5/2 #>> 2.5

// (Floor division):

Performs division and eliminates the decimal values from the output.

Example: 5//2 #>> 2

** (Power Operator):

 $A^{**}B \xrightarrow{\dots > A}$ to the power of B

3**3 = 27

Relational Operators:

Relational Operators are used for returning a Boolean value as a input.

These relational operators are used in conditional statements and while loop.

$$>$$
 , $>=$, $==$, $<$, $>$, $<=$, $!=$

Number to number comparisons can be done easily.

But in case of strings, comparison is done based on the ascii values of characters.



Logical Operators:

 $Logical\ Operators\ are\ used\ to\ return\ Boolean\ values\ as\ output.$

Logical and:

It returns True when both operands are true, else it returns False.

Truth table of and:

Operand 1 Operator Operand 2 Output

True	And	True	True
True	And	False	False
False	And	True	False
False	And	False	False

Example:

```
1 | 1 | and 6 | # T1 | and T2 --> T2 | 6 |

1 | 'hai' | and [90] | # T1 | and T2 -> T2 |

[90]

1 | 'hai' | and [] | #T1 | and F2 -> F2 |

[]

1 | '' | and [78] | # F1 | and T2 -> F1 |

...

1 | '' | and [] | # F1 | and F2 -> F2 |
```

Logical Or Operator:

It returns True if any one operand is True, else it returns False.

Truth table:

Operand1	Operator	Operand 2	Output
True	Or	True	True
False	Or	True	True
True	Or	False	True
False	Or	False	False

```
1 '' or [] # F1 or F2 --> F2

[]

1 '' or [109] # F1 OR T2 --> T2

[109]

1 'uyt' or [] #T1 OR F2 --> T1

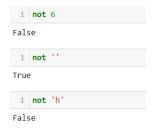
'uyt'

1 'uyt' or [765] #T1 OR T2 --> T1

'uyt'
```

<u>Logical Not Operator:</u>

It is used for performing negation operation. Logical Not is used only with one operand.



Assignment Operator:

They are used for assigning some values to the variable. By using = operator, we can assign the value for a variable.

```
1 a=10

2 a+=20

3 a

30

1 a-=10

2 a

20

1 a*=10

2 a

200

1 a/=10

2 a

20.0

1 a**=3

2 a

8000.0
```

Membership Operator:

It is used for checking whether the given element is part of a given collection or not.

In Operator:

It returns True if a value is present in a given collection, else it will return False.

Syntax:

SVDT / CDT in Collection

Note:

- Right side value must be a CDT
- If we use string in RHS, then mandatorily, we need to use string in RHS as well
- In RHS if we use a set or dictionary, we should use only immutables in LHS



Not in Operator:

It returns True if value is present in given collection, else it will False.

Syntax:

SVDT/CDT not in CDT

```
1 1 not in [11,22,33]

True

1 {1} not in {11,22,33}

True

1 1 not in {11:2, 1:3}

False
```

Identity Operator:

Note:

- o If same value of immutable types are stored in a variable, then they will point to same memory address.
- o But even though same mutable types are stored in a variable, they will point to different memory address.
- o == operator compares values directly
- \circ Is not operator compares memory address.

These are the operators used for comparing memory addresses.

Is operator:

It returns True if variables are pointing to same memory addresses, else returns False.

Syntax:

```
operand1 id operand2
```

Is not operator:

It returns True if variables are not pointing to same memory address, else it returns False.

Syntax:

```
operand1 is not operand2
```

Example:

```
1  a={11:'hai'}
2  b = {11:'hai'}
3  print('a=b -->',a=b)
4  print('a is b -->', a is b)
5  print('a is not b -->', a is not b)

a=b --> True
a is b --> False
a is not b --> True

1  a=(11,'hai')
2  b = (11,'hai')
3  print('a=b -->',a=b)
4  print('a is b -->', a is not b)

a=b --> True
a is b --> False
a is not b --> True
```

Bitwise Operator:

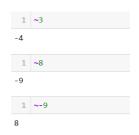
These are the operators used for performing operations on binary values of given numbers.

Bitwise & ---> returns 1 when both operands are 1, else 0 Bitwise | ---> returns 0 when both operands are 0, else 1 Bitwise $^{\wedge}$ ---> returns 0 if both operands are same, else 1

```
1 1 & 2 --> 2
                          2 & 3 --> 2
2 0001
3 &
4 0010
                           0010
                           0011
0010 --> 2
                         2 | 3 --> 3
                           0010
                           0011
12 ----
13 0011 --->3
                          0011 --> 3
14 ------
15 1 ^ 2 --> 3
                     2 ^ 3 --> 1
16 0001
17 ^
                          0010
18 0010
                           0011
19 ----
20 0011 --->3
                          00101 --> 1
```

Bitwise Not (\sim):

Bitwise ~ performs 2's complements i.e., $\sim n = -(n+1)$



Right shift (>>):

Shifts binary values to right side by specified number of positions.

Syntax:

Operand >> no. of positions to shift

Left shift

Shifts binary values to left side by specified number of positions.

Syntax

Operand << no. of positions to shift

Print and eval

24 June 2024 12:28 PM

Print:

- It is used for printing the specified value in the console.
- By default, print function will print given data and navigate to next line.
- Syntax:

Print(value1, value2, value3,..... value-n, sep=' ', end='\n')

• We can change the functionality of print by changing the values of sep and end

```
1 print(90)
90

1 print(100)
100

1 print(90,100,sep=',', end=' ')
90,100

1 print(200,300)
200 300

1 print('hai')
hai
```

Comments:

- Comments are non-executable statements
- Comments are used by the developers for providing hints about their code.

We have 2 types of comments:

- 1. Single line comment
- 2. Multi-line comment
- 1. Single line comment:

Eg: # print(100)

2. Multi-line comment:

Eg: "' print('hai' print('hello')

Collecting data from the user during runtime and assigning to a variable:

- 1. Input() function is responsible for collecting the data from the user
- 2. Output format of collected data will be in string format
- 3. So we can typecast based on our requirement

Eval function:

It is a special function which is responsible for evaluating the data into its equivalent submitted data type.