One thing computers are capable of doing exceptionally well is performing arithmetic. Addition, subtraction, multiplication, division, and other numeric calculations are easy to do in most programming languages, and Python is no exception. Some examples:

mirthful\_addition = 12381 + 91817 amazing\_subtraction = 981 - 312 trippy\_multiplication = 38 \* 902 happy\_division = 540 / 45 sassy\_combinations = 129 \* 1345 + 120 / 6 - 12

Above are a number of arithmetic operations, each assigned to a variable. The variable will hold the final result of each operation. Combinations of arithmetical operators follow the usual [order of operations](https://en.wikipedia.org/wiki/Order_of_operations).

Python also offers a companion to division called the modulo operator. The modulo operator is indicated by % and returns the remainder after division is performed.

is\_this\_number\_odd = 15 % 2 is\_this\_number\_divisible\_by\_seven = 133 % 7

In the above code block, we use the modulo operator to find the remainder of 15 divided by 2. Since 15 is an odd number the remainder is 1.

We also check the remainder of 133 / 7. Since 133 divided by 7 has no remainder, 133 % 7 evaluates to 0.

**Numbers**

Variables can also hold numeric values. The simplest kind of number in Python is the integer, which is a whole number with no decimal point:

int1 = 1 int2 = 10 int3 = -5

A number with a decimal point is called a *float*. You can define floats with numbers after the decimal point or by just including a decimal point at the end:

float1 = 1.0 float2 = 10. float3 = -5.5

You can also define a float using scientific notation, with e indicating the power of 10:

# this evaluates to 150: float4 = 1.5e2

**Two Types of Division**

In Python 2, when we divide two integers, we get an integer as a result. When the quotient is a whole number, this works fine:

quotient = 6/2 # the value of quotient is now 3, which makes sense

However, if the numbers do not divide evenly, the result of the division is truncated into an integer. In other words, the quotient is rounded down to a whole number. This can be surprising when you expect to receive a decimal and you receive a rounded-down integer:

quotient = 7/2 # the value of quotient is 3, even though the result of the division here is 3.5

To yield a float as the result instead, programmers often change either the numerator or the denominator (or both) to be a float:

quotient1 = 7./2 # the value of quotient1 is 3.5 quotient2 = 7/2. # the value of quotient2 is 3.5 quotient3 = 7./2. # the value of quotient3 is 3.5