

005 Python Operators

Operators and Operands

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The values the operator uses are called operands.

```
c = a + b
Here a and b are called operands and '+' is an operator
```

Python supports following operators.

Operator: Commands

Module of functions that provide the functionality of operators.

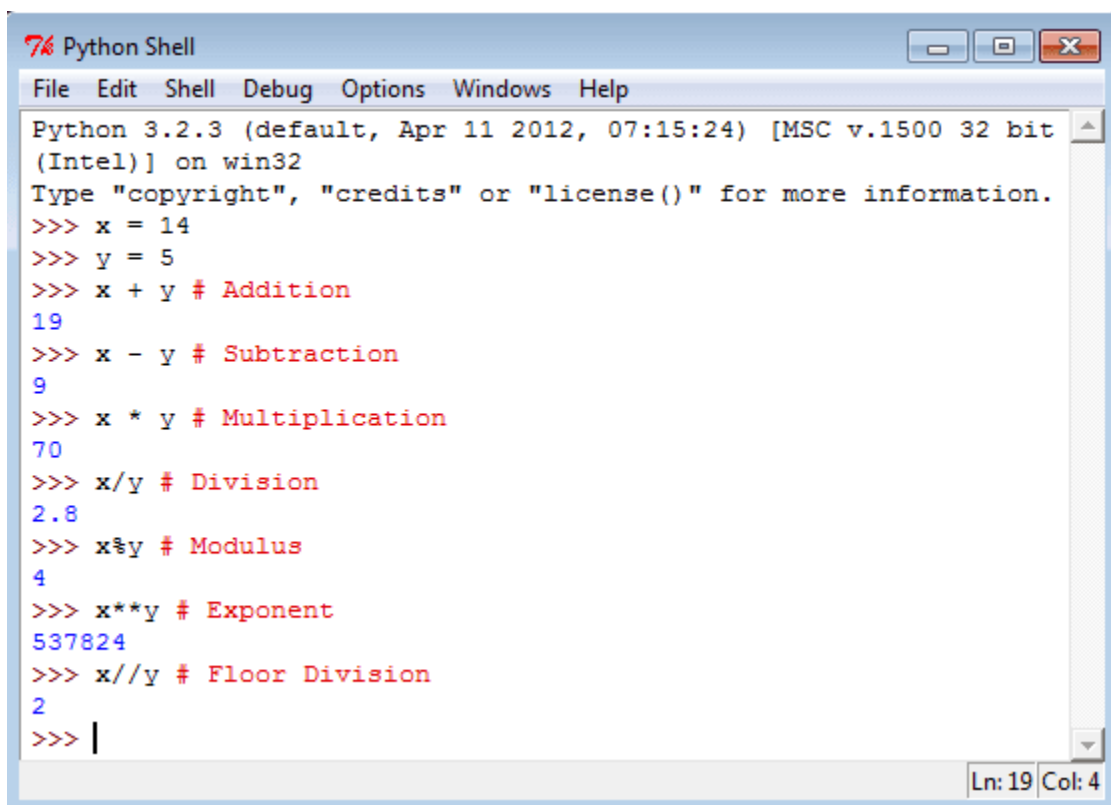
```
from operator import add, sub, mul, truediv, floordiv, mod, pow,
neg, abs
from operator import eq, ne, lt, le, gt, ge
from operator import and_, or_, not_
from operator import itemgetter, attrgetter, methodcaller
import operator as op
sorted_by_second = sorted(<collection>, key=op.itemgetter(1))
sorted_by_both = sorted(<collection>, key=op.itemgetter(1, 0))
product_of_elems = functools.reduce(op.mul, <collection>)
LogicOp = enum.Enum('LogicOp', {'AND': op.and_, 'OR' :
op.or_})
last_el = op.methodcaller('pop')(<list>)
```

Python Arithmetic Operators

Operator	Name	Example	Result
+	Addition	x+y	Sum of x and y.
-	Subtraction	x-y	Difference of x and y.

