

Operators

Operators are used to perform operations on variables and values.

Arithmetic operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Meaning	Example
+	Add two operands or unary plus	$x + y + 2$
-	Subtract right operand from the left or unary minus	$x - y - 2$
*	Multiply two operands	$x * y$
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	$x \% y$ (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	$x // y$
**	Exponent - left operand raised to the power of right	$x ** y$ (x to the power y)

Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	$x == y$
!=	Not equal	$x != y$

>	Greater than	$x > y$
<	Less than	$x < y$
>=	Greater than or equal to	$x \geq y$
<=	Less than or equal to	$x \leq y$

Logical Operators

Logical operators are used to combine conditional statements:

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Bitwise operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

Operator	Meaning	Example
&	Bitwise AND	$x \& y = 0$ (0000 0000)
	Bitwise OR	$x y = 14$ (0000 1110)
~	Bitwise NOT	$\sim x = -11$ (1111 0101)
^	Bitwise XOR	$x \wedge y = 14$ (0000 1110)

>>	Bitwise right shift	$x \gg 2 = 2$ (0000 0010)
<<	Bitwise left shift	$x \ll 2 = 40$ (0010 1000)

Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Assignment operators

Assignment operators are used in Python to assign values to variables.

Operator	Example	Equivalent to
=	$x = 5$	$x = 5$
+=	$x += 5$	$x = x + 5$
-=	$x -= 5$	$x = x - 5$
*=	$x *= 5$	$x = x * 5$
/=	$x /= 5$	$x = x / 5$
%=	$x \% = 5$	$x = x \% 5$
//=	$x //= 5$	$x = x // 5$

**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
 =	x = 5	x = x 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Operators

Operators are used to perform operations on variables and values.

Arithmetic operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Meaning	Example
+	Add two operands or unary plus	x + y+ 2

-	Subtract right operand from the left or unary minus	x - y- 2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

Logical operators are used to combine conditional statements:

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Bitwise operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
	Bitwise OR	x y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

Membership Operators

Membership operators are used to test if a sequence is presented in an object:

Operator	Description	Example
----------	-------------	---------

in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Assignment operators

Assignment operators are used in Python to assign values to variables.

Operator	Example	Equivalent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
=	x = 5	x = x 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5

<<=	x <<= 5	x = x << 5
-----	---------	------------

Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location:

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

if statement

if statement is the most simple decision making statement. It is used to decide whether a certain statement or block of statements will be executed or not i.e if a certain condition is true then a block of statement is executed otherwise not.

Syntax:

if condition:

Statements to execute if
condition is true

if- else

The if statement alone tells us that if a condition is true it will execute a block of statements and if the condition is false it won't. But what if we want to do something else if the condition is false. Here comes the else statement. We can use the else statement with if statement to execute a block of code when the condition is false.

Syntax:

if (condition):

Executes this block if
condition is true

else:

Executes this block if
condition is false

if-elif-else ladder

Here, a user can decide among multiple options. The if statements are executed from the top down. As soon as one of the conditions controlling the if is true, the statement associated with that if is executed, and the rest of the ladder is bypassed. If none of the conditions is true, then the final else statement will be executed.

Syntax:-

if (condition):

statement

elif (condition):

statement

.

.

else:

statement