

Operators in Python

Operators

Hello and welcome to this exciting module on Python Operators.

In this module you will learn about Python operators and their different types.

So, let's start with the most fundamental question: what are operators in Python?

Well, an operator is a symbol used to perform mathematical and logical operations in a program. Operator is a special symbol that tells the compiler to perform mathematical or logical operations. Python programming language supports a rich set of operators that are classified as follows.

Arithmetic Operators, Bitwise Operators, Assignment Operators, Relational Or Comparison Operators, Logical Operators, Membership Operators (in, not in), Identity Operators (is, is not)

You will go through each operator in detail with plenty of examples in a few minutes.

Let's start with the basic arithmetic operators. An arithmetic operator is a mathematical function that performs a calculation on two operands. Here operand signifies that variable or a value used in the arithmetic operation. It's the most well-known operator that most of us learned as kids.

Addition, subtraction, multiplication, and division are just a few of the typical operators we use in our daily lives.

Addition adds the two operands, subtraction deducts one operand from the other operand. Multiplication multiplies the two operands and division divides the left operand by the right one. Please note that division always results in a float or gives a decimal answer.

Moving on to the modulus operator which gives the remainder of the division.

Floor division results in the whole number adjusted to the left in the number line.

Exponents return the first operand raised to power with the second operand.

Now it's important to understand the difference between division and modulus operators. The quotient is computed using the division operator, and the remainder is computed using the modulus operator. It's worth noting that the modulus operator can work with both integer and float variables.

This leads us to the next point: the division operator in Python can be used with mixed data types.

Let's have a look at an example. The variables can be of either integer or float data types, however for all division combinations, the output is float data type.

This is because the operands upgrade or convert themselves to higher data types. In this case the float data type will be the result.

Hope this demonstration helps to understand how well Python handles mixed data types.

Allright! Hope you all have understood about arithmetic operators.



Relational and Comparison Operators

Hello and welcome, everyone! It will be interesting because we will be examining relational and comparison operators in this module.

You will learn about six different types of relational operators that will be discussed.

Let's get started with the definition of relational or comparison operators.

The symbols used to compare two values are known as relational operators. They're used to see how two values or variables relate to each other. There are two possible outcomes for any relational operator: TRUE or FALSE. For instance, your grades are higher than Sam's, or $9 > 6$. The relational operators, in simple terms, are used to define conditions in a program.

Take a look at the table for more information about relational operators.

A and B in these instances can carry any type of value, including integers, floats, and expressions.

Greater than: True if left operand is greater than the right

Less than: True if left operand is less than the right

Less than or equal to: True if left operand is less than or equal to the right

Greater than or equal to: True if left operand is greater than or equal to the right

Equal to: True if both operands are equal

Not equal to – True if operands are not equal

I hope that you understand what a relational or comparison operator is.



Logical Operators

Dear Learners! I hope you enjoy learning about operators. Until now, we've covered arithmetic, assignment, and relational or comparison operators.

In this module, you will learn about logical operators with examples.

Let's start with a definition of logical operators.

The logical operators are symbols for combining many conditions into a single condition. It's used to combine multiple conditional expressions that are either true or false.

The first operator is a Logical AND, which returns TRUE if all conditions are true and FALSE otherwise.

Similarly, with logical OR, if all conditions are FALSE, the operator returns FALSE. Logical OR returns TRUE if any of the conditions returns TRUE.

If the condition is FALSE, the logical NOT operator returns TRUE, and if the condition is TRUE, it returns FALSE.

This module has covered a fascinating topic that can be used for a wide range of Python programming applications.



Membership Operators

Hi,

With examples, you will learn about the membership operator in this module.

Let's start with a definition of membership operators.

In Python, the membership operators are used to see if a value is located within a sequence. Because the string Hello is present in the string Hello world!, 'Hence the output will become True.

There are two types of membership operators: in and not in.

In - returns true if the specified value is present in the sequence.

For example it checks if 5 is present in x. Hence the output is True.

Not in - returns True if value/variable is not found in the sequence.

For example it checks if 5 is not present in x. Hence the output is False.

Let's have a look at another example to help you understand.

Let's assume a has the value of "Welcome to python"

In the first scenario Greetings is not present in the sequence so the output will be true.

In the second scenario Welcome is present in the sequence so the output will be true.

Hope you have learned about how we may use this operator to locate a single item in a long sequence;

Identity Operators

Welcome back to the Python language's operators module. You've already learned about a couple of operators and how to use them.

In this module you will learn about identity operators and their types with examples.

Let's start with a definition of identity operators.

Identity operators are used to determine whether a value is of a certain class or type. They are usually used to determine the type of data a certain variable contains.

There are two types of identity operators.

is - will return output as True if the operands are identical or refer to the same object.

Such as x is true.

is not - will return output as True if the operands are not identical or do not refer to the same object.

is - will return output as True if the operands are identical or refer to the same object.

In the given example

```
x = [1, 2, 3, 4, 5, 6]
```

```
print(x is x)
```

We are trying to verify x is the same object with itself.