*	Multiplication	$x*y$	Product of x and y.
/	Division	x/y	Quotient of x and y.
%	Modulus	$x\%y$	Remainder of x divided by y.
**	Exponent	$x**y$	$x**y$ will give x to the power y
//	Floor Division	x/y	The division of operands where the result is the quotient in which the digits after the decimal point are removed.

See the following statements in Python Shell.



The screenshot shows a Python Shell window titled "Python Shell" with a menu bar (File, Edit, Shell, Debug, Options, Windows, Help). The shell is running Python 3.2.3. The user has entered several commands to demonstrate arithmetic operations on variables x and y.

```

Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = 14
>>> y = 5
>>> x + y # Addition
19
>>> x - y # Subtraction
9
>>> x * y # Multiplication
70
>>> x/y # Division
2.8
>>> x%y # Modulus
4
>>> x**y # Exponent
537824
>>> x//y # Floor Division
2
>>> |

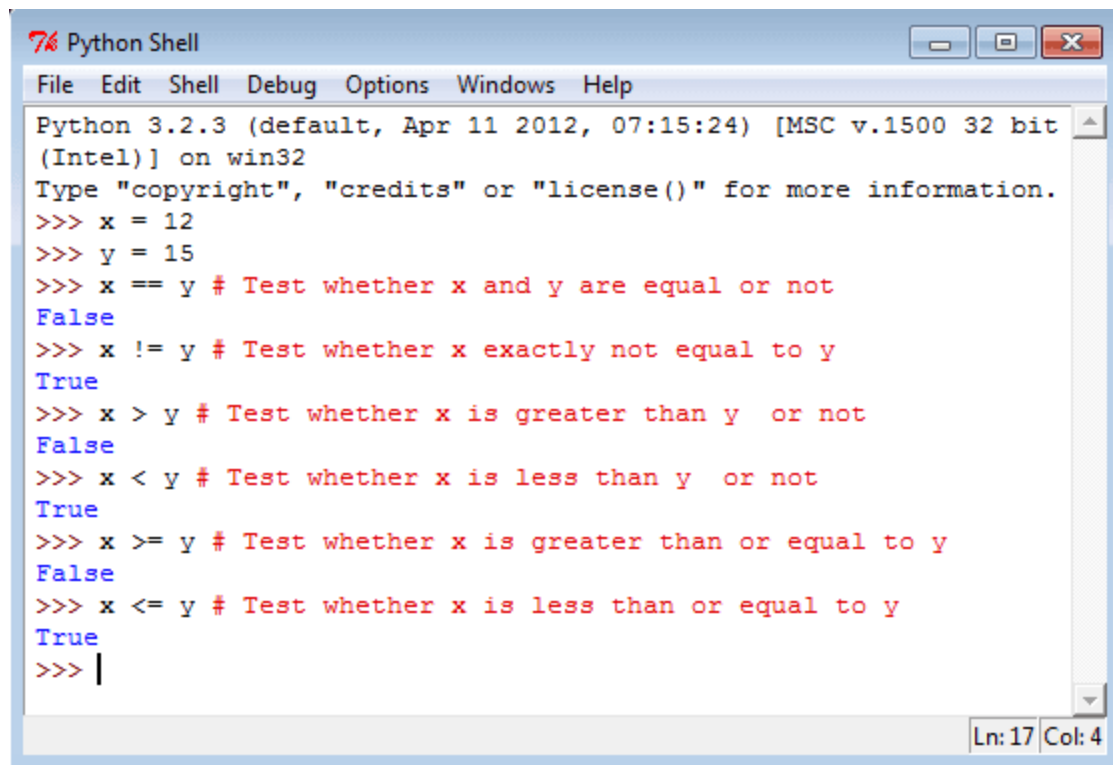
```

The status bar at the bottom right indicates "Ln: 19 Col: 4".

