Slide 1

Administrivia

- Reminder: Homework 2 due Thursday. Office hours tomorrow and Thursday if you need help (and if you're new to programming you very well may).

  Script on "sample programs" page that you can use to turn in homework from the command line. (Might be useful if you're working remotely using PuTTY.)
- Sample solutions for quizzes, and code from class, will usually be on the Web sometime after class.

Slide 2

Conditional Execution — Review/Recap

- "If/else" allows for conditional execution of statements and also for expressing similar idea within expressions.
- Look again at "compute grade" example from last time — a reasonable sketch (which we could fill in later with complete list of assignments and letter grades), but not as human-readable as it might be. (Remember that in writing source code you're generally writing for two audiences — the compiler/interpreter, and any human readers.)
- Notice also that there seems to be some repetition. That's a good lead-in to . . .

Slide 3

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 4

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 5

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 6

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)

Example — Grades Program Revisited

Slide 7

- 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.
- (Also, let's give names to those maximum values, to make the code more understandable to humans.)

Example — Finding Roots of a Quadratic Equation

Slide 8

- As a rather math-y example, let's write a function to compute and print the roots of a quadratic equation

$$ax^2 + bx + c = 0$$

- We'll use the formula

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

and try to account for as many cases as we can . . .

- (We will just write the function for now, and test it interactively — load it into the REPL with `:load`.)

Minute Essay

- How did the quiz compare to your expectations? with regard to difficulty and topic?
- Do you have any questions about the material so far?