Python - Assignment Operators

Python Assignment Operator

The = (equal to) symbol is defined as assignment operator in Python. The value of Python expression on its right is assigned to a single variable on its left. The = symbol as in programming in general (and Python in particular) should not be confused with its usage in Mathematics, where it states that the expressions on the either side of the symbol are equal.

Example of Assignment Operator in Python

Consider following Python statements –

```
a = 10
b = 5
a = a + b
print (a)
```

At the first instance, at least for somebody new to programming but who knows maths, the statement "a=a+b" looks strange. How could a be equal to "a+b"? However, it needs to be reemphasized that the = symbol is an assignment operator here and not used to show the equality of LHS and RHS.

Because it is an assignment, the expression on right evaluates to 15, the value is assigned to a.

In the statement "a+=b", the two operators "+" and "=" can be combined in a "+=" operator. It is called as add and assign operator. In a single statement, it performs addition of two operands "a" and "b", and result is assigned to operand on left, i.e., "a".

Augmented Assignment Operators in Python

In addition to the simple assignment operator, Python provides few more assignment operators for advanced use. They are called cumulative or augmented assignment operators. In this chapter, we shall learn to use augmented assignment operators defined in Python.

Python has the augmented assignment operators for all arithmetic and comparison operators.

Python augmented assignment operators combines addition and assignment in one statement. Since Python supports mixed arithmetic, the two operands may be of different types. However, the type of left operand changes to the operand of on right, if it is wider.

Example

The += operator is an augmented operator. It is also called cumulative addition operator, as it adds "b" in "a" and assigns the result back to a variable.

The following are the augmented assignment operators in Python:

- Augmented Addition Operator
- Augmented Subtraction Operator
- Augmented Multiplication Operator
- Augmented Division Operator
- Augmented Modulus Operator
- Augmented Exponent Operator
- Augmented Floor division Operator

Augmented Addition Operator (+=)

Following examples will help in understanding how the "+=" operator works −

Open Compiler

```
a=10
b=5
print ("Augmented addition of int and int")
a+=b # equivalent to a=a+b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented addition of int and float")
a+=b  # equivalent to a=a+b
print ("a=",a, "type(a):", type(a))

a=10.50
b=5+6j
print ("Augmented addition of float and complex")
```

```
a+=b #equivalent to a=a+b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented addition of int and int
a= 15 type(a): <class 'int'>
Augmented addition of int and float
a= 15.5 type(a): <class 'float'>
Augmented addition of float and complex
a= (15.5+6j) type(a): <class 'complex'>
```

Augmented Subtraction Operator (-=)

Use -= symbol to perform subtract and assign operations in a single statement. The "a-=b" statement performs "a=a-b" assignment. Operands may be of any number type. Python performs implicit type casting on the object which is narrower in size.

</> Open Compiler

```
a=10
b=5
print ("Augmented subtraction of int and int")
a-=b #equivalent to a=a-b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented subtraction of int and float")
a-=b #equivalent to a=a-b
print ("a=",a, "type(a):", type(a))

a=10.50
b=5+6j
print ("Augmented subtraction of float and complex")
a-=b #equivalent to a=a-b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented subtraction of int and int
a= 5 type(a): <class 'int'>
Augmented subtraction of int and float
a= 4.5 type(a): <class 'float'>
Augmented subtraction of float and complex
a= (5.5-6j) type(a): <class 'complex'>
```

Augmented Multiplication Operator (*=)

The "*=" operator works on similar principle. "a*=b" performs multiply and assign operations, and is equivalent to "a=a*b". In case of augmented multiplication of two complex numbers, the rule of multiplication as discussed in the previous chapter is applicable.

Open Compiler

```python
a=10
b=5
print ("Augmented multiplication of int and int")
a*=b #equivalent to a=a*b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented multiplication of int and float")
a*=b #equivalent to a=a*b
print ("a=",a, "type(a):", type(a))

a=6+4j
b=3+2j
print ("Augmented multiplication of complex and complex")
a*=b #equivalent to a=a*b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented multiplication of int and int
a= 50 type(a): <class 'int'>
Augmented multiplication of int and float
a= 55.0 type(a): <class 'float'>
```

```
Augmented multiplication of complex and complex
a= (10+24j) type(a): <class 'complex'>
```

Augmented Division Operator (/=)

The combination symbol "/=" acts as divide and assignment operator, hence "a/=b" is equivalent to "a=a/b". The division operation of int or float operands is float. Division of two complex numbers returns a complex number. Given below are examples of augmented division operator.

Open Compiler

```python
a=10
b=5
print ("Augmented division of int and int")
a/=b #equivalent to a=a/b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented division of int and float")
a/=b #equivalent to a=a/b
print ("a=",a, "type(a):", type(a))

a=6+4j
b=3+2j
print ("Augmented division of complex and complex")
a/=b #equivalent to a=a/b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented division of int and int
a= 2.0 type(a): <class 'float'>
Augmented division of int and float
a= 1.8181818181818181 type(a): <class 'float'>
Augmented division of complex and complex
a= (2+0j) type(a): <class 'complex'>
```

Augmented Modulus Operator (%=)

To perform modulus and assignment operation in a single statement, use the %= operator. Like the mod operator, its augmented version also is not supported for complex number.

Open Compiler

```
a=10
b=5
print ("Augmented modulus operator with int and int")
a%=b #equivalent to a=a%b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented modulus operator with int and float")
a%=b #equivalent to a=a%b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented modulus operator with int and int
a= 0 type(a): <class 'int'>
Augmented modulus operator with int and float
a= 4.5 type(a): <class 'float'>
```

Augmented Exponent Operator (**=)

The "**=" operator results in computation of "a" raised to "b", and assigning the value back to "a". Given below are some examples −

Open Compiler

```
a=10
b=5
print ("Augmented exponent operator with int and int")
a**=b #equivalent to a=a**b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
```

```
print ("Augmented exponent operator with int and float")
a**=b #equivalent to a=a**b
print ("a=",a, "type(a):", type(a))

a=6+4j
b=3+2j
print ("Augmented exponent operator with complex and complex")
a**=b #equivalent to a=a**b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented exponent operator with int and int
a= 100000 type(a): <class 'int'>
Augmented exponent operator with int and float
a= 316227.7660168379 type(a): <class 'float'>
Augmented exponent operator with complex and complex
a= (97.52306038414744-62.22529992036203j) type(a): <class 'complex'>
```

Augmented Floor division Operator (//=)

For performing floor division and assignment in a single statement, use the "//=" operator. "a//=b" is equivalent to "a=a//b". This operator cannot be used with complex numbers.

Open Compiler

```
a=10
b=5
print ("Augmented floor division operator with int and int")
a//=b #equivalent to a=a//b
print ("a=",a, "type(a):", type(a))

a=10
b=5.5
print ("Augmented floor division operator with int and float")
a//=b #equivalent to a=a//b
print ("a=",a, "type(a):", type(a))
```

It will produce the following **output** −

```
Augmented floor division operator with int and int
a= 2 type(a): <class 'int'>
Augmented floor division operator with int and float
a= 1.0 type(a): <class 'float'>
```