**Arithmetic Operators in Python**

The ***Arithmetic Operators*** in Python are:

| Operator | Meaning | Example |
| --- | --- | --- |
| + | addition | 4 + 5 = 9 |
| - | subtraction | 10 – 8 = 2 |
| / | division | 5/2 = 2.5 |
| \* | multiplication | 4 \* 5 = 20 |
| \*\* | exponent | 4\*\*2 = 16 |
| // | floor division | 14//4 = 3 |
| % | modulus | 14%4 = 2 |

The first five operators should need no explanation. The last two are as follows:

***floor division***: This divides two numbers and truncates the answer (removes the decimal part of the answer). In other words, it gives your answer to the nearest low integer. It seems like a strange operation, but it turns out to be something we do a lot in programming.

Here is an example with regular division:

a = 20.0

b = 7.0

c = a/b # divides two floating point variables

print (c)

Here is an example with floor division:

a = 20.0

b = 7.0

c = a//b # floor-divides two floating point variables

print (c)

Notice that the answer is not rounded according to standard rules (i.e. anything greater than 0.5 gets rounded up). Floor division always rounds *down* (to the floor). There is also a type of rounding that always rounds *up* – this is called ceiling rounding. Why don’t we have a ceiling rounding operator? The answer is simple: if you want to find the ceiling, first find the floor. And then add 1.

floor = 20//8 ceiling = 20//8 + 1

**Modulus**

The last operator in the table is the ***modulus*** operator. This operator finds the remainder of a division. For example:

20 / 6 = 3 *remainder 2*

Why is this? 20 divides by 6 three times. 3 x 6 is 18, so there is two “left over” to make it 20. The modulus will find the remainder of any division for us. Here’s how it looks:

a = 20

b = 6

r = a%b # r = remainder of 20 / 6

print (r)

In this case, the variable r represents a remainder, but we can use any variable to calculate the remainder.

See if you can predict the output:

a = 20

b = 4

c = a%b

print (c)

As you can see from the example above, when one number divides into another, the remainder must be zero. This is a useful fact.

Overloaded Operators

Some of the arithmetic operators are able to work with strings. What does this do:

a = "hello"

b = 4

print (a\*b) # overloaded multiplication

Now try this:

a = "hello"

b = "goodbye"

print (a+b) # overloaded addition

The process of adding strings together is called concatenation.

FInally, how about this...

a = "hello"

b = "goodbye"

print (a\*b) # overloaded multiplication ?

And this?

a = 5

b = "goodbye"

print (a+b) # overloaded addition ?

The last two examples don’t work, because the computer doesn’t know how to add a word and a number, or multiply two words.

Exercises

1. Predict the output of this code:

b = 5.0

c = 3.0

print (b\*c)

print (b\*\*c)

print (b/c)

print (b//c)

print (b%c)

Copy and run from your python editor. Be sure to ask if you don’t understand the results.

Key Terms: ***integer, string, floating point, arithmetic operator, floor division, modulus, concatenation, overloaded operator.***