Python Comparison Operators

Operator	Name	Example	Result
==	Equal	x==y	True if x is exactly equal to y.
!=	Not equal	x!=y	True if x is exactly not equal to y.
>	Greater than	x>y	True if x (left-hand argument) is greater than y (right-hand argument).
<	Less than	x<y	True if x (left-hand argument) is less than y (right-hand argument).
>=	Greater than or equal to	x>=y	True if x (left-hand argument) is greater than or equal to y (left-hand argument).
<=	Less than or equal to	x<=y	True if x (left-hand argument) is less than or equal to y (right-hand argument).

See the following statements in Python Shell.

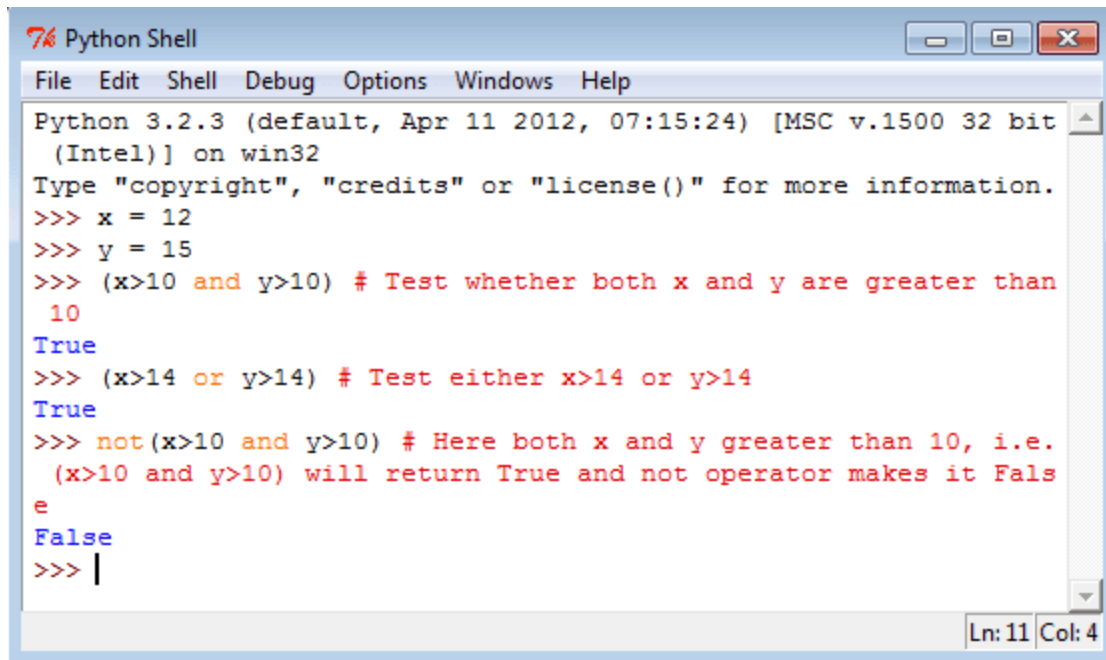


```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = 12
>>> y = 15
>>> x == y # Test whether x and y are equal or not
False
>>> x != y # Test whether x exactly not equal to y
True
>>> x > y # Test whether x is greater than y or not
False
>>> x < y # Test whether x is less than y or not
True
>>> x >= y # Test whether x is greater than or equal to y
False
>>> x <= y # Test whether x is less than or equal to y
True
>>> |
```

Python Logical Operators

Operator	Example	Result
and	(x and y)	is True if both x and y are true.
or	(x or y)	is True if either x or y is true.
not	(x not y)	If a condition is true then Logical not operator will make false.

See the following statements in Python Shell.