The output becomes "True"

Is not - will return output as True if the operands are not identical or do not refer to the same object.

In the given example

```
x = [1, 2, 3, 4, 5, 6]
```

```
print(x is not x)
```

We are trying to verify x is not the same object with itself.

The output becomes "False"

Let us understand a new built-in function called type function.

The type() function returns the type of the specified object.

Python is truly object oriented language, where every variable is an object, and every piece of code is a method.

Let us see a few examples using the type function.

#Assign the values to a,b,c,d objects

```
a = 10
```

```
b = "Hello World"
```

```
c = 33.33
```

```
d = 2+3j
```

#use type() function to understand the type of #the data stored in the variable or referred by #the object

```
print(type(a))
```

```
print(type(b))
```

```
print(type(c))
```

```
print(type(d))
```

The output would be:

The Output:

```
<class 'int'>
```

```
<class 'str'>
```

```
<class 'float'>
```

```
<class 'complex'>
```

Consider an example of the 'is' operator.

Let's see if the x variable is an integer in the first case.

To check with 'is not' operator, you will use the type function in the statement.

Since the data type of x is float and it is not an integer, the output will be true.

Similarly in the next example let's check using 'is' operator.

The value of x is an integer therefore the output is true.

As a last note, I'd like to point out that identity operators in Python are used to determine whether a value is of a certain class or type.



Bitwise Operators

Good morning, everyone! We are learning new concepts of operators in the Python language with incredible ease.

We'll go through bitwise operators in depth in this section, with plenty of examples. You'll also learn how to convert integers to binary.

Let's start with a definition of bitwise operators.

The bitwise operators are used to perform bit level operations.

When we use the bitwise operators, the operations are performed based on the binary values.

The result is always returned in decimal form.

Note: the bitwise operator works only on integers.

Let's start with how to convert numbers or integers to binary?

In order to convert the numbers, you need to understand the concept of MSB and LSB.

Let's discuss what MSB and LSB are.

The Most Significant Bit (MSB) is the bit in a multiple-bit binary number with the largest value. This is usually the bit farthest to the left, or the bit transmitted first in a sequence.

The Least Significant Bit (LSB) is the bit in a binary number which is of the lowest numerical value.

In this example, MSB is 1 and LSB is 0.

Let's see how to convert Binary to Decimal?

The base of the binary number system is 2, and it is increased by the factor of two. The first digit has 2^0 weights, the second has 2^1 weights, the weight of the third digit is 2^2 and so on.

In binary to decimal conversion when there is one in a digit position of a binary number, then the weight of the position is added. But when there is the zero in a binary position the weight of the position is disregarded.

Let us say for example how the binary number 1010 is converted to decimal.

The LSB 0 is multiplied with 2^0 weight

The next bit 1 is multiplied with 2^1 weight

The next bit 0 is multiplied with 2^2 weight

At last the MSB 1 is multiplied with 2^3 weight

Finally add all the obtained multiplication results. The result would become 10

An easy method of converting decimal to binary number equivalents is to write down the decimal number and to continually divide-by-2 (two) to give a result and a remainder of either a "1" or a "0" until the final result equals zero.

Let us look into an example of decimal to binary conversion with the number 29. We need to do successive division by 2 until getting the remainder as either 0 or 1. Finally collect all the remainders from the bottom to top. The result would be 11101.

Now that you've understood the concept of conversion. Let's have a look at the different types of bitwise operators.

Bitwise and operator Returns 1 if both the bits are 1 otherwise 0.

Bitwise or operator Returns 1 if either of the bits is 1 else 0.

Bitwise xor operator Returns 1 if one of the bits is 1 and the other is 0 else returns false.

Bitwise not operator Returns one's complement of the number. You can consider it as flipping.

Now let us look at the truth table to understand these conditions better.

Let's recap the Bitwise AND operator - It will return 1 only if both the operators are 1. Therefore in this case the rest are 0.

Similarly the bitwise OR operator will return 1 if any one of the two bits is 1. Therefore only one condition is 0.

Now the bitwise XOR operator will return 1 if one of the bits is 1 and the other is 0. So if the bits are the same then the output will be 0.

In the case of bitwise NOT in the input is 1 then the output will be 0 and if the input is 0 then the output will be 1. The output will be flip or inverse of the input.

Now we'll look at the bitwise LEFT SHIFT OPERATOR, which shifts all the bits to the left by a defined number of positions. It shifts the number's bits to the left and fills any vacancies with 0 as a result. Every left shift operation results with an equivalent answer to a multiplication of the given number by 2.

In the case of bitwise operations, THE RIGHT SHIFT OPERATOR shifts all the bits to the right by a predetermined number of places. As a result, the bits of the number are shifted to the right, and vacancies are filled with 0 as a result. Every right shift operation results with an equivalent answer to a division of the given number by 2.

Let us look into a few examples for Left Shift Operator

.

