Slide 1

Administrivia

- Reminder: Homework 2 due today, 11:59pm.

  A request: When you turn in homework by e-mail, please do say on the subject line which assignment and which course. (I'm teaching more than one in which students turn in homework this way.)
- Reminder: Quiz 1 Thursday. Open book, open notes, topics from chapters 3 and 4. Possibly questions such as "what does this Scala code do?" or "sketch some Scala code to do X", both to be answered without just using the REPL.
- (E-mail about non-classroom machine updates.)

Slide 2

Functions in Programming — Motivation

- Writing the same, or similar, code over and over is tedious and error-prone. Can we somehow capture frequently-done computations in a way we can reuse?
- Non-trivial programs can be huge (as much as tens of millions of lines of code for an operating system, e.g.). Humans are not (usually?) very good at understanding large and complicated things — need some way to "manage complexity".

Slide 3

Functions in Programming

- Functions are one way to solve both problems (code reuse and managing complexity). Most if not all programming languages provide some way to do this (possibly under another name, e.g., "procedures").
- Similar, but not identical, to functions in mathematics.

  In math, a function has a domain and a range, and maps elements of the domain to elements of the range.

  In programming, a function's domain is represented by the number and types of its *parameters* (a.k.a. arguments), and its range is represented by a *return type*.
- A key difference is that functions in programming can have "side effects" — effects other than mapping input(s) to output(s).

Slide 4

Functions in Scala

- To create a function in Scala, you use the keyword `def` and then give the function's name, parameters, return type, and some code. Return type can be omitted if the function will be used only for its side effects.
- Very simple examples:

```
def sum(x : Int, y : Int) : Int = x + y
def hello() { println("hello") }
```

Slide 5

Functions in Scala, Continued

- To use a function, give its name and values for parameters.
- Very simple examples:

  `sum(10, 2)`

  `hello()` (can omit parentheses)

Slide 6

Example — Previous Programs Revisited

- Now look again at the grades program. It starts by prompting for several input values. It might be nice to include in the prompt a maximum value. So we would be writing similar code over and over. Let's make a function for that.
- We could also use a function to "refactor" the making-change program once more — not changing its behavior but simplifying the repetitive parts of the code.

Slide 7

Example — Calculating Bounding Boxes

- Something that's of interest in graphics programs is finding the smallest rectangle that encloses one or more other shapes — a "bounding box".
- Let's try writing a function that computes a bounding box for two rectangles. We'll represent rectangles as tuples of four `Int`s — the x and y coordinates of the top left corner and the width and height.
- (Tuples were discussed in chapter 3. Briefly, tuples in Scala are similar to tuples in math — ordered lists of elements, of fixed size. The syntax for making one in Scala is to enclose one or more values in parentheses.)
- For now just write the function, load into REPL with `:load`, and test interactively.

Slide 8

Minute Essay

- What (if anything!) was interesting, difficult, noteworthy, whatever, about Homework 2?