The screenshot shows a Python Shell window with the following content:

```
Python 3.2.3 (default, Apr 11 2012, 07:15:24) [MSC v.1500 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> x = 12
>>> y = 15
>>> (x>10 and y>10) # Test whether both x and y are greater than
10
True
>>> (x>14 or y>14) # Test either x>14 or y>14
True
>>> not(x>10 and y>10) # Here both x and y greater than 10, i.e.
(x>10 and y>10) will return True and not operator makes it Fals
e
False
>>> |
```

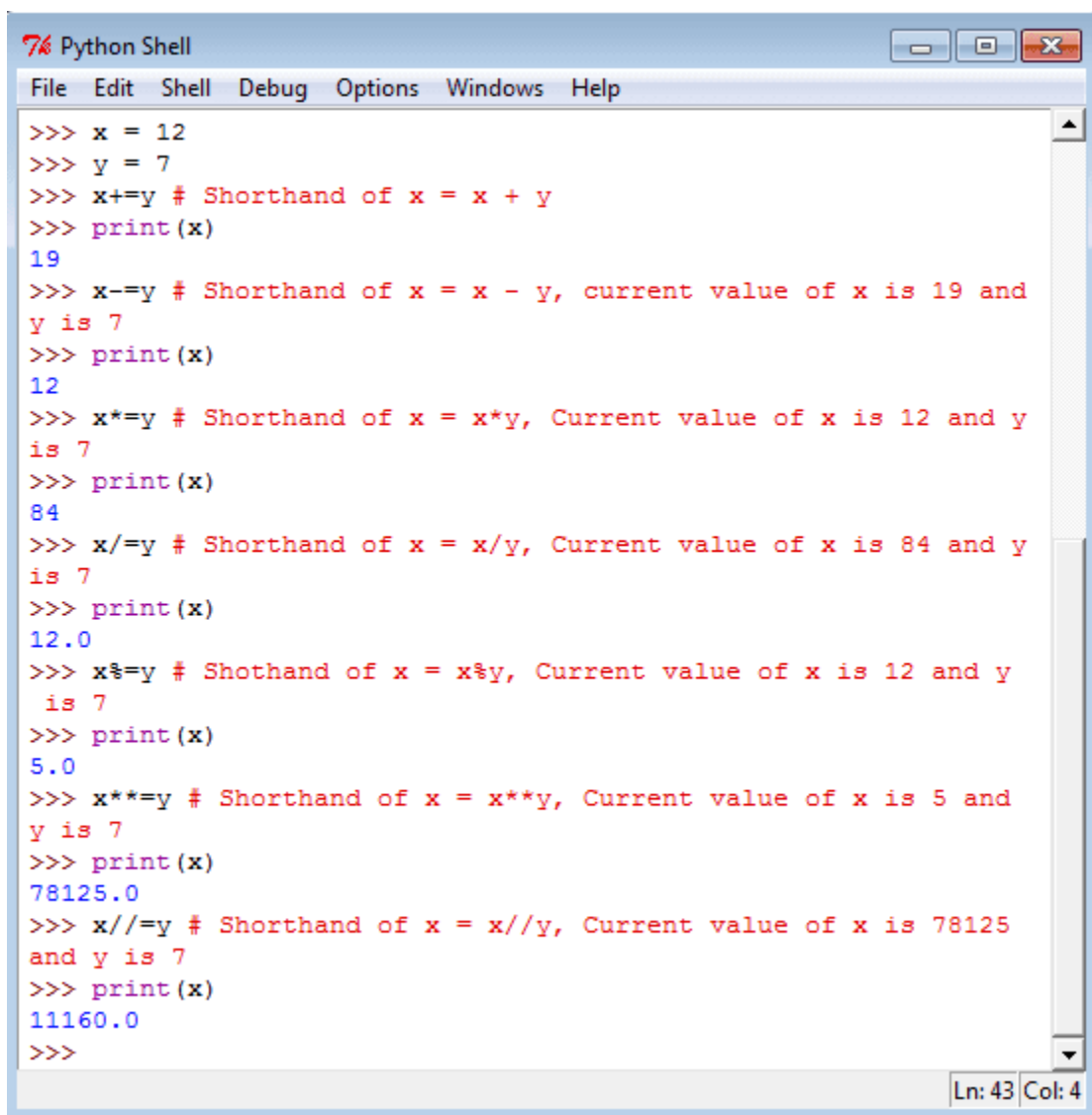
The status bar at the bottom right indicates 'Ln: 11 Col: 4'.

