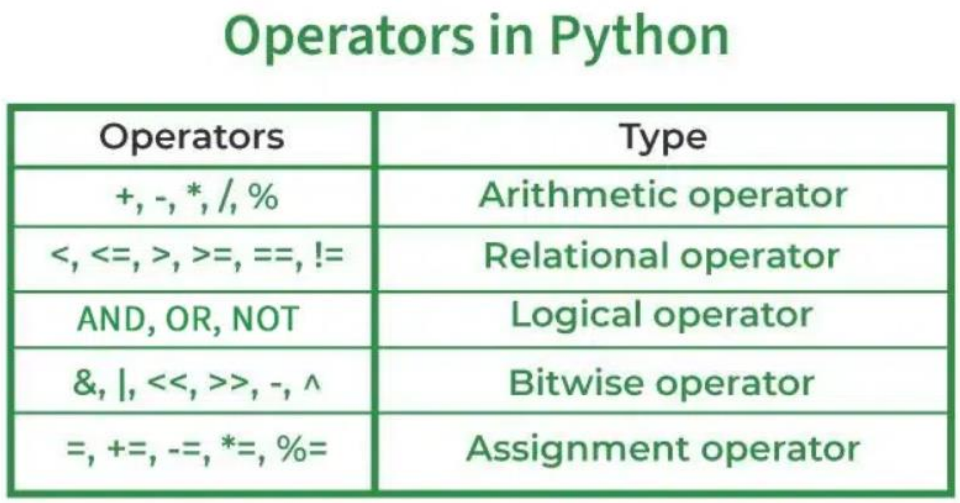
Python Operators

In Python programming, Operators in general are used to perform operations on values and variables. These are standard symbols used for logical and arithmetic operations. In this article, we will look into different types of Python operators.

OPERATORS: These are the special symbols. Eg- + , * , /, etc.
OPERAND: It is the value on which the operator is applied.

