# Expressions Have Types

coursera.org/learn/programming-fundamentals/supplement/SCzBA/expressions-have-types

In the Expressions lesson, we learned that expressions are evaluated to values—if you have a+b*2, the current value of b is read out of its box, multiplied by 2, then the value of a is read out of its box, and added to the product of b*2. The expression evaluates to the resulting sum.

Expressions also have types, which are determined by the types of the sub-expressions that make them up. The simplest expressions are constants, which have type int if they are integer constants (e.g., 2 or 46), or type double if they are real constants (e.g., 3.14, or -8.19). The types of constants can be modified by applying a letter suffix if needed (U for unsigned, L for long, and f for float): 3.14f is a constant with type float, and 999999999999L is a constant with type long int. The next simplest type of expression is a variable, which has whatever type it was declared to have.

Most (but not all) expressions with binary operators—e1 op e2 (e.g., a + b or c * 4)—have the same type as their operands. If a and b are doubles, then a + b is a double as well. Likewise, if c is an int, then c * 4 is also an int (note that 4 is an int).

The type of a function is its declared return type. That is, if you have

```
int f (int x, int y) { … }

int g (double d, char c) { … }
```

then the expression **f(3, 4)** + **g(42.6, 'a')** has type int. We can see this from the fact that **f(3, 4)** has type int (f is declared to return an int), as does **g(42.6, 'a')**. As we just discussed, adding two ints results in an int.