Python Assignment Operators

Operator	Shorthand	Expression	Description
+=	x+=y	x = x + y	Adds 2 numbers and assigns the result to left operand.
-=	x-= y	x = x -y	Subtracts 2 numbers and assigns the result to left operand.
=	x= y	x = x*y	Multiplies 2 numbers and assigns the result to left operand.
/=	x/= y	x = x/y	Divides 2 numbers and assigns the result to left operand.
%=	x%= y	x = x%y	Computes the modulus of 2 numbers and assigns the result to left operand.

<code>**=</code>	<code>x**=y</code>	<code>x = x**y</code>	Performs exponential (power) calculation on operators and assign value to the equivalent to left operand.
<code>//=</code>	<code>x//=y</code>	<code>x = x//y</code>	Performs floor division on operators and assign value to the left operand.

See the following statements in Python Shell.



```

Python Shell
File Edit Shell Debug Options Windows Help
>>> x = 12
>>> y = 7
>>> x+=y # Shorthand of x = x + y
>>> print(x)
19
>>> x-=y # Shorthand of x = x - y, current value of x is 19 and
y is 7
>>> print(x)
12
>>> x*=y # Shorthand of x = x*y, Current value of x is 12 and y
is 7
>>> print(x)
84
>>> x/=y # Shorthand of x = x/y, Current value of x is 84 and y
is 7
>>> print(x)
12.0
>>> x%=y # Shorthand of x = x%y, Current value of x is 12 and y
is 7
>>> print(x)
5.0
>>> x**=y # Shorthand of x = x**y, Current value of x is 5 and
y is 7
>>> print(x)
78125.0
>>> x//=y # Shorthand of x = x//y, Current value of x is 78125
and y is 7
>>> print(x)
11160.0
>>>
Ln: 43 Col: 4

```

Python Bitwise Operators

Operator	Shorthand	Expression	Description
&	And	x & y	Bits that are set in both x and y are set.
	Or	x y	Bits that are set in either x or y are set.
^	Xor	x ^ y	Bits that are set in x or y but not both are set.
~	Not	~x	Bits that are set in x are not set, and vice versa.
<<	Shift left	x <<y	Shift the bits of x, y steps to the left
>>	Shift right	x >>y	Shift the bits of x, y steps to the right.

Each step means 'multiply by two'

* Each step means 'divide by two'

Conditional Operators

Conditional expressions or ternary operator have the lowest priority of all Python operations. The expression x if C else y first evaluates the condition, C (not x); if C is true, x is evaluated and its value is returned; otherwise, y is evaluated and its value is returned.

Operator precedence

Operator precedence determines how operators are parsed concerning each other. The following table summarizes the operator precedence in Python, from lowest precedence (least binding) to highest precedence (most binding).

Operators in the same box have the same precedence. Unless the syntax is

explicitly given, operators are binary. Operators in the same box group left to right (except for exponentiation, which groups from right to left).

Note: Comparisons, membership tests, and identity tests, all have the same precedence and have a left-to-right chaining feature as described in the Comparisons section.

Operator Name	Description
<code>:=</code>	Assignment expression
<code>lambda</code>	Lambda expression
<code>if – else</code>	Conditional expression
<code>or</code>	Boolean OR
<code>and</code>	Boolean AND
<code>not x</code>	Boolean NOT
<code>in, not in, is, is not, <, <=, >, >=, !=, ==</code>	Comparisons, including membership tests and identity tests
<code> </code>	Bitwise OR
<code>^</code>	Bitwise XOR
<code>&</code>	Bitwise AND

<<, >>	Shifts
+, -	Addition and subtraction
*, @, /, //, %	Multiplication, matrix multiplication, division, floor division, remainder
+x, -x, ~x	Positive, negative, bitwise NOT
**	Exponentiation
await x	Await expression
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
(expressions...), [expressions...], {key: value...}, {expressions...}	Binding or parenthesized expression, list display, dictionary display, set display