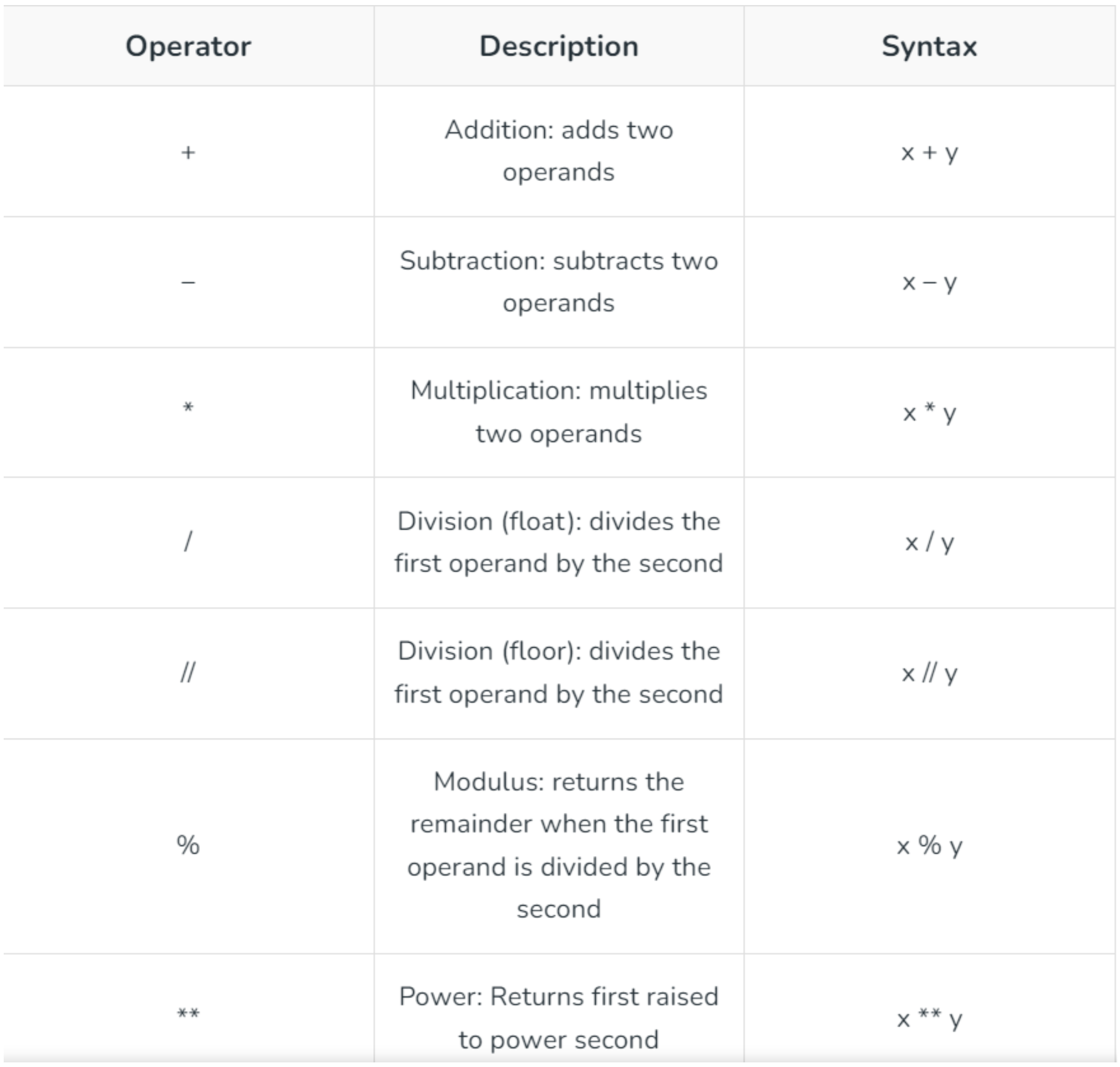


Operators in Python

Operators	Type
+, -, *, /, %	Arithmetic operator
<, <=, >, >=, ==, !=	Relational operator
AND, OR, NOT	Logical operator
&, |, <<, >>, -, ^	Bitwise operator
=, +=, -=, *=, %=	Assignment operator

Arithmetic Operators in Python

Python Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, and division.



Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Here are examples of arithmetic operators in Python:

Addition (+):

```
a = 5
b = 3
result = a + b
print(result)  # Output: 8
```

Subtraction (-):

```
a = 10
b = 4
result = a - b
print(result)  # Output: 6
```

Multiplication (*):

```
a = 7
b = 6
result = a * b
print(result)  # Output: 42
```

Division (/):

```
a = 20
b = 4
result = a / b
print(result)  # Output: 5.0
```

Floor Division (//):

```
a = 15
b = 4
result = a // b
print(result)  # Output: 3
```

Modulus (%):