Example 1:

Let us consider $a = 5 = 0000\ 0101$

$a \ll 1 = 0000\ 1010 = 10$

$a \ll 2 = 0001\ 0100 = 20$

Example 2:

Let us consider $b = -10 = 1111\ 0110$

$b \ll 1 = 0000\ 1010 = -20$

$b \ll 2 = 0001\ 0100 = -40$

Let us look into a few examples for Right Shift Operator

Example 1:

Let us consider $a = 10$

$a \gg 1 = 5$

Example 2:

Let us consider $a = -10$

$a \gg 1 = -5$

I hope you have understood this complex concept of bitwise operators clearly.



vityarthi

Assignment Operators

Welcome back to the python language's operators module. In this module, you will learn about assignment operators and their many types, as well as a few examples.

Let's get started.

The first thing you should learn is about assignment operators.

Assignment operators, such as $a=5$, are used to assign the right-hand side value (Rvalue) to the left-hand side variable (Lvalue).

The majority of assignment operators, as well as arithmetic operators, are employed together for various operations.

The assignment operators are used to assign the right hand side value (Rvalue) to the left hand side variable (Lvalue).

Let's take a look at each of these assignment operators one by one to see what they do.
Assign the value from the right side to the left side variable.

Please keep in mind that at least one operand must be the same as the one used on the left. Only when a variable becomes common in both LHS and RHS can it be used.

Add both left and right hand side values and assign to the left hand side variable.

Subtract right hand side value from left hand side variable value and assign to left hand side variable.

Multiply right hand side value with left hand side variable value and assign to left hand side variable.

Divide left hand side variable value with right hand side variable value and assign to left hand side variable.

Divide left hand side variable value with right hand side variable value and store the remainder into left hand side variable.

Floor Division: It divide left operand with right operand and then assign the value(floor) to left operand

As previously stated, the above examples have A on both the right and left hand sides to employ the appropriate assignment operator.

Exponent - The left operand raised to the power of right.

The result of Bitwise AND is 1 if all the bits are 1 otherwise it is 0.

The result of Bitwise OR is 0 if all the bits are 0 otherwise it is 1.

The result of Bitwise XOR is 0 if all the bits are same otherwise it is 1.

The Bitwise left shift operator shifts all the bits to the left by specified number of positions.

The Bitwise right shift operator shifts all the bits to the right by specified number of positions.

Now, Let me say a few applications of bitwise operators.

- Useful in image processing, video processing, speech processing (these are called data compression) and data security (called encryption technique).
- Cryptography, computer graphics, hash functions, compression algorithms, and network protocols are just some examples where **bitwise operations** are extremely useful.

I'm sure that you'll be able to use all of the Python operators.
See you in the next module


Assignment Operators

Dear Learners! I hope you're enjoying yourself while you learn about operators. We've covered arithmetic operators and in this module we will discuss assignment operators.

What are assignment operators?

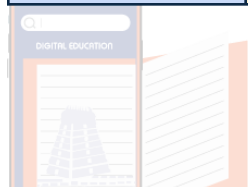
The assignment operators are used to assign the right hand side value (Rvalue) to the left hand side variable (Lvalue). The assignment operator is used in different variations along with arithmetic operators.

Let us look at the various assignment operators.



Operator	Meaning	Numerical Example
=	Assign the right hand side value to the left hand side variable.	A = 15
+=	Add both left and right hand side values and store the result into the left hand side variable.	A += 10 ⇒ A=A+10
-=	Subtract right hand side value from left hand side variable value and store the result into left hand side variable.	A -= B ⇒ A=A-B
*=	Multiply the right hand side value with the left hand side variable value and store the result into the left hand side variable.	A *= B ⇒ A=A*B

Operator	Meaning	Numerical Example
/=	Divide the left hand side variable value with the right hand side variable value and store the result into the left hand side variable.	A /= B $\Rightarrow A=A/B$
%=	Divide left hand side variable value with right hand side variable value and store the remainder into left hand side variable.	A %= B $\Rightarrow A=A\%B$
//=	Floor Division: Divide left operand with right operand and then assign the value(floor) to left operand.	A// = 5
=	Exponent - left operand raised to the power of right	A **= 5 $\Rightarrow A=A5$



vituorathi

Operator	Meaning	Numerical Example
&=	The result of Bitwise AND is 1 if all the bits are 1 otherwise it is 0.	A &= B $\Rightarrow A=A\&B$
 =	The result of Bitwise OR is 0 if all the bits are 0 otherwise it is 1	A = B $\Rightarrow A=A B$
^=	The result of Bitwise XOR is 0 if all the bits are same otherwise it is 1.	A ^= B $\Rightarrow A=A\^B$
>>=	The Bitwise left shift operator shifts all the bits to the left by specified number of positions	A >>= B $\Rightarrow A=A>>B$
>>=	The Bitwise right shift operator shifts all the bits to the right by specified number	A <<= B

	of positions	$\Rightarrow A=A \ll B$
--	--------------	-------------------------

These are the special operators that perform mathematical, logical, and bitwise operations. I hope you completely understand it.



vityarthi