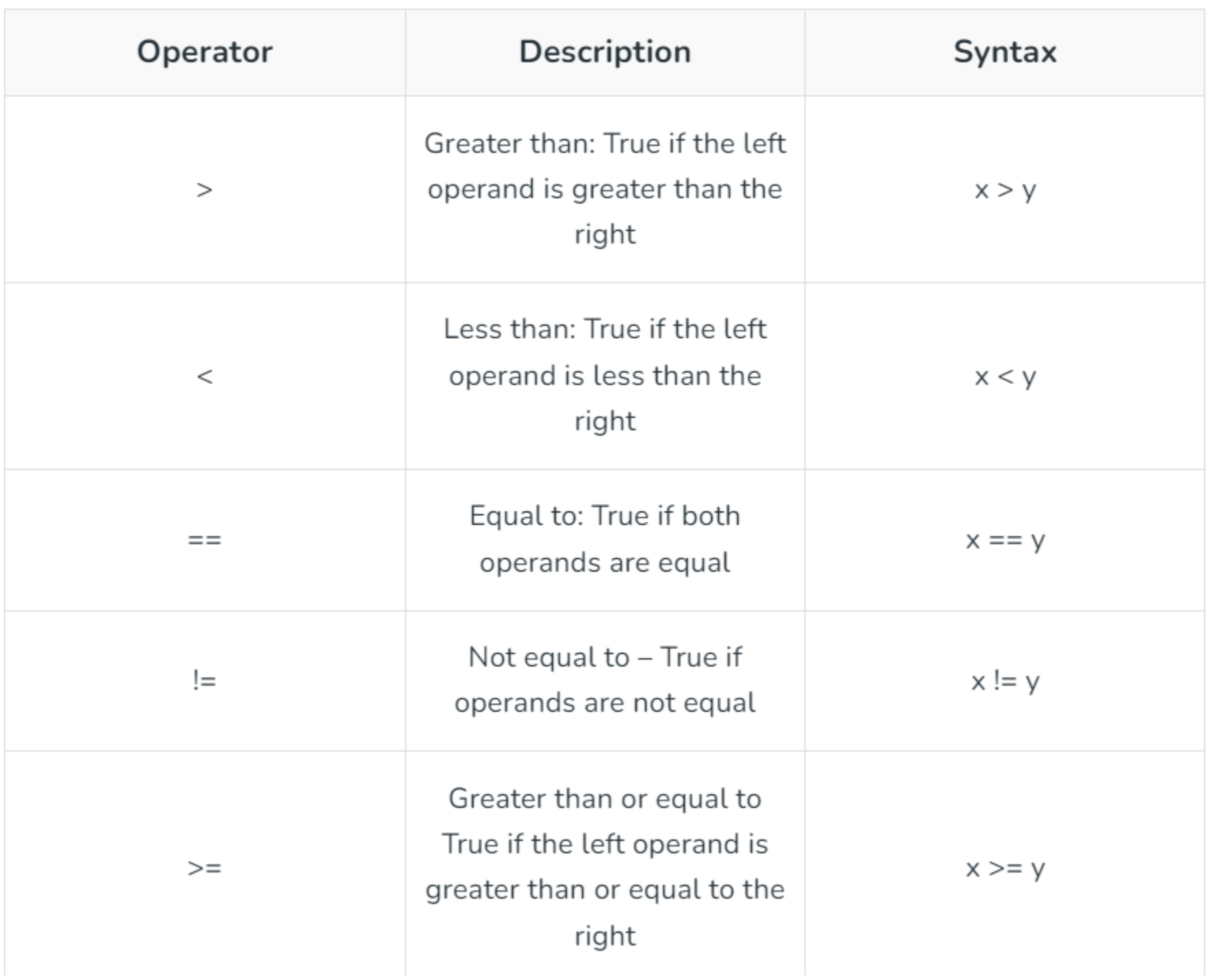
```
a = 17
b = 5
result = a % b
print(result)  # Output: 2
```

Exponentiation (**):

```
a = 3
b = 4
result = a ** b
print(result)  # Output: 81
```

Relational/Comparison Operators in Python

In Python Comparison of Relational operators compares the values. It either returns True or False according to the condition.



Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to True if the left operand is greater than or equal to the right	x >= y

<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y	

Example: The code compares the values of ‘a’ and ‘b’ using various comparison Python operators and prints the results. It checks if ‘a’ is greater than, less than, equal to, not equal to, greater than, or equal to, and less than or equal to ‘b’.

```
a = 13
b = 33

print(a > b)
print(a < b)
print(a == b)
print(a != b)
print(a >= b)
print(a <= b)
```

Output

```
False
True
False
True
False
True
```

Logical Operators in Python

Python Logical operators perform Logical AND, Logical OR, and Logical NOT operations. It is used to combine conditional statements.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

Example: The code performs logical operations with Boolean values. It checks if both ‘a’ and ‘b’ are true (‘and’), if at least one of them is true (‘or’), and negates the value of ‘a’ using ‘not’. The results are printed accordingly.

```
a = True
b = False
print(a and b)
print(a or b)
print(not a)
```

Output

```
False
True
False
```

Bitwise Operators in Python

Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Example: The code demonstrates various bitwise operations with the values of ‘a’ and ‘b’. It performs bitwise AND (&), OR (|), NOT (~), XOR (^), right shift (>>), and left shift (<<) operations and prints the results. These operations manipulate the binary representations of the numbers.

```
a = 10
b = 4
print(a & b)
print(a | b)
print(~a)
print(a ^ b)
print(a >> 2)
print(a << 2)
```

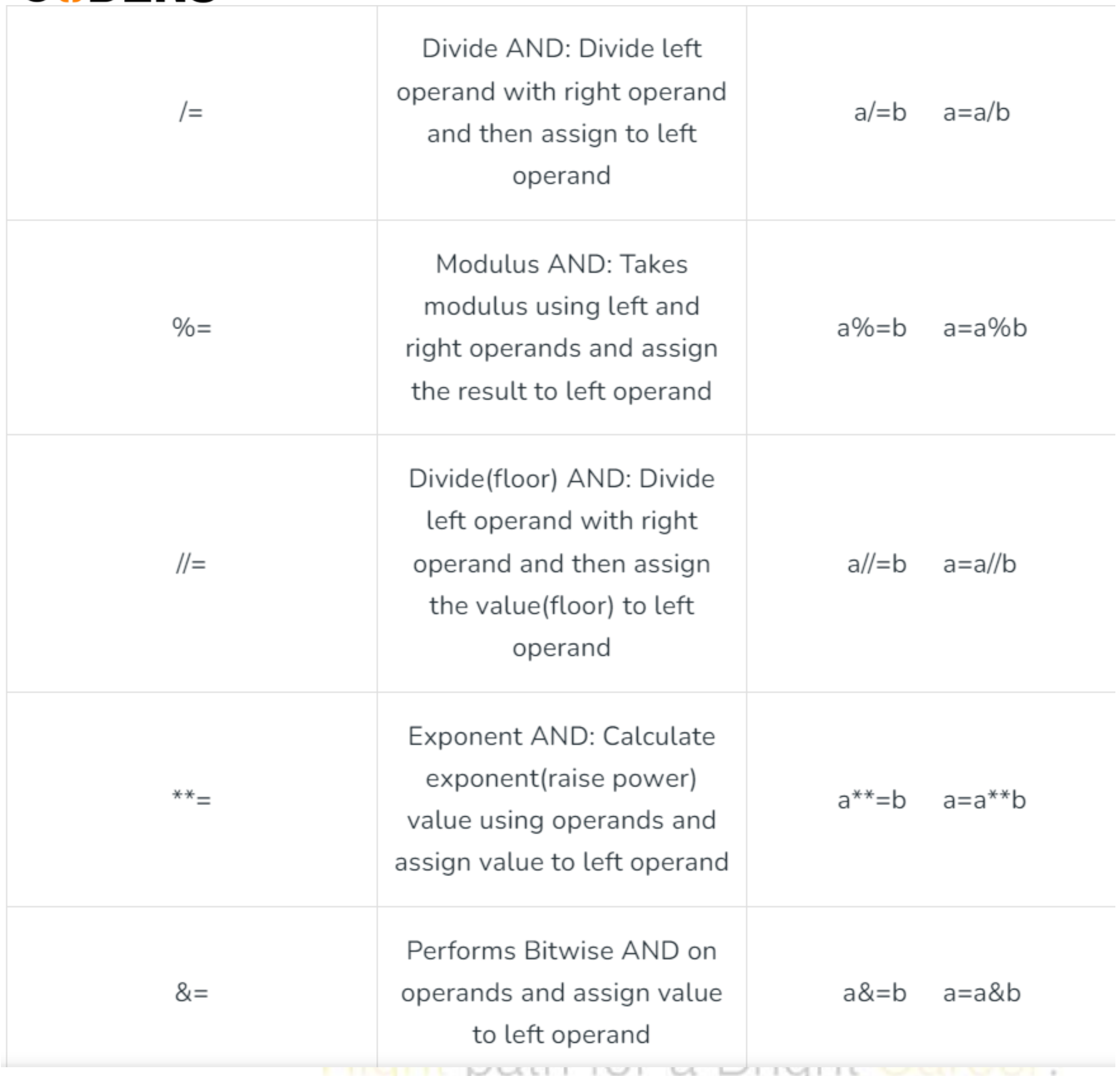
Output

```
0
14
-11
14
2
40
```

Assignment Operators in Python

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

Operator	Description	Syntax
=	Assign the value of the right side of the expression to the left side operand	x = y + z
+=	Add AND: Add right-side operand with left-side operand and then assign to left operand	a+=b a=a+b
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b a=a*b



/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b	a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b	a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b	a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b	a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b	a=a&b

Example: The code starts with ‘a’ and ‘b’ both having the value 10. It then performs a series of operations: addition, subtraction, multiplication, and a left shift operation on ‘b’. The results of each operation are printed, showing the impact of these operations on the value of ‘b’.

```
a = 10
b = a
print(b)
b += a
print(b)
b -= a
print(b)
b *= a
print(b)
b <<= a
print(b)
```

Output

```
10
20
10
100
102400
```